Dialogue management with linear logic: the role of metavariables in questions and clarifications

Vladislav Maraev[*] — **Jean-Philippe Bernardy**[*] — **Jonathan Ginzburg**[**]

[*] *Centre for Linguistic Theory and Studies in Probability (CLASP), Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg*
[**] *Laboratoire de Linguistique Formelle (LLF), CNRS – UMR 7110, Université de Paris*

ABSTRACT. *In this paper, we study the formalisation of a dialogue management system using proof-search on top of a linear logic. We argue that linear logic is the natural formalism to implement information-state dialogue management. We give particular attention to modelling question-answering sequences, including clarification requests, and argue that metavariables, arising from unification in the proof search, play a decisive role in providing a natural formalisation. We show that our framework is not only well suited from a theoretical perspective, but it is also suitable for implementation which we exemplify with a small scale implementation.*

RÉSUMÉ. *Cet article propose une formalisation de la gestion de dialogue via la recherche de preuves de formules de la logique linéaire. C'est-à-dire que nous proposons que la logique linéaire constitue une base naturelle de la formalisation de systèmes de gestion de dialogue basé sur un état d'information. Nous prêtons une attention particulière à la modelisation des séquences de questions-reponses (y compris les demandes de clarification), et nous arguons que les métavariables, résultant des unifications issue de la recherche de preuves, jouent un rôle décisif dans la formalisation. Nous montrons que notre système est non seulement adéquat d'un point de vue théorique, mais également d'un point de vue pratique. Ainsi, nous complétons notre argument d'une implementation d'un système de recherche de preuve générique, ainsi que d'un exemple de gestion de dialogue l'utilisant.*

KEYWORDS: *Symbolic dialogue management, Linear logic, Metavariables, Question answering, Clarification requests*

MOTS-CLÉS : *Gestion de dialogue symbolique, Logique linéaire, Métavariables, Question/Réponse, Demande de clarification*

TAL. Volume 61 – n°3/2020, pages 43 à 67

1. Introduction

A key aspect of dialogue systems design is the coherence of the system's responses. In this respect, a key component of a dialogue system is the dialogue manager, which selects appropriate system actions depending on the current state and the external context.

Two families of approaches to dialogue management can be considered: hand-crafted dialogue strategies (Larsson, 2002; Jokinen, 2009) and statistical modelling of dialogue (Rieser and Lemon, 2011; Young *et al.*, 2010; Huang *et al.*, 2020). Frameworks for hand-crafted strategies range from finite-state machines and form-filling to more complex dialogue planning and logical inference systems, such as Information State Update (ISU) (Larsson, 2002) that we employ here. Statistical models help to contend with the uncertainty that arises in human interaction; from noisy signals from speech recognition and other sensors to pragmatic ambiguities.

End-to-end systems that do not specify a dialogue manager as an explicit component have gained lots of attention recently (Huang *et al.*, 2020). Although most of them are focused on chit-chat dialogues, coherence plays a crucial role there too. Typically the main issues associated with such systems are related to memory limitations which cause repetition, contradiction and forgetfulness. Having a policy for dialogue coherence would be beneficial for such systems.

Although there has been a lot of development in dialogue systems in recent years, only a few approaches reflect advancements in *dialogue theory*. Our aim is to closely integrate dialogue systems with work in theoretical semantics and pragmatics of dialogue. This field has provided accounts for linguistic phenomena intrinsic to dialogue such as non-sentential utterances (Schlangen, 2003; Fernández *et al.*, 2007; Ginzburg, 2012), clarification requests (Purver, 2006; Ginzburg, 2012) and self-repair (Ginzburg *et al.*, 2014; Hough and Purver, 2012), where the resolution is intuitively tied to the coherence of what is being said.

To this end, a formal and in particular a logical representation is instrumental. This paper is concerned with the representation of participant states and transitions in a unified logical framework.

Even though the progress in bridging dialogue management and theoretical research of dialogue is promising, we believe that it is crucial to use formal tools which are most appropriate for the task, so that the formalisation and implementation of dialogue semantics closely matches the mental picture that experts have. In the view of Dixon *et al.* (2009) this is best done by representing the information-state of the agents as updatable sets of propositions. Subsets of propositions in the information state can be treated independently, and, therefore, a suitable and flexible way to represent updates is as propositions in linear logic. We adopt this view here, and further argue for it in the body of the paper.

We further extend the framework of Dixon *et al.* (2009) to deal with unclarity (and certain cases of non-probabilistic ambiguity). Indeed, asking a question is typi-

cally not done in one utterance which leaves nothing to interpretation. Typically, in a conversation a question and its answer may be many utterances apart, and the intermediate utterances form insertion sequences (Schegloff, 1972), for instance a follow-up clarification request and a corresponding answer. The insertion sequences are, in turn, conditioned on the preliminaries for the original question (Levinson, 1983, Chapter 6). In this paper, we deal with question-answering and clarification requests in a unified way, within a framework of dialogue management and using linear logic formalisation.

To deal with unclarity of an initial question, we propose here the use of *metavariables*, thereby leveraging much research on unification and proof search in various logical frameworks. That is, metavariables will stand in for any piece of information which is left to further interpretation. In particular, in this paper we explore the potential of using metavariables in the representation of question/answer exchanges.

By using a solid logical basis (Bratko, 2001; Girard, 1995) which corresponds well with the intuition of information-state based dialogue management, we are able to provide a fully working prototype [1] of the components of our framework:

1) a new proof-search engine based on linear logic, modified to support inputs from external systems (representing inputs and outputs of the agent);

2) a set of rules which function as a core framework for dialogue management (in the style of KoS (Ginzburg, 2012) theoretical account). The rules which we present below are provided to this engine, in the same form (modulo typesetting);

3) several examples which use the above to construct potential applications of the system. The engine is able to run domain-specific rules and generic rules together, forming a working system.

The rest of the paper is structured as follows. In section 2 we review important background for formalisation and implementation theories: dialogue management, linear logic and proof search. In section 3 we sketch a treatment of question, answers and clarification using the aforementioned formalisms. This treatment ignores certain dialogue management complexities, which we address in section 4. We discuss related work in section 5. Concluding remarks are provided in section 6.

2. Background

2.1. *Dialogue management*

2.1.1. *KoS*

KoS (not an acronym but loosely corresponds to Conversation Oriented Semantics) (Ginzburg, 2012) provides one of the most detailed theoretical treatments of domain-general conversational relevance, especially for query responses—see the work of

1. Source code and documentation are available at https://github.com/GU-CLASP/ProLin.

Purver (2006) on Clarification Requests, and Łupkowski and Ginzburg (2017) for a general account—and this ties into the KoS treatment of non-sentential utterances, again a domain crucial for naturalistic dialogue systems and where KoS has one of the most detailed analyses (Fernández *et al.*, 2007; Ginzburg, 2012).

In KoS (and other dynamic approaches to meaning), language is compared to a game, containing players (interlocutors), goals and rules. KoS represents language interaction by a dynamically changing context. The meaning of an utterance is then how it changes the context. Compared to most approaches, which represent a single context for both dialogue participants, KoS keeps separate representations for each participant, using the *Dialogue Game Board* (DGB). Thus, the information states of the participants comprise a private part and the dialogue gameboard that represents information arising from publicised interactions. The DGB tracks, at the very least, shared assumptions/visual field, moves (= utterances, form and content), and questions under discussion.

