Classical planning and causal implicatures

Luciana Benotti[1] and Patrick Blackburn[2]

[1] NLP Team, Universidad Nacional de Córdoba, Argentina
`benotti@famaf.unc.edu.ar`,
`http://cs.famaf.unc.edu.ar/~luciana/`
[2] Department of Philosophy, Roskilde University, Denmark
`patrickb@ruc.dk`
`http://www.patrickblackburn.org/`

Abstract. In this paper we motivate and describe a dialogue manager (called Frolog) which uses classical planning to infer causal implicatures. A causal implicature is a type of Gricean relation implicature, a highly context dependent form of inference. As we shall see, causal implicatures are important for understanding the structure of task-oriented dialogues. Such dialogues locate conversational acts in contexts containing both pending tasks and the acts which bring them about. The ability to infer causal implicatures lets us interleave decisions about "how to sequence actions" with decisions about "when to generate clarification requests"; as a result we can model task-oriented dialogue as an interactive process locally structured by negotiation of the underlying task. We give several examples of Frolog-human dialog, discuss the limitations imposed by the classical planning paradigm, and indicate the potential relevance of our work for other relation implicatures.

1 Introduction

In conversation, an important part of the content conveyed is not explicitly stated, rather it is *implicated*. However, Grice's [11] classic concept of *conversational implicature* (CI) is far from fully understood. Traditionally, CIs have been classified using the Gricean maxims: there are *relation CIs* (also known as relevance CIs), *quantity CIs*, *quality CIs* and *manner CIs*. In linguistics, the most widely studied CIs are quantity CIs, probably because they are the ones most obviously amenable to context-independent analysis; see [10] for a survey of the state of the art. Far less studied are relation CIs, but these are arguably the most interesting of all. For a start, relation CIs are the most obviously context-dependent type of implicature, so studying them is important if we want to understand contextual reasoning. Moreover, it has been argued that all other types of CIs can be viewed as relation CIs [23]. Whether or not this is correct, it is undeniable that the maxim of relation ("be relevant") collapses a mixed bag of implicatures, which differ mainly in the the kind of contextual "relation" driving the inference. So gaining a more precise understanding of relation implicatures is an important task, and this paper is a step towards their computational formalization. We shall analyze a kind of relation CI that we call *causal CIs* and we will work in *task-oriented dialogues* where there is a clear notion of *task-domain causality*. Consider the following example:

Mary: The chest is locked, the crown is inside
Bill: Give me the crown
Bill causally implicated: Open the chest

To spell this out a little, in order to carry out Bill's request (giving him the crown) it is necessary to open the chest. Hence Bill is implicating, by trading on the domain causal relations (after all, the contents of a chest are not accessible unless the chest is open) that Mary is to open the chest. To put it another way, Bill has tacitly conveyed (by exploiting contextual knowledge of the task domain) that Mary should carry out an "open the box" subtask [21]. What are Mary's options once she has inferred this causal CI? There are two clear choices: to accept this subtask silently or to negotiate it. Mary might decide to silently accept it (that is, she might simply open the chest without further ado) because she has the key that unlocks the chest or knows how to get it; in such cases we say that Mary has constructed an *internal bridge* from the current task situation (the crown being inside the locked chest) to the proposal made by Bill (giving him the crown). On the other hand, Mary might decide that she has insufficient information to construct the internal bridge (perhaps she has no key, or sees that the lock is rusty) so she may make a *clarification request*, such as *But how can I open the chest?* We call such a subdialogue an *external bridge*. The internal process of bridging (silent acceptance) is often called *accommodation* [14] or (plain old) *bridging* [8]. The external process of bridging, on the other hand, constitutes an important part of conversation.

A real task-oriented dialogue situation is typically made up of a (seemingly effortless) interplay of internal and external bridging as the dialogue participants explore the task at hand. The main claims of the paper are that understanding causal implicature is crucial to understanding this "seemingly effortless" interactive process, and that the inferences required can (at least to a first approximation) be modeled computationally in the classical planning paradigm.

Our motivation is both theoretical and practical. On the theoretical side, we believe that it is crucial to explore CIs in real dialogue settings. Strangely enough (after all, Grice did call them *conversational* implicatures) this view appears to be novel, perhaps even controversial. In the formal pragmatics literature, CIs are often simply viewed as inferences drawn by a hearer on the basis of a speaker's utterance and the Gricean maxims. We find this perspective too static. CIs (especially relation CIs) are better viewed as intrinsically *interactional* inferences, arising from the dynamics of conversations and situated in changing contexts. As conversations progress, speakers and hearers switch roles, meaning are negotiated, and contexts are updated via a grounding process [22]. Moreover, even within a single turn, hearers are not restricted to simply drawing (or failing to draw) "the" CI; in fact, choosing between internal and external bridging is better viewed as part of a process of *negotiating what the CI at stake actually is.*

As a result of the negotiation, inferred CIs are grounded in the evolving context. We believe that modeling the changing context, and the inferences drawn from it, is necessary to extend the theory of CIs beyond the relatively narrow domain of quantity CIs towards the more challenging problems posed by relation CIs. We also believe that the dialogue-centered approach we advocate may have practical consequences. In particular, we believe that modeling external bridging is an important step towards defining an incremental dialogue manager (DM) in the spirit of that sketched in [6]; such a DM would be able to handle clarification requests in a principled way. This explains our interest in classical planning [17]. If the causal inferences can be computed using the well-understood and computationally-efficient technology this paradigm offers, we are well on the way to having a practical tool for dialogue management . The rest of the paper presents the dialogue manager Frolog, which infers *causal* CIs in task-oriented dialogue; Frolog is intended as a proof-of-concept of the ideas just sketched.

