Foundations of Collaborative Task-Oriented Dialogue: What's in a Slot?[1]

Philip R. Cohen
Laboratory for Dialogue Research
Faculty of Information Technology
Monash University

Abstract

In this paper, we examine the foundations of task-oriented dialogues, in which systems are requested to perform tasks for humans. We argue that the way this dialogue task has been framed has limited its applicability to processing simple requests with atomic "slot-fillers". However, such dialogues can contain more complex utterances. Furthermore, situations for which it would be desirable to build task-oriented dialogue systems, e.g., to engage in collaborative or multiparty dialogues, will require a more general approach. In order to provide such an approach, we give a logical analysis of the "intent+slot" dialogue setting that overcomes these limitations.

1 Introduction

An important problem that forms the core for many current spoken dialogue systems is that of "slot-filling" — the system's ability to acquire required and optional attribute-values of the user's requested action, for example, finding the date, time, and number of people for booking a restaurant reservation, or the departure date, departure time, destination, airline, arrival date, arrival time, etc. for booking a flight (Bobrow et al., 1977, Zue et al., 1991). If a required argument is missing, the system asks the user to supply it. Although this may sound simple, building such systems is more complex than one might suppose. For example, real task-related dialogues may be constraint-based rather than slot-filling, and are usually collaborative, such that dialogue participants may together fill slots, and people go beyond what was literally requested to address higher-level goals.

In this paper, we discuss the limitations of the general slot-filling approach, and provide a formal theory that can be used not only to build slot-filling task-oriented dialogue systems, but also other types of dialogues, especially multiparty and collaborative ones. We argue first that without being explicit about the mental states and the logical forms that serve as their contents, systems are too tightly bound to the specific and limited conversational task of a single user's getting a system to perform an action.

1.1 Intent+Slots (I+S)

The spoken language community has been working diligently to enable users to ask systems to perform actions. This requires the system to recover the user's "intent" from the spoken language, meaning the action the system is being requested to perform, and the arguments needed to perform it, termed "slots".[2] The most explicit definition of "slot" we can find is from (Henderson, 2015) in describing the Dialog State Tracking Challenge (DSTC2/3):

> The slots and possible slot values of a slot-based dialog system specify its domain, i.e. the scope of what it can talk about and the tasks that it can help the user complete. The slots inform the set of possible actions the system can take, the possible semantics of the user utterances, and the possible dialog states… For each slot $s \in S$, the set of possible values for the slot is denoted Vs.

Henderson goes on to describe a system's dialog state and two potentially overlapping slot

[1] Inspired by Woods (1975), "What's in a Link: Foundations for Semantic Networks"
[2] See https://developer.amazon.com/docs/custom-skills/create-intents-utterances-and-slots.html for an example of the commercial interest in "intent + slots".

Proceedings of the SIGDial 2019 Conference, pages 198–209
Stockholm, Sweden, 11-13 September 2019.

types, so-called "informable" and "requestable" slots, denoted by sets S_{inf} and S_{req}, respectively.

> The term *dialog state* loosely denotes a full representation of what the user wants at any point from the dialog system. The dialog state comprises all that is used when the system makes its decision about what to say next. … the dialog state at a given turn consists of:

- The **goal constraint** for every informable slot $s \in S_{inf}$. This is an assignment of a value $v \in Vs$ that the user is specifying as a constraint, or a special value *Dontcare*, which means the user has no preference, or *None*, which means the user is yet to specify a valid goal for the slot.
- A set of **requested slots**, the current list of slots that the user has asked the system to inform. This is a subset of S_{req}.[3,4] (Henderson, 2015) …

Most papers in the field at best have informal definitions of "intent" and "slot". In order to clarify these concepts, we frame their definitions in a logic with a precise semantics. We find the following topics require further explication.

2 Limitations of Slot-Filling

2.1 Representation of Actions

The DSTC proposes a knowledge representation of actions with a fixed set of slots, and atomic values with which to fill them, such as reserve(restaurant=Mykonos, cuisine=Greek, Location = North) to represent the user's desire that the system reserve Mykonos, a Greek restaurant in the north of town, or reserve(restaurant=none, cuisine=Greek, Location = dontcare), which apparently says that the user wants the system to reserve a Greek restaurant anywhere. However, missing from this representation is the agent of the action. At a minimum, we need to be able to distinguish between the user's performing and the system's performing an action. Thus, such a representation cannot directly accommodate the user's saying "I want to eat at Guillaume" because the user is not explicitly requesting the system to perform an action.[5] Also missing are variables used as values, especially *shared* variables. This severely limits the kinds of utterances people can provide. For example, it would prevent the system from representing the meaning of "I want you to reserve that Greek restaurant in the north of Cambridge *that John ate at last week*."

2.2 Restrictions on Logical Forms (LFs)

Next, the slot-filling approach limits the set of logical forms the dialogue system can consider by requiring the user to supply an atomic value (including *Dontcare* and *None*) to fill a slot. For example, slot-filling systems can be trained to expect simple atomic responses like *"7pm"* to such questions as *"what time do you want me to reserve a table?"* However, I+S systems typically will not accept such reasonable responses as *"not before 7pm," "between 7 and 8 pm,"* or *"the earliest time available."* What's missing from these systems are true logical forms that employ a variety of relations and operators, such as and, or, not, all, if-then-else, some, every, before, after, count, superlatives, comparatives, as well as proper variables. Critically, adequate meaning representations are compositional often employing relative clauses, such as the LF underlying *"What are the three best Chinese or Japanese restaurants that are within walking distance of Century Link Field?"* Compositional utterances often require scoped representations, as in *"What is the closest parking to the Japanese restaurant nearest to the Space Needle?"* which has two superlative expressions, one embedded within the other. These phenomena are also problematic for requests, as in: *Book a table at the closest good Italian restaurant to the Orpheum Theater on Monday for 4 people.* Although current I+S systems cannot parse or represent such utterances (Ultes et al. 2018), complex logical forms such as those underlying the above can now be produced robustly from competent semantic parsers (e.g., (Duong et al., 2017; Wang et al., 2015)). What we claim is necessary is to move from an I+S representation language of actions with attributes and atomic values to a true logical form language with which to represent the meaning of users' utterances.