KoS is based on Cooper's formalism, Type Theory with Records (TTR) thus can leverage a wide range of work based on it, including the modelling of intentionality and mental attitudes (Cooper, 2005), generalised quantifiers (Cooper, 2013), co-predication and dot types in lexical innovation, frame semantics for temporal reasoning, reasoning in hypothetical contexts (Cooper, 2011), spatial reasoning (Dobnik and Cooper, 2017), enthymematic reasoning (Breitholtz, 2014), clarification requests (Purver, 2006; Ginzburg, 2012), negation (Cooper and Ginzburg, 2012), non-sentential utterance resolution (Fernández *et al.*, 2007; Ginzburg, 2012) and iconic gesture (Lücking, 2016). Being based on types and record-like contexts, we hope that our framework can also benefit from all this literature.

2.1.2. *Information-state update approach*

In this work we are employing an information-state update (ISU) approach, following several authors, including Larsson (2002) and Ginzburg (2012). In this view we present the information available to each participant of the dialogue (either a human or an artificial agent) in a rich information state. Being rich entails that the information state contains a hierarchy of facts, including the ones that are thought to be shared and the ones that have not been yet publicised.

Let us now consider the *update*, another essential component of ISU. In this case, we rely on a set of rules that will govern the updates. For instance, Ginzburg (2012) defines one of the most basic rules – the rule of QUD-incrementation – the procedure of updating the current set of questions under discussion (QUD) if the latest utterance is a question. This operation is salient to a user and therefore it constitutes the update of the public part of the information state.

The main benefit of using a rich representation of the information state with underspecified components is to be able to address a wide range of clarifications from both parties. This is especially beneficial in the case of automatic speech recognition or natural language understanding errors. But even putting such errors aside, we can

also consider topically relevant follow-up questions by the system, or contributions when the user provides more information than they were asked (over-answering).

2.1.3. *Questions and clarifications*

One of the greatest challenges in theoretical semantics and pragmatics is the treatment of interrogatives in the context of dialogue (Wiśniewski, 2015; Ginzburg, 2012). Here we distinguish *questions* as a general surface form and more contextualised forms of them, such as questions that initiate side sequences and constitute clarification requests (CRs). Side sequences usually refer to introducing some new question under discussion, for instance, requesting some additional information, whereas clarification requests generally account for cases of non-understanding, but the boundaries between them are often blurred. In the current study we exemplify our approach by accounting for requests for additional information, but it is only tested for the cases of system-initiated CRs.

For spoken dialogue systems it is crucial to be able to produce and process clarifications requests (Purver, 2004). Even though this is not our focus here, in the context of the low confidence of speech recognition and NLU, the system could clarify its input with the user. Further, with recent advances in speech recognition and statistical NLU, users expect to be able to initiate CRs themselves. Because our theory is symmetric with respect to users and systems roles, it can be useful in this context.

2.2. *Proof search as a programming language*

The prevailing tradition in formal semantics, including in most pieces of work cited above, is to represent (declarative) statements as propositions, formalised in an underlying logic (often first-order logic). In particular, in linguistic theories based on intuitionistic logic (such as TTR), true statements corresponds to propositions which admit a proof.

There is a long history of using proof search as a declarative programming paradigm, where the programmer specifies *axioms* and *rules of inference* which model their application domain. Typically such a system of axioms and rules represents a database of facts. For example, the axiom $(Leave\ 55\ \ Valand\ 11.50)$ can model the fact that bus 55 leaves from Valand at 11:50. The rule $(Leave\ x\ \ Valand\ y \rightarrow Arrive\ x\ \ CentralStationen\ (y + 45\ minutes))$ can represent travelling times on a certain line.

Then, the user may define a query (or goal) as a logical formula. The system can then search for a proof of a goal as a way to query the database of facts. Often, goals contain *metavariables,*[2] which play the role of unknowns for unification: their value can be fixed to any term for a goal to be reached. For example, the goal

2. Here, we use the convention that metavariables start with a lowercase letter, and constants (including predicates) with an upper case.

$(Leave\ x\ Valand\ y)$ corresponds to a request to list all the buses leaving from Valand (as x) together with their departure time (as y).

Because statements are propositions, it is only natural to use proof-search as a means to represent possible moves in dialogue seen as a game (Larsson, 2002).

2.3. *Linear logic as a Dialogue Management Framework*

Typically, and in particular in the archetypal logic programming language prolog (Bratko, 2001), axioms and rules are expressed within the general framework of first-order logic. However, several authors (Dixon *et al.*, 2009; Martens, 2015) have proposed to use linear logic (Girard, 1995) instead. For our purpose, the crucial feature of linear logic is that hypotheses may be used *only once*. For example, one could have a rule $IsAt\ x\ Valand\ y \multimap IsAt\ x\ CentralStationen\ (y + 45\ minutes)$. Consequently, after firing the above rule, the premiss $(Is\ x\ Valand\ y)$ becomes unavailable for any other rule. Thereby the linear arrow $\multimap$ can be used to conveniently model that a bus cannot be at two places simultaneously. [3]

In general, the linear arrow corresponds to *destructive state updates*. Thus, the hypotheses available for proof search correspond to the *state* of the system. In our application they will correspond to the *information state* of the dialogue participant. [4]

This way, firing a linear rule corresponds to triggering an *action* of an agent, and a complete proof corresponds to a *scenario*, i.e. a sequence of actions, possibly involving action from several agents. Hence, the actions realised as actual interactions constitute the observable dialogue. That is, an action can result in sending a message to the outside world (in the form of speech, movement, etc.). Conversely, events happening in the outside world can result in updates of the information state (through a model of the perceptory subsystem).

At any point in the scenario, the multiset of available *linear hypotheses* represents the current information-state of the agent which is modelled. To clarify, the information-state (typically in the literature and in this paper as well), corresponds to the state of a *single* agent. Thus, a scenario is conceived as a sequence of actions and updates of the information state of a single agent a, even though such actions can be attributed to any other dialogue participant b. (That is, they are a's representation of actions of b.)

3. If several arrows are present in a rule (such as $A \multimap B \multimap C$) then both A and B are consumed and C is produced.

4. We note, that in linear logic, facts (or hypotheses) do not come in a hierarchy. Either we have a fact, or we don't. However, in second-order variants of intuitionistic logic, like the one we use, one can conveniently wrap propositions in constructors, to indicate that they come with a qualification. For example, we can write $Unsure\ P$ to indicate that the proposition P may hold (for example if clarification is required).

To reiterate, in our implementation, the information-state can be queried using *rules* (such as those we list below). Because they are linear, these hypotheses can also be removed from the state, as we discuss in detail in section 4.

It is important to note that we will not forego the unrestricted (i.e. non-linear) implication ($\rightarrow$). Rather, both implications will co-exist in our implementation, thus we can represent simultaneously transient facts, or states (introduced by the linear arrow) and immutable facts (introduced by the unrestricted arrow). Besides, we have a *fixed* set of rules (they remain available even after being used), such as $(IsAt\ x\ Valand\ y \multimap IsAt\ x\ CentralStationen\ (y + 45\ minutes))$ above. Each such rule manipulates a part of the information state (captured by its premisses) and leaves everything else in the state unchanged.

3. Questions and clarifications

3.1. *Question-answering with metavariables*

In this subsection we show how a metavariable can represent what is being asked, as the unknown in a proposition. A first use for metavariables is to represent the requested answer of a question.