The paper proceeds as follows. In Section 2, we motivate the study of causal CIs by showing that their inference is critical for dialogues situated in physical task situations. In Section 3 we present the computational model for inferring causal CI using classical planning that underlies Frolog. In Section 4 we examine in detail the kinds of external and internal bridging that Frolog can (and cannot) handle. Section 5 concludes.

2 Causal implicatures and physical tasks

In this paper we focus on causal implicatures. There are two main reasons for this. First, we view talking as a special case of purposive behavior. And so did Grice; indeed, he even showed that his maxim of relation (the maxim governing relation CIs) was relevant to *physical* acts:

> *I expect a partner's contribution to be appropriate to immediate needs at each stage of the transaction; if I am mixing ingredients for a cake, I don't expect to be handed a good book, or even an oven cloth (though this might be an appropriate contribution at a later stage).* [11, page 47]

The maxim of relation is traditionally viewed as important but obscure (it simply says: "Be relevant"). But if we restrict our attention to causal CIs arising in dialogues situated in a task domain, we get a useful parallelism between talk and actions: both are purposive behavior, and both are governed by the same maxim. This gives rise to an informational interplay between the task level and the dialogue level that we can exploit computationally.

Second, causal CIs give us an *empirical* handle on CIs. It is not controversial that (in non-conversational activities) the causal relations between acts define the expectations of the interaction. But it turns out that the same holds when conversational activities are situated in a physical task: there too causal relations guide the interaction. We did an empirical study on a task-oriented dialogue corpus [5] and found that most CIs which were made explicit—by being externally bridged as clarification requests—could be explained in terms of causal CIs. Let us briefly review this work before introducing the Frolog dialogue manager.

2.1 Empirically studying implicatures using the SCARE corpus

For our empirical study, we annotated and classified clarification requests (CRs) that appear in the SCARE corpus [19]. This corpus consists of fifteen spontaneous English dialogues situated in an instruction giving task. It was collected using the Quake environment, a first-person virtual reality game.

In the corpus, one of the dialogue participants plays the role of the direction giver (DG), who gives instructions to the other participant, the direction follower (DF) on how to complete several tasks in the game world. The DF has no prior knowledge of either the world map or the tasks and thus must rely on his partner, the DG, to guide him in playing the game. The DG has a map of the world and a list of tasks to complete. As the participants collaborate on the tasks, the DG has instant feedback of the DF's location in the simulated world. So the corpus is a treasure trove of lengthy task-oriented dialogues set in a richly-structured and well-understood situation.

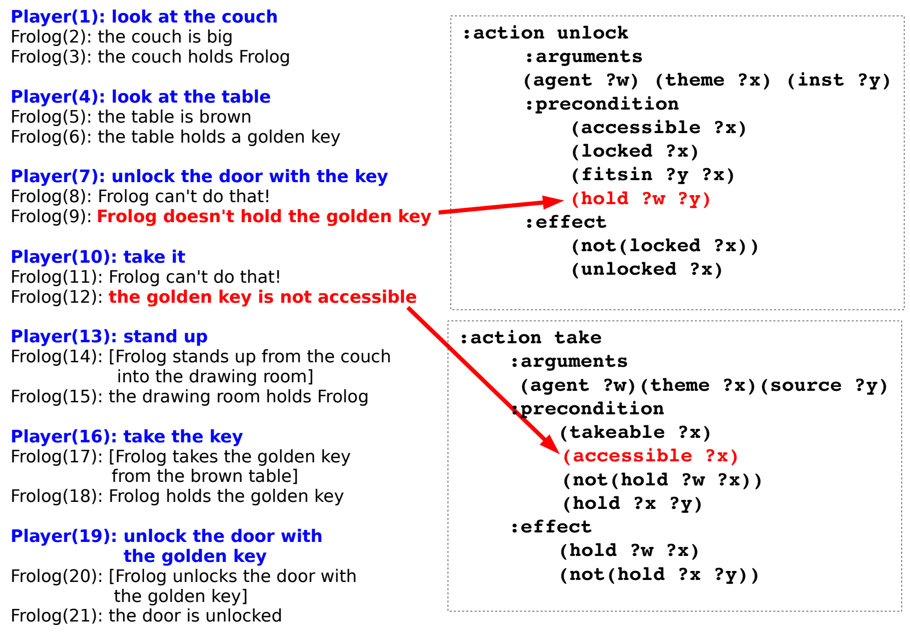
We randomly selected one dialogue; its transcript contains 449 turns. We classified the clarification requests according to the four-level model of communication independently developed by Clark [9] and Allwood [2]. We found 29 clarification requests (CRs). Of these, 65% indicated an obstacle in carrying out the act proposed by the source utterance, and thus belong to level 4 of the model. As for the rest, 31% belonged to level 3 (indicating an obstacle in reference resolution) while 4% belonged to level 2 (indicating acoustic problems).