2.3 Explicit Attitudes

However, this is still not sufficient. The I+S approach, as incorporated into the DSTC 2 (Henderson, 2015), says that the dialogue state

[3] This appears to be the reverse of the definition in (Gašić et al., 2016, p. 557)

[4] At least implicitly, the DSTC must allow a distinguished symbol (e.g., '?') to indicate what slot values are being requested. Alternatively, we have seen request(<attribute>) with an unstated value, meaning the user is asking for the value of the attribute.

[5] In order to handle this as an indirect request, a system would need to reason about users' plans and how the system can help the user achieve them.

"loosely denotes a full representation of what the user wants at any point from the dialog system", but treats as implicit the desire attitude associated with the intent content. Thus, when a user says "I want you to reserve for Monday" the notion of "want" is taken to be just syntactic sugar and is generally thrown away, resulting in a representation that looks like this: inform(reserve(day = monday)). But this is too simplistic for a real system as there are many types of utterances about actions that a user might provide that cannot be so expressed. For example, the user might want to personalize the system by telling it never to book a particular restaurant, i.e., the user wants the system *not* to perform an action. Moreover, a virtual assistant positioned in a living room may be expected to help multiple people, either as individuals or as a group. A system needs to keep separate the actions and parameters characterizing one person's desires from another's, or else it will be unable to follow a discussion between two parties about an action. For example, John says *he wants* the system to reserve Vittorio's for he and Sue on Monday, and Sue says *she wants* the reservation on Tuesday. In addition to specifying agents for actions, we need to specify the agent of the inform, so that we can separate what John and Sue each said, as in: inform(agent=john, reserve(patron=[john,sue],day=monday)), and inform(agent=sue,reserve(patron=[john,sue], day =tuesday)). But, since I+S slots encode the *speaker's* desire, how can *John's* saying "Sue wants you to reserve Monday" be represented? Does this utterance fill slots in Sue's desired reservation action, both of theirs, or neither? And what if Sue replies *"no, I don't"*? What then is in the day slot for Sue? Dontcare? She didn't say she doesn't care what day a table is reserved. In fact, she *does* care — she does *not want* a reservation on Monday. By merely having an implicit attitude, we cannot represent this.[6]

All these representational weaknesses compound. Imagine John's being asked by the system *"when do you want me to reserve Vittorio's?"* and he replies *"whenever Sue wants."* Again, whose slot and attitude is associated with the utterance— John's or Sue's? Without a shared variable, agents for actions, and explicit desires, we cannot represent this either.

2.4 Mixed initiative and collaboration

Finally, in the dialogue below, apart from the fact that I+S cannot represent utterance (1), question (2) is answered with a subdialogue starting at question (3) that shifts the dialogue initiative (Bohus and Rudnicky, 2002; Horvitz, 2007; Litman and Allen, 1987; Morbini et al., 2012). In utterances (4) and (6), the system is proposing a value and in (5) and (7), the user is rejecting or accepting the proposal. Thus, *both* system and user are *collaboratively* filling the slot (Clark and Wilkes-Gibbs, 1986), not just one or the other. I+S systems cannot do this.

(1) U: Please book a reservation at the closest good restaurant to the Orpheum Theater on Monday for 4 people.
(2) S: OK, I recommend Guillaume. What time would you like to eat?
(3) U: what's the earliest time available?
(4) S: 6 pm
(5) U: too early
(6) S: how about 7 pm?
(7) U: OK

2.5 Dialogue state and belief

The DSTC approach to I+S represents dialogue state in terms of the user's desires. We claim that task-oriented dialogue systems, especially those that could engage in multiparty conversations, will also need to explicitly represent other mental states, including but not limited to people's beliefs.[7] The naive approach to representing beliefs is as an embedded database (Cohen, 1978; Moore, 1977). Such an approach could perhaps work until one attempts to deal with vague beliefs. For example, you know Joe is sitting by a window and able to look outside. You can reasonably ask Joe "Is it raining?" because you believe that *either* Joe believes it is raining, *or* Joe believes it is not raining, i.e., Joe knows whether it is raining or not. This is different than believing that Joe believes that Rain ∨ ~Rain, which is a tautology. But to use the database approach, what should the system put into Joe's database? It can't put in Rain, and it can't put in ~Rain, or else it would not need to ask. It needs to represent something

[6]Some researchers have advocated a "*negate(a=x)*" action with an informal semantics that the user does not want the slot *a* to be filled with the value *x* (Young et al., 2010). In the multiparty case, one would need to be more explicit about whose slot and desire this is.

[7] This is a different notion of "belief" than "belief state" as used in POMDP dialogue modeling (Williams & Young, 2007).

more vague – *that* Joe knows if it is raining, a concept that was described as KNOWIF $=_{def}$ (BEL x P) ∨ (BEL x ~P) (Allen 1979; Cohen and Levesque, 1990b; Cohen and Perrault, 1979; Miller et al., 2017; Perrault and Allen, 1980; Sadek et al., 1997, Steedman and Petrick, 2015). In the case of a multiparty dialogue system, the system should direct the yes/no question of whether it is raining to the person whom it believes knows the answer without having to know what they think it is.