In this paper, we represent a question by a predicate P over a type A. That is, using a typed intuitionistic logic:

$$A : Type \qquad\qquad P : A \rightarrow Prop$$

The intent of the question is to find out about a value x of type A which makes $P\ x$ true, or at least entertained by the other participant. We provide several examples in Table 1. It is worth stressing that the type A can be large (for example asking for any location) or as small as a boolean (if one requires a simple yes/no answer). We note in passing that, typically, polar questions can be answered not just by a boolean but by qualifing the predicate in question, for example "maybe", "on Tuesdays", etc. (Table 1, last two rows). In this instance $A = Prop \rightarrow Prop$.

One complication are polar questions phrased in the negative (Cooper and Ginzburg, 2012); for example: "Doesn't John like bananas?". In this instance, a simple "no" answer can be ambiguous, and a possible model would be a multi-valued kind of answer ("yes he does" represented as $DefiniteYes$; "no he doesn't", represented as $DefiniteNo$, "no" as $AmbiguousNo$, and "He does in the weekend" as $Qualifier\ OnWeekend$):

$$\begin{array}{lll} Q\ Multi\ (\lambda x.\mathbf{case}\ x\ \mathbf{of} & AmbiguousNo & \rightarrow Trivial \\ & DefiniteNo & \rightarrow \neg\ (Like\ John\ Bananas) \\ & DefiniteYes & \rightarrow Like\ John\ Bananas \\ & Qualifier\ m & \rightarrow m\ (Like\ John\ Bananas)) \end{array}$$

To represent ambiguity in the case of $AmbiguousNo$, we make the answer provide no information, in the form of a trivial proposition (which is always true regardless

of context). This is a natural account, because the meaning of short answers (such as "no") always depends on the context. ("Paris" does not mean the same thing in the context of "Where do you live?" as in the context "Where were you born?".) Additionally, in the framework of a full dialogue management system, the $AmbiguousNo$ case should be treated as unresolving (the question effectively remains unanswered). However, in such a framework, it is always possible to receive a biasing answer ("I don't know") or no answer whatsoever. Even more complications are possible, by introduction of cases such as rhetorical questions. We deem such complications out of the scope of the current paper.

Within the state of the agent, if the value of the requested answer is represented as a metavariable x, then the question can be represented as: $Q\ A\ x\ (P\ x)$. That is, the pending question (Q denotes a question constructor) is a triple of a type, a metavariable x, and a proposition where x occurs. We stress that $P\ x$ is *not* part of the information state of the agent yet, rather the fact that the above question is *under discussion* is a fact. For example, after asking "Where does John live?", we have:

$$haveQud : QUD\ (Q\ Location\ x\ (Live\ John\ x))$$

Resolving a question can be done by communicating an answer. An answer to a question $(A : Type; P : A \rightarrow Prop)$ can be of either of the two following forms: i) A **ShortAnswer** is a pair of an element $X : A$ and its type A, represented as $ShortAnswer\ A\ X$ or ii) An **Assertion** is a proposition $R : Prop$, represented as $Assert\ R$. Therefore, one way to process a short answer is by the $processShort$ rule:

$$\begin{aligned} processShort : (a : Type) \rightarrow (x : a) \rightarrow (p : Prop) \rightarrow \\ ShortAnswer\ a\ x \multimap QUD\ (Q\ a\ x\ p) \multimap p \end{aligned}$$

Above we use Π type binders to declare (meta)variables (written here $(a : Type) \rightarrow$, $(x : a) \rightarrow$, etc.). This terminology will make sense to readers familiar with dependent types. For the others, such binders can be thought as universal quantification ($\forall a, \forall x$, etc.), the difference is that the type of the bound variable is specified. (The reader worried about any theoretical difficulty regarding mixing linear and dependent types is directed to Atkey (2018) and Abel and Bernardy (2020).)

We demand in particular that types in the answer and in the question match (a occurs in both places). Additionally, because x occurs in p, the information state will mention the concrete x which was provided in the answer. For example, if the QUD was $(Q\ Location\ x\ (Live\ John\ x))$ and the system processes the answer $ShortAnswer\ Location\ Paris$, then x unifies with $Paris$, and the new state will include $Live\ John\ Paris$.

To process assertions, we can use the following rule:

$$\begin{aligned} processAssert : (a : Type) \rightarrow (x : a) \rightarrow (p : Prop) \rightarrow \\ Assert\ p \multimap QUD\ (Q\ a\ x\ p) \multimap p \end{aligned}$$

That is, if (1) p was asserted, and (2) the proposition q is part of a question under discussion, and (3) p can be unified with q (we ensure this unification by simply using

question	A	P	reply	x
Where does John live?	$Location$	$\lambda x.Live\ John\ x$	in London	$ShortAnswer\ Location\ London$
Does John live in Paris?	$Bool$	$\lambda x.\mathbf{if}\ x\ \mathbf{then}\ (Live\ John\ Paris)$ $\mathbf{else}\ Not\ (Live\ John\ Paris)$	yes	$ShortAnswer\ Bool\ True$
What time is it?	$Time$	$\lambda x.IsTime\ x$	It is 5am.	$Assert\ (IsTime\ 5.00)$
Does John live in Paris?	$Prop \to Prop$	$\lambda m.m\ (Live\ John\ Paris)$	yes	$ShortAnswer\ (Prop \to Prop)$ $(\lambda x.x)$
Does John live in Paris?	$Prop \to Prop$	$\lambda m.m\ (Live\ John\ Paris)$	from January	$ShortAnswer\ (Prop \to Prop)$ $(\lambda x.FromJanuary\ (x))$

Table 1. *Examples of questions and the possible corresponding answers. The type A is the type of possible short answers. The proposition $P\ x$ is the interpretation of a short answer x. The x column shows the formal representation of a possible answer, either in short form or assertion form.*

the same metavariable p in both roles in the above rule), then the assertion resolves the question. Additionally, the metavariable x is made ground to a value provided by p, by virtue of unification of p and q. For example, "John lives in Paris" answers both questions "Where does John live?" and "Does John live in Paris?" (there is unification), but, not, for example "What time is it?" (there is no unification). Note that, in both cases ($processAssert$ and $processShort$), the information state is updated with the proposition posed in the question.

3.2. *Notion of unique and concrete values*

However, one should consider the question resolved only if the answer is "unique". For example, the assertion "John lives somewhere" generally does not resolve the question "Where does John live?". That is, if "somewhere" is represented by a metavariable, then the answer is not resolving.

Assume a two-place predicate Eat with agent as first argument and object as second argument. The phrase "John eats Mars" could then be represented as $(Eat\ John\ Mars)$. According to our theory, one can then represent the phrase "John eats" as $(Eat\ John\ x)$, with x being a metavariable. Assume now a system with the following state:

$$Eat\ John\ Mars$$

Then the question "What does John eat?, represented as $(Q\ Food\ x\ (Eat\ John\ x))$, can be answered. From the point of view of modelling with linear logic, we could attempt to model the answering by the rule as follows:

$$(a : Type) \to (x : a) \to (p : Prop) \to \\ QUD\ (Q\ a\ x\ p) \to p \multimap (p \otimes Answer\ x\ (Q\ a\ x\ p))$$

Note: taking a linear argument and producing it again is a common pattern, which can be spelled out $A \multimap (A \otimes P)$. It is so common that from here on we use the syntactic sugar $A \twoheadrightarrow P$ for it, so the above rule will be written:

$$(a : Type) \to (x : a) \to (p : Prop) \to \\ QUD\ (Q\ a\ x\ p) \to p \twoheadrightarrow Answer\ x\ (Q\ a\ x\ p)$$

The above states that if x makes the proposition p true (more precisely, provable — we require that p is a fact in the last argument) then it is valid to answer x if $Q\ a\ x\ p$ is under discussion. However, there is an issue with the above rule: there are several values making p true, i.e. if x is *not unique*, then intuitively one would not consider x a suitable answer. Indeed, assume instead that the system is in the state:

$$Eat\ John\ x$$

Then the question cannot be answered, because x stands for some unknown thing. The proper answer is then "I do not know".