We agree with Thomason's view [20] that CIs are added to the context in order to *eliminate obstacles to the conversation*. Therefore, as we argue in [4], when CIs cannot be internally bridged, they are made explicit as level 4 CRs. To put it another way: *CRs in level 4 make explicit the CIs that cannot be silently accommodated*. We then classified the CRs in level 4 using the Gricean maxims. We found that *all* of them made explicit relation CIs (no other kinds of CIs were found). Finally, we further classified the relation CI according to the kind of relation involved: causal relations, at 73%, were far and away the most frequent.

2.2 Computationally studying causal implicatures using Frolog

How do we infer causal CIs of the sort found in the SCARE corpus? And what kinds of mechanisms lead to their being made explicit in CRs? To investigate such questions, we designed a system that mimics the type of instruction giving found in the SCARE corpus. In our setup, the DF is a dialogue system that we call Frolog. The role of the DG is played by a human who we call "the player".

Let's start right away with a sample interaction: we'll walk through the example reproduced in Figure 1. Incidentally, the point of this first example is to show just how unnatural interactions can be if a dialogue participant *can't* internally bridge causal implicatures. In the example, Frolog is in a room with a locked door, sitting on a big couch. The player is asking Frolog to look around (turns (1) to (6)) when Frolog says that there is a golden key on a table in the room. Then (in turn (7)) the player tells Frolog “Unlock the door with the key.” In order to be executed in the virtual world, the action “unlock” has several preconditions (specified in the definition of the action “unlock”, shown in the upper right-hand side the figure). One precondition is that Frolog must be holding the instrument required by the action. But this does *not* hold in the current state of the virtual world (the key is on the table). The fact that there is an obstacle to carrying out the action is made explicit by Frolog, who verbalizes the failed precondition of the unlock action (turns (8) and (9)). So the player tries to overcome this obstacle by making the causal CI “take the key!’ explicit in turn (10), only to find that another obstacle gets in the way: this time, the key is not accessible (turn (12)) because it is on the table and Frolog is sitting on the couch! Again the player is forced to externally bridge the intended CI, and so says “Stand up”. Since all the preconditions of the “stand up” action hold, Frolog is able to do this (turn (15)) and so the context is changed with the effects of the action. Once Frolog is standing in the drawing room the key is accessible, and so the command “take the key” in turn (16) can be directly executed and the context updated with the effects of this action (turn (18)). Finally, all the preconditions of the action “unlock the chest with the key” hold and Frolog can execute this command (turns (19) to (21)) without inferring any CI.



Player(1): look at the couch
Frolog(2): the couch is big
Frolog(3): the couch holds Frolog

Player(4): look at the table
Frolog(5): the table is brown
Frolog(6): the table holds a golden key

Player(7): unlock the door with the key
Frolog(8): Frolog can't do that!
Frolog(9): **Frolog doesn't hold the golden key**

Player(10): take it
Frolog(11): Frolog can't do that!
Frolog(12): **the golden key is not accessible**

Player(13): stand up
Frolog(14): [Frolog stands up from the couch into the drawing room]
Frolog(15): the drawing room holds Frolog

Player(16): take the key
Frolog(17): [Frolog takes the golden key from the brown table]
Frolog(18): Frolog holds the golden key

Player(19): unlock the door with the golden key
Frolog(20): [Frolog unlocks the door with the golden key]
Frolog(21): the door is unlocked

```
:action unlock
     :arguments
     (agent ?w) (theme ?x) (inst ?y)
     :precondition
         (accessible ?x)
         (locked ?x)
         (fitsin ?y ?x)
         (hold ?w ?y)
     :effect
         (not(locked ?x))
         (unlocked ?x)
```

```
:action take
    :arguments
     (agent ?w)(theme ?x)(source ?y)
    :precondition
        (takeable ?x)
        (accessible ?x)
        (not(hold ?w ?x))
        (hold ?x ?y)
    :effect
        (hold ?w ?x)
        (not(hold ?x ?y))
```

Fig. 1. Frolog *without* internal bridging

Much ado about nothing, and all to get a door unlocked! This example illustrates the (decidedly unhelpful) behavior of a dialogue system that does not internally bridge causal CIs. In such a situation we would expect a human conversational partner to simply stand up, take the key, and unlock the chest—no muss, no fuss. Frolog, however, forces the player to do all the work: the player does all the bridging unaided.