2.6 Knowledge acquisition

Any task-oriented dialogue system will need to acquire information, usually by asking wh-questions, which we have argued will require it to deal somehow with variables. Again, for a multiparty context, in order to ask a wh-question, the system should be asking someone whom it thinks knows the answer. We need to be able to represent such facts as "John knows Mary's mobile phone number", which is different from saying "John knows Mary has a mobile phone number". In the former case, I could ask John the question "what is Mary's phone number?", while in the latter case, it would be uncertain whether he could reply. This ability to represent an agent's knowing the referent of a description, was called KNOWREF (Allen 1979; Cohen and Levesque, 1990b; Cohen and Perrault, 1979; Perrault and Allen, 1980), Bref (Sadek et al., 1997), or KNOWS_VAL (Young et al., 2010), and is intimately related to the concept of quantifying-into a modal operator (Barcan, 1946; Kaplan, 1968; Kripke, 1967; Quine, 1956), about which a huge amount of philosophical ink has been spilled. For a database approach to representing belief, the problem here revolves around what to put in the database to represent Mary's phone number. One cannot put in a constant, or one is asserting that to be her phone number. And one cannot put in an ordinary variable, since that provides no more information than the existentially quantified proposition that she has a phone number, not that John knows what it is! Over the years, various researchers have attempted to incorporate special types of constants (Cohen, 1978; Konolige, 1987), but to no avail because the logic of these constants requires that they encode all the modal operators in whose scope they are quantified. Rather, one needs to represent and reason with quantified beliefs like

∃X (BEL john phone_number(mary,X))

To preview our logic below, we define some syntactic sugar using roles and Prolog syntax (and a higher-order schematic variable ranging over predicates Pred):

(KNOWREF agent:X variable:Var predicate:Pred) $=_{def}$ ∃ Var (BEL x Pred), with Var bound in Pred

In other words, the agent X knows the referent of the description 'Var such that Pred'. For example, we can represent "John knows Mary's phone number" as

(KNOWREF agent:john,variable:Ph,
predicate:phone_number(mary,Ph))

In summary, a system's beliefs about other agents cannot simply be a database. Rather, the system needs to able to represent such beliefs without having precise information about what those beliefs are.[8] If it can do so, it can separate what it takes to be one agent's beliefs from another's, which would be needed for a multiparty dialogue system. Dialogue state for task-oriented dialogue systems is thus considerably more complex than envisioned by I+S approaches.

3 Logic of Task-Oriented Conversation

Let us now cast the I+S dialogue setting into a logical framework. We will examine intent vs. intention, semantics of slots, and dialogue state.

3.1 What is an Intent?

How does the action description in such utterances as those above relate to an "intent"? First, let us assume "intent" bears some relation to "intention". What appears to be the use within the spoken language community is that an "intent" is the action content of a user request that (somehow) encodes the user's intention. To be precise here, we need to review some earlier work that can form the basis for a logic of task-oriented conversation.

3.2 The Language $\mathcal{I}$

We will use Cohen and Levesque's (1990) formal language and model theory for expressing the relations among belief, goal, and intention (see Appendix for precise description of $\mathcal{I}$). Other formal languages that handle belief and intention (e.g., (Rao and Georgeff, 1995)) may do just as

[8] Note that this has nothing to do with uncertainty in the probabilistic sense. I can be certain that John knows Mary's phone number, but still not know what it is.

well, but this will provide the expressivity we need. The language $\mathcal{L}$ is a first-order multi-modal logical language with basic predicates, arguments, constants, functions, objects, quantifiers, variables, roles, values (atomic or variables), actions, lists, temporal operators (Eventually (◊, LATER), DOES and DONE), and two mental states, BEL and GOAL. The logic does not consider agents' preferences, assuming the agent has chosen those it finds superior (according to some metric such as expected utility). These are called GOALs in the logic. Unlike preferences, at any given time, goals are consistent, but they can change in the next instant. As is common, we refer to this as a BDI logic. See the Appendix for examples of well-formed formulas.

3.3 Possible worlds semantics

Again from (Cohen and Levesque, 1990), the propositional attitudes BEL and GOAL are given a relatively standard possible worlds semantics, with two accessibility relations *B* and *G*. However, for modelling slot-filling, we are critically interested in the semantics of "quantifying-in" (Barcan, 1946; Kaplan, 1968; Kripke, 1967; Quine, 1956). Briefly, a variable valuation function *v* in the semantics assigns some value chosen from the domain of the world and time at which the formula is being satisfied. When "quantifying-into" a BEL or GOAL formula, that value is chosen and then the BEL or GOAL formula is satisfied. As is standard in modal logic after (Kripke, 1967), the semantics of these modal operators is given in terms of a universal quantifier ranging over *B*- and *G*-related possible worlds. Thus, the semantics of satisfying ∃y(BEL x p(y)) in world *W* is that there is a single value that is assigned by the variable assignment function *v* to y, such that *for all* worlds *W'* that are *B*-related to *W*, p(y) is true in *W'*. In other words, the value assigned to y is the same for all the related worlds *W'*. If the quantifier is within the scope of the modal operator as in (BEL x ∃y p(y)), then a different value could be assigned to the variable in each *B*-related world. Likewise, one can quantify into GOAL, and even iterated modalities or modalities of different agents. This gives rise to the theorems below, and analogous ones for GOAL.

$$\models \exists y\,(\mathrm{BEL}\ x\ p(y)) \supset \subset (\mathrm{BEL}\ x\ \exists y\ p(y)),\ \text{and}$$

$$\models \mathrm{BEL}\ x\ p(c) \supset \subset \exists y\,(\mathrm{BEL}\ x\ p(y))\ \text{for constant c.}$$

This paper shows why quantifying into BEL and GOAL is key for slot-filling systems.