Hence, we introduce another type-former $(x : A) \rightarrow_! B$. As for $(x : A) \rightarrow B$, it introduces the metavariable x. However, the rule fires only when x is made *ground* (it is bound to a term which does not contain any metavariable) and *unique* by matching the rule — this is what we call a unique and concrete value. That is, it won't match in the previous example, because the answer is not made ground (it contains unknowns). Additionally, it won't match if the state of the system is composed of the two hypotheses $(Eat\ John\ Mars)$ and $(Eat\ John\ Twix)$: the answer is not unique.

Thus, the rule for answering can be written like so:

$$produceAnswer : (a : Type) \rightarrow (x : a) \rightarrow_! (p : Prop) \rightarrow \\ QUD\ (Q\ a\ x\ p) \rightarrow p \rightarrowtriangle ShortAnswer\ a\ x$$

For example, if we have the following state:

$$QUD\ (Q\ Food\ x\ (Eat\ John\ x)) \\ Eat\ John\ Mars$$

The system can unify $QUD\ (Q\ Food\ x\ (Eat\ John\ x))$ and $QUD\ (Q\ a\ x\ p)$, yielding $a = Food$ and $p = (Eat\ John\ x)$. Then, we search for a proof p, and to do this, we can unify $(Eat\ John\ x)$ with $(Eat\ John\ Mars)$, giving finally the answer $x = Mars$ and therefore the state becomes:

$$Eat\ John\ Mars \\ ShortAnswer\ Food\ Mars$$

Note that the fact $Eat\ John\ Mars$ is found both as hypothesis and a conclusion of $produceAnswer$, and therefore it remains in the information state.

3.3. *Clarification requests and follow-up questions*

In this section we discuss an alternative kind of responding, which is to issue clarification requests. To see how they can occur, consider again the question "What does John eat?", in the information state $Eat\ John\ Mars$ and $Eat\ John\ Twix$. A proper answer could be "Mars and Twix" or even "Mars or Twix". However we consider here a third possibility: instead of answering, the agent can issue a clarification request.

To illustrate, consider the question "What is being eaten?" represented as $Q\ x\ (Eat\ y\ x))$, with the state

$$Eat\ John\ Mars \\ Eat\ Mary\ Mars$$

Then the agent can unambiguously answer "Mars": even if we do not know who we're talking about, it does not matter: only Mars is being eaten. However, if the state is

$Eat\ John\ Mars$
$Eat\ Mary\ Twix$

then, a probable answer would be a *clarification request*, namely "By whom?".

To detect situations where a clarification request can be issued, we can use the following rule (we leave unspecified the exact form of the CR abstract for now and come back to it below in section 4):

$$[\, a : Type; x : a; p : Prop; qud :: QUD\ (Q\ x\ p); proof :: p\,] \rightarrow_?\ CR$$

The conditions are similar to that of the answering rule. The principal difference is the use of the $\rightarrow_?$ operator, which takes as left operand the specification of a request and tests whether it has a non-unique solution or cannot be made fully ground. Essentially this does the opposite of the $\rightarrow_!$ operator. However, because the components of the query are indeterminate, they cannot be fixed when firing the rule, and therefore the state update cannot depend on them. Therefore we use a record syntax to limit their scope, ensuring that they won't occur in the state update. Such a record can be understood as a conjunction which additionally binds components to field names. Additionally, note the use of the single colon (:) for metavariables and the double colon for information-state hypotheses (::).

We can then turn our attention to the formulation of this clarification request. It is itself a question, and has a tricky representation:

$$Q\ Person\ z\ (z = y)$$

That is, the question is asking about some aspect which was left implicit in the original question (what is being eaten). In our terms, it must refer to the metavariable (y) which the original question included. After getting an answer (say $Mary$), z will be bound to a ground term, and, in turn, the fact $z = y$ will ensure that y becomes ground.

$Eat\ John\ Mars$
$Eat\ Mary\ Twix$
$ori :: QUD\ (Q\ Food\ x\ (Eat\ y\ x))$
$cr\ \ :: QUD\ (Q\ Person\ z\ (z = y))$
$a\ \ \ :: ShortAnswer\ Person\ Mary$

after applying $processShort$:

$Eat\ John\ Mars$
$Eat\ Mary\ Twix$
$ori :: QUD\ (Q\ Food\ x\ (Eat\ y\ x))$
$r\ \ \ :: Mary = y$

This means the original question will, by unification, become $Q\ \ Food\ \ x\ \ (Eat\ \ \ Mary\ \ \ x)$, and it can be unambiguously answered using the

$produceAnswer$ rule. We note that the logical form of the question (z such that $z = y$) is typically realised in a complicated way. In our example, it could be "By whom?"; echoing part of the original question and assuming cooperative communication so that the questioner properly relates the clarification request to the implicits of the original questions. In practice, the form of clarification questions will greatly vary depending on the context (Purver, 2004).

The above presupposes a clear-cut distinction: if an answer is unique, it is given; otherwise a clarification request is issued. However, answers could simply be exhaustive ("Mars or Twix"). If the original questioners are unhappy with the ambiguity, they are free to issue more precise questions. In practice, one can easily imagine an ambiguity threshold after which clarification requests are preferred. In the simplest form, this ambiguity threshold could be expressed by the length of the answer. In our example, if one has to list, say, 20 different kinds of food, it is easy to imagine that the answer won't be fully given. In fact, this question can be the topic of an experimental study.

3.3.1. *Clarification via adding extra arguments*

The scope of what is subject to clarification is anything which can be represented as an argument in a relation. For instance, consider the question "Where does John live?" with the short answer "Paris". The questionee may decide that there is some ambiguity about *which* location one is talking about — after all there are several places with this name. To be able to model this, the $Live$ relation needs to be generalised to be a 3-place predicate, where the country is specified.

However most of the time one may choose to leave this parameter implicit. This is what is done for example when asking the above question:

$$Q\ Location\ x\ (Live\ John\ x\ y)$$

If the question can be answered without regard for the country, then the metavariable will remain free for the duration of the dialogue. If on the other hand, answering the question demands clarification, this can be done using the mechanisms described above. In sum, in our model, to support clarification requests, a system must integrate many arguments and use metavariables.

The same technique can apply to polar questions. Considering "Does John live in Paris?", we can assume that the question can be encoded (for simplicity) as $\lambda x.\mathbf{if}\ x\ \mathbf{then}\ (Live\ John\ Paris\ y)\ \mathbf{else}\ Not\ (Live\ John\ Paris\ y)$.

If the system has the following facts:

$$Live\ John\ Paris\ France$$
$$Not\ (Live\ John\ Paris\ Denmark)$$

then both "True" and "False" are valid answers, and a clarification requests should be issued: $Q\ Country\ z\ (z = y)$. We see again that the realisation of the clarification request depends highly on the formulation of the question and the context. In this case "Do you mean Paris, France?" would be suitable.

3.3.2. *Clarification via adding named contextual parameters*

The above presentation (using a ternary predicate) is useful conceptually, but not ideal in practice: in the most general case one would end up with predicates with lots of arguments, for example country, county, district, etc.