This gives rise to a clear feeling of non-cooperativeness; it's as though Frolog is being willfully unhelpful. But Frolog is not malicious—he simply does not have the capabilities to bridge internally, as he lacks the ability to reason about the context in which he finds himself. So what happens if we give Frolog these capabilites? Figure 2 shows the result.[3]

```
Player(1): look at the couch
Frolog(2): the couch is big
Frolog(3): the couch holds Frolog

Player(4): look at the table
Frolog(5): the table is brown
Frolog(6): the table holds a golden key

Player(7): unlock the door
           with the golden key
Frolog(8): [Frolog stands up from the couch
            into the drawing room]
Frolog(9): the drawing room holds Frolog
Frolog(10): [Frolog takes the golden key
             from the brown table]
Frolog(11): Frolog holds the golden key
Frolog(12): [Frolog unlocks the door
             with the golden key]
Frolog(13): the door is unlocked
```

```
:definition here
  (some hold player)
:definition accessible
  (or here
      (some (inv hold) here)
      (some (inv hold)
            (and accessible open)))
```

```
:action standup
    :arguments
     (agent ?a)(source ?s)(goal ?g)
    :precondition
        (hold ?s ?a)
        (seated ?a)
        (hold ?g ?s)
    :effect
        (not(hold ?s ?a))
        (hold ?g ?a)
```

Fig. 2. Frolog *with* internal bridging

Adding reasoning capabilities has resulted in a bright new Frolog who behaves more like a collaborative human partner. But what exactly are these capabilites, and how do they enable Frolog to use contextual knowledge to infer the appropriate causal CIs?

3 Causal implicatures and classical planning

In a nutshell, Frolog uses classical planning to compute causal implicatures. That is, Frolog uses classical planning (a well-explored and reasonably efficient AI technique) to fill out the micro-structure of discourse (the bridging information

[3] Figure 2 also shows the definitions and action specifications which, together with those of Figure 1, complete the causal relations used in the example.

required in the next interactional step).[4] In particular, Frolog uses the classical planner FastForward [12]. Like all classical planners, FastForward takes three inputs—the initial state, the goal, and the available actions—and outputs a sequence of actions which, when executed in the initial state, achieve the goal. These three inputs are the crucial pieces of contextual information that Frolog must keep updated in order to use classical planning to infer appropriate CIs. Let's see what the content of each one should be.

3.1 The initial state

In Frolog, two types of information are recorded and maintained: complete and accurate information about the game world is kept in the *world KB*, and a representation of the common ground (the local context constructed during the interaction) is kept in the *common ground KB*. So: which KB should be used as the initial state? As it turns out, we need both.

To see this, let's modify our running example. Suppose that the golden key (last seen lying on a table) is taken by a thief without either Frolog or the player realizing it. As a consequence, in the common ground KB the key is still (incorrectly) recorded as being on the table, whereas the world KB (correctly) notes that the thief has it. Now suppose the player issues the command "Unlock the door with the golden key" in this scenario. If we included in the initial state the complete information recorded in the game KB, Frolog would automatically take the key from the thief (for example, by using the "steal" action) and unlock the door. But Frolog should *not* be able do this—after all, Frolog does not know where the key actually is! So Frolog should *not* be able to use the world KB in order to infer the appropriate CIs.

But what happens if we use the common ground KB instead? In this case, Frolog would decide to take the key from the table and use it to unlock the door. But this sequence of actions is *not* executable in the game world because the key is no longer accessible (the thief has it). That is, a sequence of causal CIs found by reasoning over the common ground KB might not be executable in the game world because the common ground KB may contain information inconsistent with respect to the world KB. Hence Frolog needs both KBs: he *infers* the actions intended by the player using the information in the common ground KB but he has to *verify* this sequence of actions on the world KB to check that it can actually be executed. The importance of the verification step is an instance of what we call the *Causality Clarification Principle (CCP)*:

> *Causal CIs become explicit when they cannot be carried out in the* context *in which the conversation is situated.*

Frolog implements this principle by attempting to execute the CIs in the virtual world KB which contains complete information. If the execution fails he will

[4] Since we use AI planning, the work reported here is very different from the classical work on plan-based dialogue managers [18, 1]. The classic work uses *plan recognition* (a computationally expensive task) to interpret utterances by inserting them into the plan the macro-structure (the global shape) of discourse.

trigger the external bridging of the CI (for instance by saying *"The key is not on the table"*).

The CI inference step can also fail because Frolog does not have enough game experience. That is, he has not collected enough information (in the common ground KB) to enable him to devise, all on his own, a sequence of CIs that will modify the context so that the player's command can be successfully executed. In such cases, Frolog will start a process of external bridging in which all the required CIs will be explicitly negotiated. Section 4 explores the circumstances that make external bridging necessary in situated dialogue.

3.2 The goal

In the Frolog dialogue manager, the goal of the planning problem is not given in advance, nor does it have to be inferred during the dialogue (as is the case in plan recognition approaches [15, 7]). For in our approach it is not the whole dialogue that is a hierarchical plan, rather *each utterance can be interpreted as a plan that links the utterance to the context.* In order to come up with the appropriate CIs of an instruction, Frolog should simply act to make the preconditions of the instruction true. So we can define the goal as the conjunction of all the preconditions of the command uttered by the player.