3.4 Persistent goals and intentions

Cohen and Levesque (1990) defined a concept of an internal commitment, namely an agent's adopting a relativized persistent goal (PGOAL x P Q), to be an achievement goal P that x believes to false but desires to be true in the future, and agent x will not give up P as an achievement goal at least until it believes P to be satisfied, impossible, or irrelevant (i.e., x believes ~Q). If the agent believes ~Q, it can drop the PGOAL. More formally, they have:

$$(\mathrm{PGOAL}\ x\ P\ Q) =_{\mathrm{def}} (\mathrm{GOAL}\ x\ (\mathrm{LATER}\ P)) \wedge (\mathrm{BEL}\ x\ \sim P) \wedge (\mathrm{BEFORE}\ (\ (\mathrm{BEL}\ x\ P) \vee (\mathrm{BEL}\ x\ \Box\sim P) \vee (\mathrm{BEL}\ x\ \sim Q))\ \sim(\mathrm{GOAL}\ x\ (\mathrm{LATER}\ P))$$

They also defined an *intention* to be a persistent goal to perform an action. More formally:

$$(\mathrm{INTEND}\ x\ A\ Q) =_{\mathrm{def}} (\mathrm{PGOAL}\ x\ (\mathrm{DONE}\ x\ A)\ \ Q).$$

In other words, an agent x intending to do an action A is internally committed (i.e., has a PGOAL) to having performed the action A in the future. So, an intention is a future-directed commitment towards an action.

3.5 What is a slot?

Given this language, how would one represent a DSTC slot, which incorporates the user's desire? We propose to separate the attitude, action, and role-value list, then reassemble them. First, we consider the role:value argument in an action expression, using upper case variables (as in Prolog), such as reserve(patron:P, restaurant:R, day:D, time:T, num_eaters:N). Here, restaurant:R is the role:value expression. Next, we need to add the desire attitude (as a PGOAL) in order to express such phrases "the day Joe *wants* me to reserve Vittorio's Ristorante for him." Here is how we would express it as part of the system's belief:

(1) ∃Day
  (PGOAL joe ∃[T ,N]
    (DONE sys reserve([patron:joe,
      restaurant:vittorios,
      day:Day, time:T,
      num_eaters:N])) Q)

In other words, there is a Day on which Joe is committed to there being a Time, and number of eaters N such that the system reserves Vittorio's

on that Day at that Time and with N eaters. The system has represented Joe as being picky about what day he wants the system to reserve Vittorio's (e.g., as a creature of habit, he always wants to eat there on Monday), but the system does not know what day that is. Here, we have quantified Day into the PGOAL, but the rest of the variables are existentially quantified within the PGOAL. That means that Joe has made no choice about the Time or Number of people. But because the system has this representation, it can reasonably ask Joe "What day do you want me to reserve Vittorio's?". We can now also represent the day Joe does *not* want the system to reserve, can distinguish between the day Joe wants the system to reserve and the day Sue wants, and we can even equate the two, saying that Joe wants the system to reserve on whatever day Sue wants (See section 2.7). So the DSTC "slot" day turns out to have a variable in an action expression all right, but one that is now quantified into an intention or PGOAL operator. This explicit representation enables the system to discuss the action with or without anyone's wanting to perform it, and to differentiate between agents' attitudes, which is essential for multiparty dialogues.

3.6 Where do the slot-filling goals and intentions come from?

In order to know what action to perform, an agent needs to know the values of the required arguments of an action. (Allen and Perrault, 1980; Appelt, 1985; Cohen and Perrault, 1979; Moore, 1977)[9]. In the case of the task-oriented dialogue setting, in which the agents are intended to be cooperative, we will have all agents obey the following rule. (We suppress roles below and hereafter.)

For any agents X and Y (who could be the same):

If: (BEL Y (PGOAL X (DONE Y A) Q)),

Then for the set of required but unfilled obligatory arguments Args, assert

(2) (PGOAL Y
(KNOWREF Y Args (PGOAL X (DONE Y A)),
(PGOAL X (DONE Y A) Q)),

In other words, assuming Y is the system and X is the user, this rule says that if the system believes the user is committed to the system's doing an action A (as would be the result of a request), then the system is committed to knowing the referents of all required arguments of the action A that the user wants the system to perform.[10] That is, the system is committed to knowing the *user's* desired "slot" values in the action that the user wants the system to perform. For example, if the system believes the user wants the system to do the action of reserving Vittorio's Ristorante for the user, then the system adopts a persistent goal to know the Time, Day, and Num, for which the user wants the system to reserve Vittorio's.[11]

Notice that this holds no matter how the system comes to infer that the user wants it to do an action. For example, the system could make an indirect offer and the user could accept (Smith and Cohen, 1996), as in *System*: *"Would you like me to reserve vittorio's for you?" User: "Sure".* Here, the offer is stated as a question about what the user wants the system to do, and the positive reply provides the system with the rule antecedent above.

3.7 Application of the logic to I+S: Expressing problematic user responses

Let us now apply the logic to handle some of the expressions we claimed were problematic for an I+S approach. Assume the system has asked the user: *"What time do you want me to reserve Vittorio's Ristorante?"* We start with the base case, i.e. with the user's supplying an atomic value, and assume the representation of the question has only the Time variable quantified-in.

User: "7 pm".
Essentially, we unify the variable quantified into the PGOAL with the atom 7pm, resulting in:

(PGOAL usr ∃[Day,N]
(DONE sys reserve([usr, vittorios,Day,7pm, N]))
Q)
This is classic slot-filling.

User: "I don't know". The system would need to assert into its database a formula like the following (assume the action variable A

[9] Required arguments will be stipulated as part of a meta-data template in the system's knowledge base. Knowing the values for arguments of actions is not the only case in which having to know an argument is required. For example, for the system to determine the number of available seats at a restaurant, it needs to know the date.

[10] When X and Y are the same agent, (PGOAL X (DONE X A)) is exactly the definition of an intention.