However, there is a standard solution to the issue: because the country is functionally dependent on the location, these two concepts should be linked directly together rather than involve the $Live$ predicate. Using an intermediary entity type for locations and binary predicates, one can represent the question "Does John live in Paris?" as follows:

$$\lambda x.\mathbf{if}\ x\ \mathbf{then}\ (Live\ John\ y \rightarrow Name\ y\ Paris)$$
$$\mathbf{else}\ Not\ (Live\ John\ y \rightarrow Name\ y\ Paris)$$

Literally, "Does John live in a place called Paris?". The ambiguity of the $Paris$ name can be represented by several locations named $Paris$, X and Y in our illustration: [5]

$$Name\ X\ Paris$$
$$Name\ Y\ Paris$$
$$Live\ John\ X$$
$$Not\ (Live\ John\ Y)$$
$$Country\ France\ X$$
$$Not\ (Country\ France\ Y)$$

Because John lives in X but not in Y the question is ambiguous. One way to lift the ambiguity is to raise the clarification request as above. Here it can be phrased as a polar question [6] again:

$$Q\ Bool\ (\lambda x.\mathbf{if}\ x\ \mathbf{then}\ Country\ France\ y\ \mathbf{else}\ Not\ (Country\ France\ y))$$

3.3.3. *Summary*

In sum, we leverage a feature of linear-logic proof search: at any point in the scenario, the context can refer to metavariables. In a dialogue application, metavariables represent a certain amount of flexibility in the scenario: *so far* the scenario works for any value which could be assigned to the metavariable. This means that at a further point the metavariable can be instantiated to some other value.

4. KoS-inspired dialogue management with linear logic

In this section we integrate our question/answering framework within more complete dialog manager (DM). We stress that this DM models the information-state of

5. The combination of negation and proof search leads to complications which are out of scope here, for this reason we simply assume that negated predicates are available in the information-state.

6. Here we use the simpler version of the treatment of polar questions.

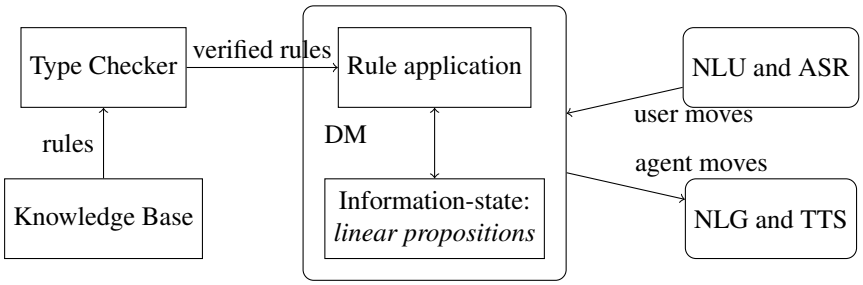

Figure 1. *Architecture of a spoken dialogue system with a dialogue manager based on a linear logic framework.*

only one participant. Regardless, this participant can record its own beliefs about the state of other participants. Figure 1 shows how such a DM can be integrated into a spoken dialogue system. In general, the core of DM is comprised of a set of linear-logic rules which depend on the domain of application. However, many rules will be domain-independent (such as generic processing of answers). We show these generic rules first, and then illustrate them with an example application.

4.1. *Domain-independent rules*

4.1.1. *Interface with language understanding and generation*

To be useful, a DM must interact with the outside world, and this interaction cannot be represented using logical rules, which can only manipulate data which is already integrated in the information state. Here, we assume that the information that comes from sources which are external to the dialogue manager is expressed in terms of semantic interpretations of moves, and contains information about the speaker and the addressee in a structured way. We provide 5 basic types of moves, specified with a speaker and an addressee, as an illustration:

$$
\begin{array}{ll}
Greet & spkr\ addr \\
CounterGreet & spkr\ addr \\
Ask & question\ spkr\ addr \\
ShortAnswer & vtype\ v\ spkr\ addr \\
Assert & p\ spkr\ addr
\end{array}
$$

These moves can either be received as input or produced as outputs. If they are inputs, they come from the NLU component, and they enter the context with $Heard : Move \rightarrow Prop$ predicate. For example, if one hears a greeting, the proposition $Heard\ (Greet\ S\ A)$ is added to the information state/context, without any rule being fired — this is what we mean by an external source.

If they are outputs, to be further used by the NLG component, some rule will place them in $Agenda$. For example, to issue a countergreeting, a rule will place the proposition $Agenda\ (CounterGreet\ A\ S)$ in the information state.

Thereby each move is accompanied by the information about who has uttered it, and towards whom was it addressed. All the moves are recored in the $Moves$ part of the participant's dialogue gameboard, as a $Cons$-list (stack).

Additionally, we record any move m which one has yet to actively react to, in an hypothesis of the form $Pending\ m$. We cannot use the $Moves$ part of the state for this purpose, because it is meant to be static (not to be consumed). $Pending$ thus allows one to make the difference between a move which is fully processed and a pending one.

4.1.2. *Initial state*

In general, we start with empty QUD and $Agenda$. A non-empty QUD can be prepared if, in a certain domain, some open questions are assumed from the start. The $Agenda$ might not be empty if one wants the system to initiate the conversation. There are also no moves: nothing has been said by either party.

$$\_ :: QUD\ Nil; \_ :: Agenda\ Nil; \_ :: Moves\ Nil;$$

(We often do not care about the proof object witnessing a propositions, in which case we denote it with an underscore.)

4.1.3. *Hearing*

The capacity of "hearing" or, in other words, starting the processing of semantic representations of utterances from the NLU component is implemented with the following rule:

$$\begin{array}{l} hearAndRemember : \\ \quad (m : DP \to DP \to Move) \to (x\ y : DP) \to (ms : List\ Move) \to \\ \quad Heard\ (m\ x\ y) \quad \multimap Moves\ ms \multimap HasTurn\ x \multimap \\ \quad [\_ :: Moves\ (Cons\ (m\ x\ y)\ ms); \_ :: Pending\ (m\ x\ y); \_ :: HasTurn\ y] \end{array}$$

where $(m\ \ x\ \ y)$ is a semantic representation of the utterance. Here we produce a record, whose fields will all be added to the information state. The rule demands that participant x has the turn and, as a result, turn was taken by his partner y.[7] The DP type stands for *dialogue participant*. As a result we do several things: i) place the move in a move list for further references ($PushMove$), ii) record the turn-switching (which in a complete system may not apply to all cases — then additional hypotheses would be added), and iii) prepare to process the move ($Pending$).

7. For now we have a very simple model of turn-taking, which can be improved in many ways: certain moves may not induce turn-change, there can be more than two participants, etc.

4.1.4. *Uttering*

The capacity of "uttering" represents an ability to generate information for the NLG component. NLP component is represented by $Agenda$ that contains a move that is just about to be uttered.

$$\begin{aligned}&utterAndRemember:\\&\quad(m:DP\to DP\to Move)\to(ms:List\ Move)\to(x\ y:DP)\to\\&\quad Agenda\ (m\ x\ y)\multimap Moves\ ms\multimap HasTurn\ x\multimap\\&\quad[\_::Utter\ (m\ x\ y);\_::Moves\ (Cons\ (m\ x\ y)\ ms);\_::HasTurn\ y]\end{aligned}$$

Here also we take care of turn-taking in the same rule. As a result, the system consumes the $Agenda$ and passes the move to the NLG component. The move is also memorised in the $Moves$ stack.