3.3 The actions

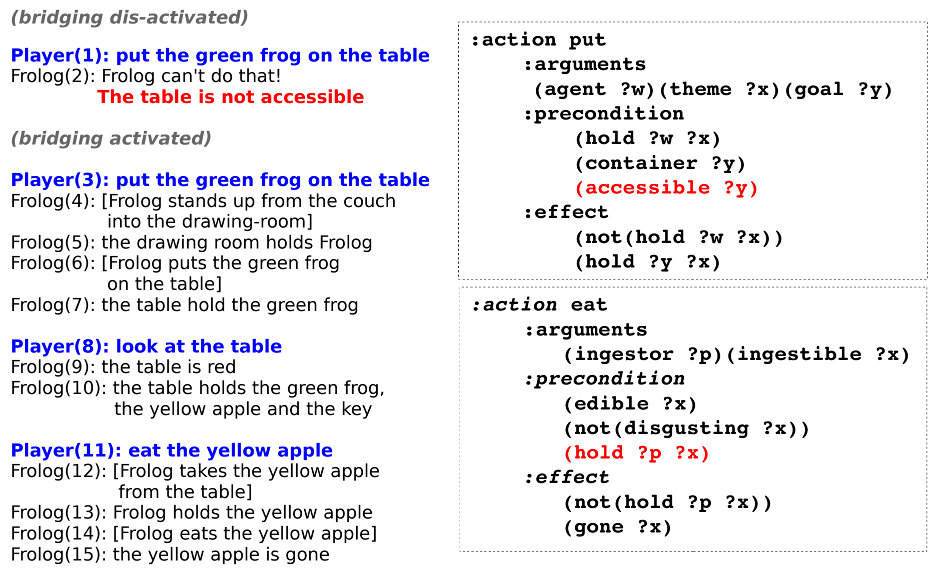
To complete the picture, the actions available to the planner are all the actions in the game action database. We assume that all the action schemes that can be executed, such as the ones in Figure 1 and Figure 2, are mutually known to Frolog and the player. Clearly, in order for bridging to be possible, it must be mutually known what the preconditions and the effects of the actions involved are. In our unlock the door example, if the player doesn't know that Frolog needs to be holding the key to use it to unlock the door, then the player cannot possibly implicate that Frolog should take the key by saying "Unlock the door with the golden key". The same goes, *mutatis mutandis*, for Frolog.

The assumption that the player and Frolog know the exact specification of all the actions that can be executed in the game world is clearly a simplifying assumption. We leave for future work the task of modeling how differing knowledge about actions gets coordinated through dialogue; computationally coping with such issues is still an open problem [16, 13].

4 Negotiating implicatures

We now present several detailed examples of how solutions to the planning problems introduced in the previous section can (and cannot) be used to perform internal bridging and trigger external bridging (when appropriate). Incidentally, Frolog's bridging ability can be activated or deactivated at will, making it simple to compare its behavior with and without inference.

In the example in Figure 3, the player issued the command *put the green frog on the table* while Frolog was sitting on the couch. The action *put* requires its *goal* parameter, namely *the table*, to be accessible, but in this context the precondition does not hold. When bridging is deactivated, Frolog will simply verbalize the failed precondition. But as soon as the bridging abilities are activated (in the same context) the same command is bridged and executed by Frolog. Frolog makes explicit its internal bridging process by verbalizing each action and its effects in order, which is not very natural but is useful to monitor Frolog's internal inferences.



```
(bridging dis-activated)

Player(1): put the green frog on the table
Frolog(2): Frolog can't do that!
           The table is not accessible

(bridging activated)

Player(3): put the green frog on the table
Frolog(4): [Frolog stands up from the couch
            into the drawing-room]
Frolog(5): the drawing room holds Frolog
Frolog(6): [Frolog puts the green frog
            on the table]
Frolog(7): the table hold the green frog

Player(8): look at the table
Frolog(9): the table is red
Frolog(10): the table holds the green frog,
             the yellow apple and the key

Player(11): eat the yellow apple
Frolog(12): [Frolog takes the yellow apple
             from the table]
Frolog(13): Frolog holds the yellow apple
Frolog(14): [Frolog eats the yellow apple]
Frolog(15): the yellow apple is gone
```

```
:action put
    :arguments
     (agent ?w)(theme ?x)(goal ?y)
    :precondition
        (hold ?w ?x)
        (container ?y)
        (accessible ?y)
    :effect
        (not(hold ?w ?x))
        (hold ?y ?x)
```

```
:action eat
    :arguments
       (ingestor ?p)(ingestible ?x)
    :precondition
       (edible ?x)
       (not(disgusting ?x))
       (hold ?p ?x)
    :effect
       (not(hold ?p ?x))
       (gone ?x)
```

Fig. 3. *Internal* bridging in Frolog

It's worth noting that planning problems are generated on the fly each time a precondition does not hold. In the example shown in Figure 3, the state to which the command "put the green frog on the table" is bridged to is different from the state to which "eat the yellow apple" is bridged to (in this last state the green frog is by the yellow apple on the table).