[11] Formula (1) is a consequence of this.

represents the act of reserving Vittorio's for the user, and that it has a free variable Time):

~ (KNOWREF usr Time
(PGOAL usr (DONE usr, A) Q))

In doing so, the system should retract its previous KNOWREF belief that enabled it to ask the original question. How a system responds to this statement of ignorance is a different matter. For example, it might then ask someone else if it came to believe that person knows the answer. Thus, if the user then said "but Mom knows" and the system believes the user, the system could then ask Mom the question.

User: "I don't care". There are only two approaches we have seen to handling this in the I+S literature. One is to put the Dontcare atom into the value of a slot (Henderson, 2015). However, it is not clear what this means. It does not mean the same thing as "I don't know." It might be the equivalent of a variable, as it matches anything as a slot value, but that begs the question of variables in slots. To express "I don't care" in the logic, we can define CAREREF, a similar concept to KNOWREF:

(CAREREF x Var Pred) $=_{def}$ ∃Var (GOAL x Pred),

where Var is free in Pred. Then for "I don't care", one could say: ~(CAREREF x Var Pred) with the formal semantics that there is no specific value v for Var towards which x has a goal that Pred be true of it.

Rather than have a distinguished "don't care" *value* in a slot, Bapna et al. (2017) create a "don't_care(slot)" *intent*, with the informal meaning that the user does not care about what value fills that slot.[12] Here, it is not clear if this applies on a slot-by-slot basis, or on an intent+slot basis. For example, if it is on a slot-by-slot basis, then if the user says "I don't care" to the question "Do you want me to reserve Monday at 7pm or Tuesday at 6pm?" it would lead to four don't_care(slot) intent expressions. Would these be disjunctions? How would the relation between Monday and 7pm be expressed?

By contrast, we can define a comparable concept to KNOWIF,

(CAREIF x P) $=_{def}$ (GOAL x P) ∨ (GOAL x ~P)

such that one can say "x doesn't care whether P", as ~(CAREIF x P), with the obvious logical interpretation. With CAREIF, one could express the reply "I don't care" to the above disjunctive question as:

~(CAREIF usr
(LATER
(DONE sys reserve([usr, mond, 7pm)]) ∨
(DONE sys reserve([usr, tues , 6pm]))))

User: "before 8 pm." Because all that the I+S approach can do is to put atomic values in slots or leave them unfilled, the only approach possible here is to put some atom like before_8_pm into the slot. If one tried to give a semantics for this, it might be a function call or λ-expression that would somehow be interpreted as a comparative relation with whatever value eventually fills the slot. But, one would need a different comparison relation for every time value, not to mention for other more complex expressions such as not_before_7_pm_or_after_9_pm, or between_7_pm_and_9_pm. How would the system infer that these are the same condition? Instead, one might think we only need a method to append new constraints to the quantified persistent goal "slot" expression, as in

∃ Time (PGOAL usr
∃ [Day,Num]
(DONE sys
reserve([usr,vittorios,Day,Time,Num]))
∧ (BEFORE Time 8:15_pm))

However, as a representation of the reply, the above is not quite what we want. Here, the user has implicated (Grice, 1975) that she does not have a goal for a particular time such that she wants a reservation at that time. Rather, she wants *whatever* time she eats to be before 8:15 pm. So, in fact, we want this constraint to be embedded within the scope of the existential quantifier:

(PGOAL usr ∃ [Day,Time,Num]
((DONE sys reserve([usr,vittorios,
Day,Time, Num]))
∧ (BEFORE Time 8:15_pm)))

The reason we need an inference like a Gricean implicature is that the system would need to reason that in response to the question, if the user knew the answer, she would have told me, and she didn't, so she (probably) doesn't know the answer. Thus, the system needs to assert a weaker PGOAL.

[12] Notice that "intent" for Bapna et al. does not indicate an action being requested, so their notion of intent is different from that of (Henderson, 2015) or that used by Amazon Alexa.

User: "whenever Mary wants." To represent the content of this utterance, one can equate the quantified-in variables T_1, T_2 (and ignoring Q):

```
∃[T1,T2] (equals T1,T2) ∧
 ((PGOAL usr ∃[Day,Num]
    (DONE sys reserve([usr,vittorios,Day, T1, Num]))) ∧
  (PGOAL  mary ∃[Day,Num]
    (DONE sys reserve([mary,vittorios,Day, T2,Num]))))
```

If the system learns that Mary wants the reservation to be at 7 pm, it can infer that the User wants it then too.

The above examples show that the logic can represent users' utterances in response to slot-filling questions that supply constraints on slot values, but not the values themselves.

4 Towards Best Practices

This paper has provided a logical definition of the DSTC 2/3 slot (and I+S slots more generally) as a quantified-in formula stating the value that the agent wants an action's role to have. In addition, the logic presented here captures a more general concept than what I+S supports, in that it can express multiple agents' desires as well as non-atomic constraints on attribute-value in logical forms.

Still, our purpose here is not merely clarity and good hygiene, but ultimately to build systems that can engage in explainable, collaborative, multiparty dialogues. Below we sketch how to build systems that can handle the above issues, some of which we have implemented in a prototype system that uses the logic in this paper to engage in collaborative knowledge-based dialogues, including slot-filling. A report on this system and approach will be provided in a subsequent paper.

4.1 Enabling an operational semantics

Systems based on a BDI logic will often have a belief-desire-intention architecture that serves as an operational semantics for the logic (Rao and Georgeff, 1995). By "operational semantics", we mean that the system's operation behaves (or at least approximates) the requirements of the logic. For example, the adoption of a persistent goal to achieve a state of affairs results in finding a plan to achieve it, which then results in the agent's intending to perform the planned action. If the system finds a persistent goal/intention to be achieved, impossible or irrelevant, it drops that mental state, which causes an unraveling of other mental states as well. Our system in fact reasons with the formulas shown here, engaging in slot-filling and related question-answering dialogues. However, other systems may be able to make such distinctions without explicit logical reasoning.