4.1.5. *Basic adjacency: greeting*

We can show how basic move adjacency can be defined in the example of counter-greeting preconditioned by a greeting from the other party:

$$\begin{aligned}counterGreeting:(x\ y:DP)\to HasTurn\ x\twoheadrightarrow Pending\ (Greet\ y\ x)\multimap\\ Agenda\ (CounterGreet\ x\ y)\end{aligned}$$

4.1.6. *QUD incrementation*

Another important rule accounts for pushing the content of the last move, in the case if it is an Ask move, on top of the questions under discussion (QUD) stack.

$$\begin{aligned}pushQUD:(q:Question)\to(qs:List\ Question)\to(x\ y:DP)\to\\ Pending\ (Ask\ q\ x\ y)\multimap QUD\ qs\multimap QUD\ (Cons\ q\ qs)\end{aligned}$$

4.1.7. *Integrating the answers*

If the user asserts something that relates to the top QUD, then the QUD can be resolved and therefore removed from the stack. The corresponding proposition p is saved as a $UserFact$.[8] This rule extends the abstract rule that were introduced in section 3.3.

$$\begin{aligned}&processAssert:(a:Type)\to(x:a)\to(p:Prop)\to(qs:List\ Question)\to\\&\quad(dp\ dp1:DP)\to Pending\ (Assert\ p\ dp1\ dp)\multimap\\&\quad QUD\ (Cons\ (Q\ dp\ a\ x\ p)\ qs)\multimap[\_::UserFact\ p;\_::QUD\ qs]\end{aligned}$$

Short answers are processed in a very similar way to assertions:

8. For the current purposes we only remove the top QUD, but in a more general case we can implement the policy that can potentially resolve any QUD from the stack.

$$
\begin{array}{l}
processShort : (a : Type) \to (x : a) \to (p : Prop) \to (qs : List\ Question) \to \\
\quad (dp\ dp1 : DP) \to Pending\ (ShortAnswer\ a\ x\ dp1\ dp) \multimap \\
\quad QUD\ (Cons\ (Q\ dp\ a\ x\ p)\ qs) \multimap [\_ :: UserFact\ p; \_ :: QUD\ qs]
\end{array}
$$

4.1.8. *Questions and clarifications*

Just as we described in 3.2, we use uniqueness check to determine whether system can resolve the question ($produceAnswer$) or it needs to initiate a clarifying side sequence ($produceCR$).

$$
\begin{array}{l}
produceAnswer : \\
\quad (a : Type) \to (x : a) \to_! (p : Prop) \to (qs : List\ Question) \to \\
\quad QUD\ (Cons\ (Q\ USER\ a\ x\ p)\ qs) \multimap p \multimap \\
\quad [\_ :: Agenda\ (ShortAnswer\ a\ x\ SYSTEM\ USER); \_ :: QUD\ qs; \\
\quad\ \_ :: Answered\ (Q\ USER\ a\ x\ p)] \\
produceCR : \\
\quad [a : Type; x : a; p : Prop; qs : List\ Question; \\
\quad\ \_ :: QUD\ (Cons\ (Q\ USER\ a\ x\ p)\ qs); \_ :: p] \to_? CR
\end{array}
$$

The clarifying side sequence itself (CR) is meant to be specified by a dialogue developer, possibly informed by machine-learning systems, because it is domain-specific and the choice of the spectrum of possible options is wide. We provide an example of a domain-specific CR in the section 4.2 below.

4.2. *Example*

We now show how the generic system of rules above can handle the exchange:

U: Hello!
S: Hello, U.
U: When is there a bus from Valand?
S: In 15 minutes.

Let us further assume the following system context, which contains up-to-date public transport information in the following format:

$$TT\ Bus\ Time\ Origin\ Destination$$

This is added to the initial domain-independent context outlined above. We also assume that the user has the turn at the start.

$$
\begin{array}{ll}
QUD & Nil \\
Agenda & Nil \\
HasTurn & U \\
Moves & Nil
\end{array}
$$

When the system hears the greeting it can be integrated into the state using $hearAndRemember$ rule, therefore system updates its state accordingly:

$$\begin{array}{ll} QUD & Nil \\ Agenda & Nil \\ HasTurn & S \\ Moves & [Greet\ U\ S] \end{array}$$

(To save space we use a list notation from now on, [A, B, C] is a shorthand for $(Cons\ A\ (Cons\ B\ (Cons\ C)))$.) In this context the system can issue a counter-greeting by firing the $counterGreeting$ rule:

$$\begin{array}{ll} Agenda & (CounterGreet\ S\ U) \\ HasTurn & S \\ Moves & [Greet\ U\ S] \end{array}$$

Everything which is on the agenda can be uttered using $utterAndRemember$ rule, given that the system has the turn. System also hands the turn over to the user. Therefore, the state becomes (we use bracket syntax instead of $Cons$ for readability):

$$\begin{array}{ll} HasTurn & U \\ Moves & [CounterGreet\ S\ U, Greet\ U\ S] \end{array}$$

Now the system hears the question $(Ask\ (Q\ U\ Time\ t0\ (TT\ n0\ t0\ Valand\ d0)))$. It is domain-specific, and basically requests the timetable information for the given departure station. Again, we use $hearAndRemember$ rule to itegrate it into state, but also, because the move is Ask, the system sets its QUD to the question that the move contains with the $pushQUD$ rule.

$$\begin{array}{ll} QUD & [\ Q\ U\ Time\ t0\ (TT\ n0\ t0\ Valand\ d0)] \\ HasTurn & S \\ Moves & [\ Ask\ (Q\ U\ Time\ t0\ (TT\ n0\ t0\ Valand\ d0))\ U\ S, \\ & \quad CounterGreet\ S\ U, Greet\ U\ S] \end{array}$$

Now, depending on the state of the knowledge base, the system will have two options: i) produce the answer straight away, or ii) integrate a clarifying side sequence.

4.2.1. *Straight answer*

For this case we will consider a knowledge base that includes information just about the unique (w.r.t. the time) entry in the timetable:

$$TT\ B18\ T15\ Valand\ Johanneberg$$

Therefore the question can be resolved and the resolving short answer can be put on the $Agenda$.

$$\begin{array}{ll} Answered & (Q\ U\ Time\ T15\ (TT\ B18\ T15\ Valand\ Johanneberg)) \\ QUD & Nil \end{array}$$

$HasTurn\ \ S$
$Agenda\quad (ShortAnswer\ Time\ T15\ S\ U)$
$Moves\quad [\dots]$ -- same as above

4.2.2. *Clarifying side sequence*

In contrast, we can extend our minimal timetable example with another entry, therefore making it non-unique, w.r.t. time.