As we said in Section 3, bridges are *inferred* using the information in the common ground KB, while their execution is *verified* with respect to the complete and accurate state of the world (kept in the world KB). So Frolog distinguishes two ways in which internal bridging can fail: when there is no bridge, and when there is a bridge—but it's not executable in the game world. We illustrated the case of a non-executable bridge in Section 3.1. We now illustrate the "no bridge" case with the interaction in Figure 4 (which continues the interaction in Figure 3). Here Frolog has just entered a room (called the treasury) which has a white wall, when the player issues command (20), "take the wall". In this case, the precondition that fails is `(takeable wall1)`. In AI planning terms, we say that the predicate `takeable` is *static* because there is no action that modifies its extension. In simpler terms, this means that there is no way of making the wall takeable. Therefore, as `(takeable wall1)` is a precondition, the planner will report that no plan can be found, and Frolog verbalizes the failed precondition. This failed-precondition verbalization could actually be considered a triggering of external bridging, at least if we dropped our assumption of completely synchronized action schemes; after all, maybe the player can teach Frolog how to use magic powers (or the strength of the incredible Hulk) to make a wall takeable.

```
Player(16): go through the south exit
Frolog(17): [Frolog goes through the south exit
             into a treasury]
Frolog(18): the treasury hold you, a chest
            and a dragon
Frolog(19): the treasury has a white wall

Player(20): take the wall
Frolog(21): Frolog can't do that!
Frolog(22): the wall is not takeable

Player(23): look at Frolog
Frolog(24): Frolog is alive
Frolog(25): Frolog holds the sword, the worm
            and a golden key

Player(26): kill the dragon with the sword
Frolog(27): [Frolog kills the dragon with
             the sword]
Frolog(28): the dragon is dead

Player(29): kill the dragon with the sword
Frolog(30): Frolog can't do that!
Frolog(31): the dragon is not alive
```

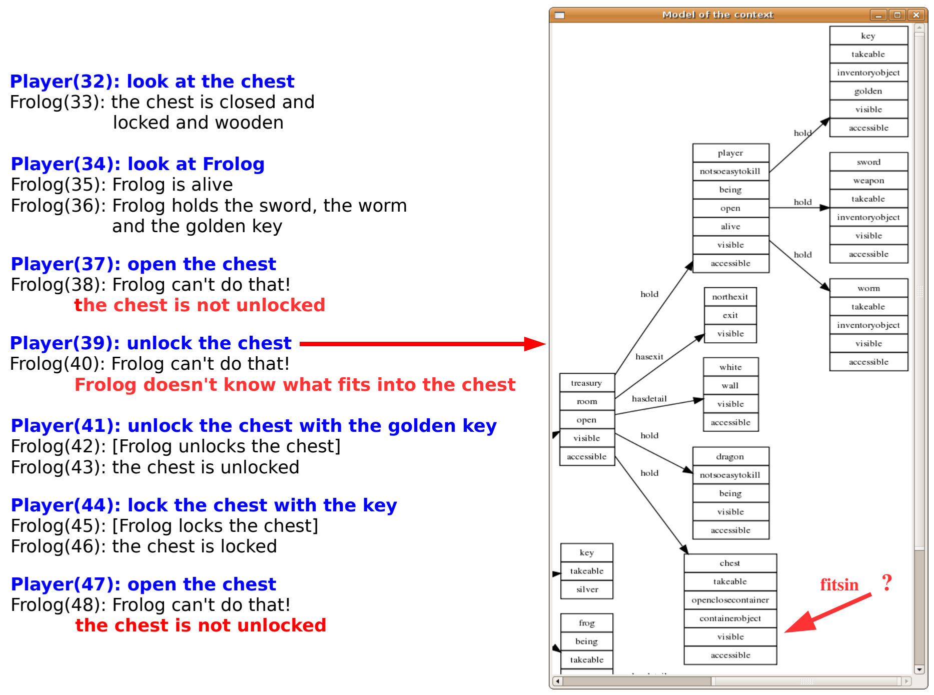
```
:action take
    :arguments
     (agent ?w)(theme ?x)(source ?y)
    :precondition
        (takeable ?x)
        (accessible ?x)
        (not(hold ?w ?x))
        (hold ?x ?y)
    :effect
        (hold ?w ?x)
        (not(hold ?x ?y))
```

```
:action kill
    :arguments
       (agent ?w)(theme ?p)(inst ?x)
    :precondition
       (alive ?p)
       (accessible ?p)
       (hold ?w ?x)
    :effect
       (dead ?p)
       (not (alive ?p))
```

Fig. 4. *External* bridging in Frolog (world constraints)

Instruction (29) is an example of an instruction that cannot be internally bridged but whose failed precondition involves an *irreversible action* (namely, killing) rather than a static predicate. In this case, the planner is invoked with the goal `(alive dragon1)` and the planner will not be able to find a plan: the dragon can be killed only once. Whenever the planner says there is no plan for all the preconditions in the goal, the planner will be invoked with each precondition separately. For at least one of these preconditions no plan will be found, and the first such precondition found will be verbalized. In our example, this results in Frolog saying *the dragon is not alive* in turn (31).