4.2 A plan-based approach to dialogue

We advocate a plan-based model of dialogue (Allen, 1979, Allen and Perrault, 1980; Allen et al., 1995; Appelt, 1985; Cohen 1978; Cohen and Perrault, 1979; Cohen and Levesque, 1990b; Galescu et al., 2017; Litman and Allen 1987; Perrault and Allen, 1980; Sadek et al., 1997; Steedman and Petrick, 2007; Stone, 2004; Traum and Hinkelman, 1992) such that the same planning and plan recognition algorithms can apply to both physical, digital, and communicative acts. When applied to communicative acts, the system plans to alter its own and the users' beliefs, goals, and intentions. For example, goal (2) as applied to the slot expression in (1) will cause it to plan the wh-question *"what day would you like me to reserve Vittorio's?"* to alter the speaker's KNOWREF in goal (2) (see Appendix for definition of whq). Conversely, as a collaborator, on identifying a user's speech act, the system asserts the user's goal was to achieve the effect of the speech act. Based on that effect, the system attempts to recognize the user's larger plan, to debug that plan, and to plan to overcome obstacles to it so that the user may achieve his/her higher level goals (Allen, 1979; Cohen, 1978; Cohen et al., 1982). In this way, a system can engage in collaborative non-I+S dialogues such as User: "Where is Dunkirk playing?" System: "It's playing at the Roxy theater *at 7:30pm, however it is sold out. But you can watch it on Netflix."* Finally, the system is in principle explainable because everything it says has a plan behind it.

4.3 A hybrid approach to handling task-oriented dialogue variability.

In order to incorporate such an approach into a useful dialogue system, we advocate building a semantic parser using the crowd-sourced "overnight" approach (Duong et al., 2018; Wang et al., 2015), which maps crowd-paraphrased utterances onto LFs derived from a backend API or data/knowledge base. This methodology involves: 1) Creating a grammar of LFs whose predicates are chosen from the backend application/data base, 2) using that grammar to generate a large number of LFs, 3) generating a "clunky" paraphrase of an LF, and 4) collecting

enough crowd-sourced natural paraphrases of those clunky paraphrases/LFs[13]. A neural network semantic parser trained over such a corpus can handle considerable utterance variability, including the creation of logical forms both for I+S utterances, and for complex utterances not supportable by I+S approaches. In the past, we have used this method to generate a corpus of utterances and logical forms that supported the semantic parsing/understanding of the complex utterances in Section 2.2 (Duong et al., 2017; Duong et al., 2018).

Whereas much utterance variability and uncertainty can be captured via the above approach, we believe there is less variability at the level of the goal/intention lifecycle, which includes goal adoption, commitment, planning, achievement, failure, abandonment, reformulation, etc. (Galescu et al., 2018; Johnson et al., 2018). This goal lifecycle would be directly supported by the BDI architecture and therefore would be available for every domain. Rather than train a dialogue system end-to-end where we would need many examples of each of these goal relationships, we believe a domain independent dialogue manager can be written once, parameterized by the contents of the knowledge representation (Allen et al., 2019; Galescu et al., 2018). Beyond learning to map utterances to logical forms, the system needs to learn how to map utterances in context to goal relationships. For example, what does "too early" in Utterance (5) of Section 2.4 mean? Is that a rejection of a contextually-specified proposal? The system also needs to learn how actions in the domain may lead to goals for which the user may want the system's assistance. In order to be helpful to the user, the system must recognize the user's goals and plan that led to his/her utterance(s) (Allen and Perrault, 1980; Sukthankar et al., 2014; Vered et al., 2016). One approach is to collect the action data needed to support plan recognition via crowdsourcing and text mining (Branavan et al., 2012; Fast et al., 2016; Jiang and Riloff, 2018). The upshot will be a collaborative dialogue manager that can be used directly in a dialogue system, or can become a next generation user simulator with which to train a dialogue manager (Schatzman et al., 2007; Shah et al., 2018).

[13] This might take longer than overnight (vs. Wang et al. 2015).

Acknowledgments

This paper benefitted from insightful comments by the reviewers, Drs. Mark Johnson, Lizhen Qu, and Mor Vered.

5 References

Allen, J. F. A plan-based approach to speech act recognition, PhD Thesis, Dept. of Computer Science, University of Toronto, 1979.

Allen, J. F. and Perrault, C. R., Analyzing intention in utterances, *Artificial intelligence 15 (3)*, 143-178.

Allen, J. F., Schubert, L K., Ferguson, G. Heeman, P. Hwang, C. H., Kato, T., Light, M. Martin, N., Miller, B., Poesio, M., Traum, D. R., The TRAINS project: A case study in building a conversational planning agent *Journal of Experimental and Theoretical Artificial Intelligence*, 1995

Allen, J. F., André, E., Cohen, P. R., Hakkani-Tür, D., Kaplan, R., Lemon, O., Traum, D., Challenge discussion: Advancing multimodal dialogue, Chapter 5 in *Handbook of Multimodal-Multisensor Interfaces*, Oviatt, S. L., Schuller, B., Cohen, P. R., Sonntag, D., Potamianos, G., and Krüger, A., ACM Press/Morgan and Claypool Publishers, 2019.

Appelt, D. *Planning English Sentences*, Cambridge University Press, Cambridge, UK, 1985

Barcan, R. C., A Functional Calculus of First Order Based on Strict Implication, *Journal of Symbolic Logic, 11*, 1946.

Bapna, A., Tür, G., Hakkani-Tür, D., and Heck, L., Sequential dialogue context modelling for spoken language understanding, *Proc. of SIGDIAL*, 2017, 103-114.

Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H., and Winograd, T. GUS, a frame-driven dialog system. *Artificial Intelligence, 8(2)*, 1977, 155-173.