$TT\ B18\ T15\ Valand\ Johanneberg$
$TT\ B55\ T20\ Valand\ SciencePark$

In order to make it unique we can either clarify the bus number or the destination. For the bus number the rule for clarification can be formulated as follows:

$$
\begin{array}{l}
specificCR: \\
\quad (t : Time) \to (n : Bus) \to (s\ d : Location) \to (qs : List\ Question) \to \\
\quad CR \multimap QUD\ (Cons\ (Q\ U\ Time\ t\ (TT\ n\ t\ s\ d))\ qs) \multimap \\
\qquad [\_ :: QUD\ (Cons\ (Q\ S\ Bus\ n\ (WantBus\ n)) \\
\qquad\quad (Cons\ (Q\ U\ Time\ t\ (TT\ n\ t\ s\ d))\ qs)); \\
\qquad\ \ \_ :: Agenda\ (Ask\ (Q\ S\ Bus\ n\ (WantBus\ n))\ S\ U)]
\end{array}
$$

As a result of applying it, the state becomes:

$Agenda\quad (Ask\ (Q\ S\ Bus\ n0\ (WantBus\ n0))\ S\ U)$
$QUD\quad [Q\ S\ Bus\ n0\ (WantBus\ n0),$
$\qquad\ Q\ U\ Time\ t0\ (TT\ n0\ t0\ Valand\ d0)]$
$HasTurn\ S$
$Moves\quad [\dots]$ -- same as above

Then, the system can utter the clarification request ($utterAndRemember$ rule):

$QUD\ [Q\ S\ Bus\ n0\ (WantBus\ n0), Q\ U\ Time\ t0\ (TT\ n0\ t0\ Valand\ d0)]$
$HasTurn\ S$
$Moves\ [Ask\ (Q\ S\ Bus\ n0\ (WantBus\ n0))\ S\ U$
$\qquad\ Ask\ (Q\ U\ Time\ t0\ (TT\ n0\ t0\ Valand\ d0))\ U\ S$
$\qquad\ CounterGreet\ S\ U, Greet\ U\ S]$

The user can reply to this with a short answer $ShortAnswer\ Bus\ B55\ U\ S$ or an assertion $Assert\ (WantBus\ B55)\ U\ S$, which can be integrated using $processShort$ or $processAssert$ rule respectively. We show the state after processing the short answer:

$QUD\ [Q\ U\ Time\ t0\ (TT\ B55\ t0\ Valand\ d0)]$
$UserFact\ (WantBus\ B55)$
$HasTurn\ S$
$Moves\ [ShortAnswer\ Bus\ B55\ U\ S,$
$\qquad\ Ask\ (Q\ S\ Bus\ B55\ (WantBus\ B55))\ S\ U, \dots]$

The reader can see that the metavariable $n0$ from the previous state is now unified with $B55$ in the QUD, therefore it now corresponds to one unique entry in the knowledge base. Hence, the answer can be issued by the $produceAnswer$ rule.

$Answered$ $(Q\ U\ Time\ T20\ (TT\ B55\ T20\ Valand\ SciencePark))$
QUD Nil
$Agenda$ $(ShortAnswer\ Time\ T20\ S\ U)$
$UserFact$ $(WantBus\ B55)$
$HasTurn$ S
$Moves$ $[\dots]$ -- same as above

5. Related work

The present work provides a minimal and fine-grained account for clarification requests initiated by any conversational party, following accounts of and supporting a subset of cases thoroughly investigated in the CLARIE Prolog-based system (Purver, 2006), following corpus studies by Purver (2004) and Rodríguez and Schlangen (2004).

One of our main sources of inspiration is Ginzburg's KoS (Ginzburg, 2012). However we recast it in the framework of proof search, and linear logic. We have argued that this has many advantages. First, it affords the use of metavariables to represent uncertainty, which is absent from TTR. Second, expressing updates using linear logic rules means that only the relevant parts of the information state must be dealt with in any given rule. Cooper's TTR has a special "asymmetric merge" operator for this purpose, but it is a less-studied *ad-hoc* addition to type-theory, though see *inter alia* (Grover *et al.*, 1994). As it stands, KoS is lacking implementations, with the exception of the work of Maraev *et al.* (2018), who adapt KoS to eschew the assymetric merge operation. An oft-touted advantage of TTR is that propositions are witnessed by proof objects. We benefit from the same advantage: we use an intuitionistic system, and as such every proposition in the information state is associated a witness, even if we have not shown them for concision (they play little role in our analysis).

Larsson (2002) proposed the use of Prolog (and hence, proof search), as a dialogue management framework. However, the lack of linear hypotheses means that destructive information-state updates are sometimes awkward to represent. Besides, he does not consider the use of metavariables to represent uncertainty — even though Prolog in principle has the capacity to do it.

To our knowledge Dixon *et al.* (2009) were the first to advocate the use of linear logic for dialogue management and planning. Compared to the present work, they focus primarily on the planning part of dialogue rather than question-answering. In particular, they do not discuss the role of metavariables and clarification requests. We additionally propose the extension of linear logic with special-purpose operators $X \to_! Y$ and $X \to_? Y$ to distinguish the presence or the absence of ambiguity.

6. Evaluation/Discussion/Future work

A kind of dialogue move often studied in parallel to clarifications are *corrections*. It would be elegant if corrections could be formalised in a way similar to clarifications. However, in our analysis, metavariables disappear once they have been grounded. Therefore, corrections cannot involve metavariables and thus require a different treatment. A solution could be to keep metavariables in terms (apply unification substitutions only at the point of testing equality between such variables). We leave a detailed study to further work.

We note that the use of (meta)variables to refer to discourse objects is a very general device. Anything which can be subject to clarification can occur as an argument to predicates. We already showed how "Paris" can be clarified. But we could also clarify "Live" by making the verb be an argument to a general $Apply$ predicate, taking say a verb and its arguments.

Prior studies have noted the phenomenon of semantic dependency relations between questions (Wiśniewski, 2015), e.g. "Who killed Bill?" can be responded by "Who was in town?". The cases of dependencies covered in this study are limited to clarification of metavariables from the original question. This is meant to serve as a proof-of-concept rather than thorough coverage of all possible cases of question dependence. A similar issue concern follow-up questions that are meant to clarify the type of the metavariable, e.g. "What does John like? Do you mean foodwise?". Generally, further work is needed to be carried out in order to extend our system to full-scale coverage of interrelations between QUDs.

A natural progression of this work is to allow the assignment of probabilities to rules and to the components of the state, and to train the probabilities according to the new observations. Our approach follows Lison (2015), which is based on probabilistic rules, but in our case the structure of information state is rich and derived from the theoretical outlook on dialogue, and dialogue management has a core set of domain-independent rules. We can also imagine combining such ideas with probabilistic meaning for sentences (Goodman and Lassiter, 2015; Bernardy *et al.*, 2018).

An important dimension of dialogue processing that the current work does not address is providing a detailed utterance processing of the user and word-by-word incremental processing. This means we cannot deal with form-based parallelism needed for various types of acknowledgements, CRs, and self-repair. Nor, as things stand, do we engage in grounding interaction, modelled extensively by Larsson (2002).

Table 2 originates from Ginzburg and Fernández (2010), who proposed a series of benchmarks for comparing different approaches to developing dialogue systems (see section 2 of that paper). For each approach the symbol ✓ indicates that the current approach safisfies the benchmark in the corresponding row; $\sim$ that the benchmark could be met with some caveats, as explained in the text above for most cases; and — that the benchmark is not met by a standard version of the current approach.