Now, it is not only when the predicate is static or the actions are irreversible that the planner will find no plan: it can also happen when Frolog has not acquired enough knowledge to infer the bridge. This is the case in instruction (37) in Figure 5. Here the player wants Frolog to open the chest—but Frolog does not know how to do this. In the screen-shot to the right you can see state of the common ground KB from turns (37) to (40). The player does not have information about what fits into the chest; this is why the planner is unable to find a plan that realizes the precondition `(unlocked chest1)` of the command "open the chest" in instruction (37).[5]



Player(32): look at the chest
Frolog(33): the chest is closed and locked and wooden

Player(34): look at Frolog
Frolog(35): Frolog is alive
Frolog(36): Frolog holds the sword, the worm and the golden key

Player(37): open the chest
Frolog(38): Frolog can't do that!
the chest is not unlocked

Player(39): unlock the chest
Frolog(40): Frolog can't do that!
Frolog doesn't know what fits into the chest

Player(41): unlock the chest with the golden key
Frolog(42): [Frolog unlocks the chest]
Frolog(43): the chest is unlocked

Player(44): lock the chest with the key
Frolog(45): [Frolog locks the chest]
Frolog(46): the chest is locked

Player(47): open the chest
Frolog(48): Frolog can't do that!
the chest is not unlocked

Fig. 5. *External* bridging in Frolog (knowledge constraints)

Let's take this example further. Frolog's behavior continues as expected until turn (48), when Frolog is again unable to bridge the command to open the chest. But at this stage it should clearly be mutual knowledge that the golden key fits into the chest, so why does Frolog not react appropriately? The problem is that the classical planning based architecture assumes that all the actions are *deterministic*, but actions that acquire information are *non*-deterministic: after the player tries to perform an unlock action with a key, he can end up either in a state where the key fits or one where it doesn't. That is, he must *learn* whether the key fits or not. Such learning cannot be modelled with deterministic actions.

In order to overcome this issue there are two options: (1) either we allow different sources to update the mutual information (thereby losing uniformity in

[5] Instruction (39) fails as well, making the obstacle explicit. Incidentally, utterance (40) is an example of a CR caused by a "required but unknown" thematic relation filler (the second most frequent CR in level 4 in the SCARE corpus).

the update process) or (2) we add non-deterministic actions. Here we describe option (1). Option (1) can be seen as a procedural version of "strict accommodation" (as defined in [3]). For a discussion of option (2), see [5].

The intuition behind the strict notion of accommodation is this: when the speaker utters a sentence, she is not only trying to achieve the obvious effects of the utterance, she is also conveying her image of how she takes the world to be. Crucially, the success of the utterance depends on letting this image "come into existence". Accordingly, we argue that the intention behind the utterance "unlock the chest with the golden key" in turn (41) is twofold. It is clear that the player wants to achieve the effects of the action (that is, she wants to have the chest unlocked). But the player also expects Frolog to recognize the assumptions she is making and on which the success of the utterance depends. In particular, she expects Frolog to recognize that she is assuming that the golden key fits into the chest's lock (this is why Frolog could coherently ask here: *but why do you think that the golden key will fit?*).

In particular, after the action (`unlock(door1 key1)`) is executed successfully, the player will believe that the assumption she made was confirmed by the success of the action. This means that, when an action is executed, the common ground KB will be updated not only with the *effects* of the action but also with its *preconditions*. And those preconditions that were not in the common ground KB before will be verbalized as in turn (9) in Figure 6.

Player(1): unlock the chest
Frolog(2): Frolog can't do that!
Frolog doesn't know
what fits-into the chest

Player(3): look at you
Frolog(4): you are alive
Frolog(5): you hold a golden key

Player(6): unlock the chest
with the golden key
Frolog(7): [Frolog unlocks the chest
with the golden key]
Frolog(8): the chest is unlocked
Frolog(9): **the key fits-into the chest**

Player(10): lock the chest
Frolog(11): [Frolog locks the chest
with the golden key]
Frolog(12): the chest is locked

Player(13): look at the chest
Frolog(14): the chest is closed and
locked and wooden

Player(15): open the chest
Frolog(16): [Frolog unlocks the chest with the golden key]
Frolog(17): the chest is unlocked
Frolog(18): [Frolog opens the chest]
Frolog(19): the chest is open
Frolog(20): the chest hold a golden crown

```
:action unlock
     :arguments
     (agent ?w) (theme ?x) (inst ?y)
     :precondition
         (accessible ?x)
         (locked ?x)
         (fitsin ?y ?x)
         (hold ?w ?y)
     :effect
         (not(locked ?x))
         (unlocked ?x)
```

```
:action open
     :arguments
     (agent ?w) (theme ?x)
     :precondition
         (accessible ?x)
         (closed ?x)
         (unlocked ?x)
     :effect
         (not(closed ?x))
         (open ?x)
```

Fig. 6. *External* bridging becomes *internal* as common ground accumulates

The rest of the interaction (from turns (10) to (15)) shows that once the proposition `(fitsin key1 chest1)` is added to the common ground KB, the action "open the chest" can be internally bridged, even when the chest is locked, because all the necessary information is in the common ground.

So what's the problem? This: the updates done during the execution of the actions no longer mimic those done when doing internal bridging. We have broken the symmetry of the update processes. And this means that solution (1) is only partial. Although Frolog can now learn during execution (as in Figure 6) he is not able to experiment to try and acquire information himself. For example, Frolog cannot decide on his own to try a key to see if it fits.

Solution (1) is able to model 78% of the causal implicatures observed in the SCARE corpus (see Section 2.1). For a more detailed discussion of cases beyond the scope of classical planning, see [5].