Bohus, D. and Rudnicky, A. I., The RavenClaw dialogue management framework, *Computer Speech and Language, 23*, 2009, 332-361.

Branavan, R.K., Kushman, N., Lei, T., Barzilay, R. Learning High-Level Planning from Text, Proc. ACL-12, 2012, 126-135.

Clark, H. H., and Wilkes-Gibbs, D., Referring as a collaborative process, *Cognition(22)*, 1986, l-39

Cohen, P. R. On knowing what to say: Planning speech acts. PhD Thesis, Dept. of Computer Science, University of Toronto, 1978.

Cohen, P. R. and Levesque, H. J., Intention is choice with commitment, *Artificial Intelligence, 42 (2-3),* 1990, 213-261.

Cohen, P. R, and Levesque, H. J. , Rational Interaction as the Basis for Communication *Intentions in Communication*, Cohen, P. R., Morgan, J. and Pollack, M.E., MIT Press, 1990a.

Cohen, P. R. and Perrault, C. R., Elements of a plan-based theory of speech acts, *Cognitive Science, 3(3)*, 1979.

Cohen, P. R., Perrault, C. R., and Allen, J. F., Beyond question-answering, in Strategies for Natural Language Processing, Lehnert, W. and Ringle, M. (eds)., Lawrence Erlbaum Associates, 1982.

Duong, L., Afshar, H., Estival, D., Pink, G., Cohen, P. R., and Johnson M. Multilingual Semantic Parsing and Code-switching, *Proc. of the 21st Conf. on Computational Natural Language Learning (CoNLL 2017)*, 2017, pp. 379-389.

Duong, L., Afshar, H. Estival, D., Pink, G., Cohen, P., Johnson M.,Active learning for deep semantic parsing. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018, 43-48.

Fast, E., McGrath, W., Rajpurkar, P. and Bernstein, M., Augur: Mining Human Behaviors from Fiction to Power Interactive Systems. *Proc. of the 2016 CHI Conference on* Human Factors in Computing Systems, ACM Press, 2016.

Galescu, L., Teng, C. M., Allen J. F., and Pereira, I. Cogent: A Generic Dialogue System Shell Based on a Collaborative Problem Solving Model, *Proceedings of SigDial,* 2018, 400-409.

Gašić, M., Mrkšić, N., Rojas-Barahona, L. M., Su, P-H., Ultes,S., Vandyke, D., Wen, T-H., and Young, S., Dialogue manager domain adaptation using Gaussian process reinforcement learning, Computer Speech and Language 45, 2016, 552-569.

Grice, H.P. Logic and Conversation, *Syntax and Semantics,* vol.3 P. Cole and J. Morgan (eds.), Academic Press, 1975.

Henderson, M., Machine learning for dialog state tracking: A review, *Proceedings of The First International Workshop on Machine Learning in Spoken Language Processing*, 2015.

Horvitz, E., Reflections on challenges and promises of mixed-initiative interaction, *AI Magazine, 28(2)*, 2007, 19-22.

Johnson B., Floyd M.W., Coman A., Wilson M.A., Aha D.W. Goal reasoning and trusted autonomy. In: Abbass H., Scholz J., Reid D. (eds), *Foundations of Trusted Autonomy. Studies in Systems, Decision and Control, vol 117*. Springer, 47-66, 2018.

Kaplan, D. Quantifying in, *Synthese 19(1/2)*, 1968, 178-214.

Konolige, K., On the relation between autoepistemic logic and circumscription: Preliminary Report, *Proc. of IJCAI*, 1989, 1213-1218.

Kripke, S. A. Semantical Considerations on Modal Logic *Acta Philosophica Fennica 16* 1963, 83-94.

Jiang, T., and Riloff, E., Learning prototypical goal activities for locations, *Proc. of Assoc. for Comp. Ling.*, 2018, 1297-1307.

Larsson, S. and Traum, D. R., Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit, *Natural Language Engineering* 6(3-4), 2000, 323-340.

Litman, D. J. and Allen, J. F., A Plan Recognition Model for Subdialogues in Conversations, *Cognitive Science, 11*, 1987, 163-200.

Miller, T., Felli, P., Muise, C., Pearce, A. R., and Sonenberg, L. 'Knowing whether' in Proper Epistemic Knowledge Bases, *Proc. of AAAI*, 2017.

Morbini, F., DeVault, D., Sagae, K., Gerten, J., Nazarian, A., and Traum D., FLoReS: A Forward Looking, Reward Seeking, Dialogue Manager, *Proceedings of the 4th International Workshop on Spoken Dialog Systems* November, 2012, 151-162.

Moore, Robert C, Reasoning about knowledge and action, *Proc. of IJCAI,* 1977.

Perrault, C. R. and Allen, J. F., A plan-based analysis of indirect speech acts, *Computational Linguistics, 6(3-4)*, 1980, 167-182.

Rao, A. and Georgeff, M. BDI-agents: From Theory to Practice". *Proceedings of the First International Conference on Multiagent Systems*, 1995.

Rao, A. and Georgeff, M. Decision procedures for BDI logics, *Journal of Logic and Computation 8(3),* 1998.

Quine, W. V. O. Quantifiers and propositional attitudes, *Journal of Philosophy 53(5)*, 1956, 177-187.

Sadek, D., Bretier, P., and Panaget, F., ARTIMIS: Natural dialogue meets rational agency, *Proc. IJCAI-15*, 1997, pp. 1030-1035.

Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., and Young, S., Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System*, Proc. of NAACL-HLT*, 2007.

Shah, P., Hakkani-Tür, D., Tür, G., Rastogi, A., Bapna, A., Nayak, N., Heck, L., Building a conversational agent overnight with dialogue self-play, *arXiv: 1801.04871v1*, Jan., 2018.

Smith, I. A., and Cohen, P. R. Toward a semantics for an agent communications language *Proc. AAAI-9*6, 24-31.