	Benchmark		Example
query and assertion	Q1 simple answers	$\sim$	A: Who slept? B: Bo/Not Bo
	Q2a non-resolving answers	✓	A: Who slept? B: A student.
	Q2b follow up queries	✓	B: A student. A: Who?
	Q3 overinformative answers	✓	A: Who? B: Bo on his own.
	Q4 sub-questions	✓	A: Who? B: Who was here?
	Q5 topic changing	—	
	A1 propositional content update	✓	
	A2 disagreement	$\sim$	A: A student. B: A teacher.
	SC scalability	$\sim$	
	DA domain adaptability	✓	
metacommunication	Ack1 completed acknowledgements	—	A: Move right. B: Mhm.
	Ack2 continuation ack.	—	A: Move- B: mm A: -to the left.
	Ack3 gestural ack.	—	
	CR1 repetition CRs	—	A: Did Bo leave? B: What?
	CR2 confirmation CRs	—	A: Bill left. B: Bill? A: Yes.
	CR3 intended content CRs	✓	A: Where is Bo? B: Which Bo?
	CR4 intention recognition CRs	—	A: Where is the bus? B: Why?
	SND distinct updates	$\sim$	
	FG fine-grained representations	$\sim$	
fragments	SF1 wide coverage of NSUs	$\sim$	
	SF2 basic answer resolution	$\sim$	
	SF3 reprise fragment resolution	—	Bo? $\mapsto$ Who is Bo?
	SF4 long distance short answers	$\sim$	
	SF5 genre sensitive initiating NSUs	$\sim$	(dialogue initially) The Aix bus?
	D1 recognize and repair disfluencies	—	
	D2 keep disfluencies in context	—	

Table 2. *System evaluation. Q5—understand that irrelevant answers imply "change the topic", A2—disagree with user if her utterance is incompatible with own belief, SND—an utterance can give rise to distinct updates across participants. SC—ensure approach scales down to monologue and up to multilogue. For other, more obvious benchmarks we refer our readers to (Ginzburg and Fernández, 2010).*

Acknowledgements

This research was supported by a grant from the Swedish Research Council for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg. We also acknowledge support by a public grant overseen by the French National Research Agency (ANR) as part of the program "Investissements d'Avenir" (reference: ANR-10-LABX-0083). It contributes to the IdEx Université de Paris — ANR-18-IDEX-0001. We also acknowledge a senior fellow-

ship from the Institut Universitaire de France to Ginzburg. In addition, we would like to thank our anonymous reviewers for their useful comments.

7. References

Abel A., Bernardy J.-P., "A unified view of modalities in type systems", *Proceedings of the ACM on Programming Languages*, 2020.

Atkey R., "Syntax and Semantics of Quantitative Type Theory", *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK*, p. 56-65, 2018.

Bernardy J.-P., Blanck R., Chatzikyriakidis S., Lappin S., "A Compositional Bayesian Semantics for Natural Language", *Proceedings of the International Workshop on Language, Cognition and Computational Models, COLING 2018, Santa Fe, New Mexico*, p. 1-11, 2018.

Bratko I., *Prolog programming for artificial intelligence*, Pearson education, 2001.

Breitholtz E., "Reasoning with topoi–towards a rhetorical approach to non-monotonicity", *Proceedings of the 50th anniversary convention of the AISB, University of London*, 2014.

Cooper R., "Austinian truth, attitudes and type theory", *Research on Language and Computation*, vol. 3, n° 4, p. 333-362, 2005.

Cooper R., "Copredication, quantification and frames", *in* S. Pogodalla, J.-P. Prost (eds), *Logical Aspects of Computational Linguistics (LACL 2011)*, Springer, 2011.

Cooper R., "Clarification and generalized quantifiers", *Dialogue and Discourse*, vol. 4, p. 1–25, 2013.

Cooper R., Ginzburg J., "Negative inquisitiveness and alternatives-based negation", *Logic, Language and Meaning*, Springer, p. 32-41, 2012.

Dixon L., Smaill A., Tsang T., "Plans, actions and dialogues using linear logic", *Journal of Logic, Language and Information*, vol. 18, n° 2, p. 251-289, 2009.

Dobnik S., Cooper R., "Interfacing language, spatial perception and cognition in Type Theory with Records", *Journal of Language Modelling*, vol. 5, n° 2, p. 273-301, 2017.

Fernández R., Ginzburg J., Lappin S., "Classifying Ellipsis in Dialogue: A Machine Learning Approach", *Computational Linguistics*, vol. 33, n° 3, p. 397-427, 2007.

Ginzburg J., *The Interactive Stance*, Oxford University Press, 2012.

Ginzburg J., Fernández R., "Computational Models of Dialogue", *The Handbook of Computational Linguistics and Natural Language Processing*, vol. 57, p. 1, 2010.

Ginzburg J., Fernández R., Schlangen D., "Disfluencies as Intra-Utterance Dialogue Moves", *Semantics and Pragmatics*, vol. 7, n° 9, p. 1-64, 2014.

Girard J.-Y., *Linear Logic: its syntax and semantics*, London Mathematical Society Lecture Note Series, Cambridge University Press, p. 1–42, 1995.

Goodman N., Lassiter D., "Probabilistic Semantics and Pragmatics: Uncertainty in Language and Thought", *in* S. Lappin, C. Fox (eds), *The Handbook of Contemporary Semantic Theory, Second Edition*, Wiley-Blackwell, Malden, Oxford, p. 655-686, 2015.

Grover C., Brew C., Manandhar S., Moens M., "Priority Union and Generalization in Discourse Grammars", *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ACL '94, Association for Computational Linguistics, USA, p. 17–24, 1994.

Hough J., Purver M., "Processing Self-Repairs in an Incremental Type-Theoretic Dialogue System", *Proceedings of SemDial 2012 (SeineDial)*, p. 136-144, September, 2012.

Huang M., Zhu X., Gao J., "Challenges in building intelligent open-domain dialog systems", *ACM Transactions on Information Systems (TOIS)*, vol. 38, n° 3, p. 1-32, 2020.

Jokinen K., *Constructive dialogue modelling: Speech interaction and rational agents*, vol. 10, John Wiley & Sons, 2009.

Larsson S., Issue-based dialogue management, PhD thesis, University of Gothenburg, 2002.

Levinson S. C., *Pragmatics*, Cambridge University Press, Cambridge, U.K., 1983.

Lison P., "A hybrid approach to dialogue management based on probabilistic rules", *Computer Speech & Language*, vol. 34, n° 1, p. 232-255, 2015.

Lücking A., "Modeling Co-Verbal Gesture Perception in Type Theory with Records", *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems*, p. 383-392, 2016.

Łupkowski P., Ginzburg J., "Query responses", *Journal of Language Modelling*, vol. 4, n° 2, p. 245-292, 2017.

Maraev V., Ginzburg J., Larsson S., Tian Y., Bernardy J.-P., "Towards KoS/TTR-based proof-theoretic dialogue management", *Proceedings of the 22nd Workshop on the Semantics and Pragmatics of Dialogue*, SEMDIAL, Aix-en-Provence, France, November, 2018.

Martens C., Programming Interactive Worlds with Linear Logic, PhD thesis, Carnegie Mellon University Pittsburgh, PA, 2015.

Purver M., "CLARIE: Handling Clarification Requests in a Dialogue System", *Research on Language & Computation*, vol. 4, n° 2, p. 259-288, 2006.

Purver M. R. J., The theory and use of clarification requests in dialogue, PhD thesis, University of London, 2004.

Rieser V., Lemon O., *Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation*, Springer Science & Business Media, 2011.

Rodríguez K. J., Schlangen D., "Form, intonation and function of clarification requests in German task-oriented spoken dialogues", *Proceedings of Catalog (the 8th workshop on the semantics and pragmatics of dialogue; SemDial04)*, 2004.

Schegloff E. A., "Sequencing in conversational openings", *Directions in sociolinguistics*, 1972.

Schlangen D., A coherence-based approach to the interpretation of non-sentential utterances in dialogue, PhD thesis, University of Edinburgh. College of Science and Engineering, 2003.

Wiśniewski A., "Semantics of questions", *The Handbook of Contemporary Semantic Theory*, Wiley Online Library, p. 273-313, 2015.

Young S., Gašić M., Keizer S., Mairesse F., Schatzmann J., Thomson B., Yu K., "The hidden information state model: A practical framework for POMDP-based spoken dialogue management", *Computer Speech & Language*, vol. 24, n° 2, p. 150-174, 2010.