5 Conclusions

Causal implicatures are a kind of relation implicature (historically Grice's most obscure yet most crucial type) whose inference—we have argued—is essential to understanding how situated dialogue works. Causal relations certainly have a direct impact on the coherence structure of task-oriented dialogues (such as those in the SCARE corpus), but can their conversational effect be computed? Frolog was designed as a simple proof-of-concept to show that they can. How does it fare?

Rather well, we believe. On the practical side, our use of classical planning shows that a well-understood tool can be used to make inferences that model negotiations observed in human-human dialogue. Furthermore, the shortcomings of Frolog's inferential model are directly traceable to the restrictions build into classical planning; when more flexible partial planners become widely available, Frolog will be able to make use of them and the non-determinism they offer. But matters are more complex on the theoretical side. We believe our account shows what a genuinely interactive view of implicature could (and in our view, should) look like. But can we extend it from causal implicatures in task-oriented dialogues to other types of relational implicature in different dialogue settings? At present, this is unclear. The tight link between clausal implicatures and clarification requests in task-oriented dialogues makes empirical work relatively straightforward; implicatures in other settings can be much harder to analyze. Nonetheless, it is our belief that our approach will eventually throw further light on Grice's seminal work. In particular, it is our hypothesis that the following *Clarification Principle (CP)*—a generalization of the *CCP* noted earlier—will play a important role:

> *Relation Implicatures become explicit when they cannot be* grounded *in the context in which the conversation is situated.*

References

1. Allen, J., Allen, R.: Natural language understanding. Addison Wesley (1994)
2. Allwood, J.: An activity based approach to pragmatics. In: Abduction, Belief and Context in Dialogue: Studies in Computational Pragmatics, pp. 47–80. John Benjamins Publishers (1995)
3. Beaver, D., Geurts, B.: Presupposition. In: Handbook of Semantics. Mouton de Gruyter (in press)
4. Benotti, L.: Clarification potential of instructions. In: Proceedings of the 2009 SIGDIAL Conference on Discourse and Dialogue. pp. 196–205 (2009)
5. Benotti, L.: Implicature as an Interactive Process. Ph.D. thesis, Université Henri Poincaré, INRIA Nancy Grand Est, France (2010)
6. Buß O., Schlangen, D.: Modelling sub-utterance phenomena in spoken dialogue systems. In: The 2010 Workshop on the Semantics and Pragmatics of Dialogue. Poznań, Poland (2010)
7. Carberry, S., Lambert, L.: A process model for recognizing communicative acts and modeling negotiation subdialogues. Computational Linguistics 25(1), 1–53 (1999)
8. Clark, H.: Bridging. In: Proceedings of the 1975 workshop on Theoretical issues in natural language processing. pp. 169–174. ACL, Morristown, USA (1975)
9. Clark, H.: Using Language. Cambridge University Press, New York (1996)
10. Geurts, B.: Quantity implicatures. Cambridge University Press (2011)
11. Grice, P.: Logic and conversation. In: Cole, P., Morgan, J. (eds.) Syntax and Semantics, vol. 3, pp. 41–58. Academic Press, New York (1975)
12. Hoffmann, J., Porteous, J., Sebastia, L.: Ordered landmarks in planning. Journal of Artificial Intelligence Research 22, 215–278 (2004)
13. Larsson, S.: Coordinating on ad-hoc semantic systems in dialogue. In: The 2007 Workshop on the Semantics and Pragmatics of Dialogue (Decalog 2007) (2007)
14. Lewis, D.: Scorekeeping in a language game. Journal of Philosophical Logic 8, 339–359 (1979)
15. Litman, D., Allen, J.: Discourse processing and commonsense plans. In: Intentions in Communication, pp. 365–388. MIT Press (1990)
16. Mills, G.: Semantic co-ordination in dialogue: the role of direct interaction. Ph.D. thesis, Queen Mary, University of London (2007), supervised by Patrick Healey
17. Nau, D., Ghallab, M., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann Publishers Inc., CA, USA (2004)
18. Perrault, R., Allen, J.: A plan-based analysis of indirect speech acts. Computational Linguistics 6(3-4), 167–182 (1980)
19. Stoia, L., Shockley, D., Byron, D., Fosler-Lussier, E.: SCARE: A situated corpus with annotated referring expressions. In: Proceedings of LREC (2008)
20. Thomason, R.: Accommodation, meaning, and implicature: Interdisciplinary foundations for pragmatics. In: Cohen, P., Morgan, J., Pollack, M. (eds.) Intentions in Communication, pp. 326–363. MIT Press, Cambridge, Massachusetts (1990)
21. Thomason, R., Stone, M., DeVault, D.: Enlightened update: A computational architecture for presupposition and other pragmatic phenomena. In: Presupposition Accommodation. Ohio State Pragmatics Initiative (2006)
22. Traum, D.: A Computational Theory of Grounding in Natural Language Conversation. Ph.D. thesis, Computer Science Dept., U. Rochester, USA (1994)
23. Wilson, D., Sperber, D.: Relevance theory. In: Handbook of Pragmatics, pp. 607–632. Blackwell, Oxford (2004)