Steedman, M. and Petrick, R. Planning dialogue actions, *Proc. of SigDial*, 2007.

Stone, M. Intention, interpretation and the computational structure of language, *Cognitive Science 28*, 2004, 781–809.

Sukthankar, G., Geib, C., Bui, H., Pynadath, D., and Goldman, R., *Plan, Activity, and Intent Recognition: Theory and Practice*, San Francisco: Morgan Kauffman Publishers, 2014.

Traum, D. R. and Hinkelman, E. A., Conversation acts in task-oriented spoken dialogue, *Computational Intelligence, 8(3),* 575-599.

Ultes, S. Budzianowski, P., Casanueva, I., Rojas-Barahona, L., Tseng B-H., Wu, Y-C., Young, S., and Gašić, M. Addressing Objects and Their Relations: The Conversational Entity Dialogue Model, Proc. of SigDial 2018.

Vered, M., Kaminka, G. A. and Biham S. Online Goal Recognition through Mirroring: Humans and Agents. In *Proceedings of the Annual Conference on Advances in Cognitive Systems (ACS)*, 2016.

Wang, Y., Berant, J., and Liang, P., Building a semantic parser overnight, Proc. of Assoc. for Comp. Ling., 2015, 1332–1342.

Williams, J. D., and Young, S. Partially observable Markov decision processes for spoken dialog systems, *Computer Speech and Language 21* (2007), 393-422.

Woods, W. A. What's in a Link: Foundations for Semantic Networks. In D. Bobrow and A. Collins (eds.), *Representation and Understanding: Studies in Cognitive Science*, New York: Academic Press, 1975.

Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management, *Computer Speech and Language 24,* 2010, pp. 150-174.

Zue, V. W., Glass, J., Goodine, D., Hirschman, L. Leung, H. C., Phillips, M., Polifroni, J., Seneff, S. "The MIT ATIS system: Preliminary development, spontaneous speech data collection, and performance evaluation", *Proc of EUROSPEECH*, 1991, 537-540.

Appendix

The Language $\mathcal{L}$

Variables and constants

<Action-var> ::= a, b, a_1 , a_2 . . . b_1, b_2 . . . e, e_1, e_2 . . .
<Agent-var> :: = x, y, x_1, x_2 . . . . . y_1, y_2 . .
<Regular-var> ::= i,j, i_1, i_2 . . . j_1, j_2 . . .
<Variable>:: = <Agent-var>|<Action-var> | < Regular-var> |
[<$Variable_1$> …<$Variable_n$>] , i.e., (a list of variables)

Predicates and Formulas

<Role>*::*= distinguished Role symbols for a given action
<Role-list> ::= [<$Role_1$:<$Variable_1$>,. . ., <$Role_n$:<$Variable_n$>]
<Pred-symbol> ::= an element of a distinguished set of predicate symbols
<Pred> :: = (<Pred-symbol>) .

Well-formed formulas (WFFS)
<Wff> ::= <Pred> | ~<Wff> | <Wff> ∨ <Wff> | <Wff> ∧ <Wff> | ∃<Variable> <Wff> |
◊<Wff> — <Wff> is true eventually
□<Wff> — <Wff> is always true (note that □<Pred>=$_{def}$ ~ ◊~ <Pred>)
<Variable> = <Variable>
(DOES <Action-expr>) — <Action-expr> happens next,
(DONE <Action-expr>) —
<Action-expr> has *just* happened,
(Agt <Agent-var> <Action-var>):
<Agent-var> is the only agent of <Action-var>,
(BEL <Agent-var> <Wff>) —
meaning <Wff> follows from <Agent-var>'s beliefs,
(GOAL <Agent-var> <Wff>) —
meaning <Wff> follows from <Agent-var>'s goals,
<Time-proposition> ::= <Numeral>
(LATER <Wff>) ::= ~<Wff> ∧ ◊<Wff>
<Wff> is false now but eventually true
(BEFORE <Wff_1> <Wff_2>)
before <Wff_1> becomes true,

Action expressions:

<Action-name> ::= an element of a designated set of action names
<Action-expr> :: =
<Action-var> or one of the following:
<Action> ::= <Action-name (Role-list>)
<Action-expr>;<Action_expr>— sequential action,
<Action-expr> | <Action-expr> — nondeterministicchoice action,
<Wff>? — test action
<Action-expr>||<Action-expr> — concurrent action
<Action-expression>*: iterative action.

Examples of Well-formed Formulas:

(DONE joe eat(joe,vittorios,mond,7pm))
Joe has just eaten at Vittorio's on Monday at 7pm.
(GOAL joe ◊(DONE joe eat(joe,vittorios, mond,7pm))
Joe's goal is to eventually have eaten at Vittorio's at Monday at 7pm.
(BEL john (PGOAL mary (KNOWREF john variable:Time
(PGOAL mary
eat(mary,vittorios,mond,Time))
John believes Mary has a persistent goal for him to know the time that Mary wants to eat at Vittorio's on Monday.

Speech Act definitions

whq([Speaker, Listener, Var, Pred])
Precondition: (KNOWREF Listener, Var, Pred)
Effect: (KNOWREF Speaker, Var, Pred)
Constraint: Speaker ≠ Listener

informref([Speaker, Listener, Var, Pred])
Precondition: (KNOWREF Speaker, Var, Pred)
Effect: (KNOWREF Listener, Var, Pred)
Constraint: Speaker ≠ Listener

ynq([**Speaker, Listener, Pred**]),
Precondition: (KNOWIF Listener Pred)
Effect: (KNOWIF Speaker Pred)
Constraint: Speaker ≠ Listener

inform(**[Speaker, Listener, Pred]**)
Precondition: (BEL Speaker Pred)
Effect: (BEL Listener Pred)
Constraint: Speaker ≠ Listener