Non-humorous use of laughter in spoken dialogue systems

Vladislav Maraev[1*], **Jean-Philippe Bernardy** [1] and **Christine Howes**[1]
[1]Centre for Linguistic Theory and Studies in Probability (CLASP), Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg
{vladislav.maraev, jean-philippe.bernardy, christine.howes}@gu.se

Abstract

In this paper we argue that laughter, an ambiguous yet ubiquitous signal in everyday interactions, can act as an important feature for task-oriented dialogue systems. We show which components of a dialogue system should be affected and modified, and more specifically how particular types of laughter can be accounted for in a dialogue manager as instances of short answers, feedbacks and vocalisations accompanying them.

1 Introduction

Laughter is very frequent in everyday interactions, for instance, in the Switchboard Dialogue Act Corpus [Jurafsky *et al.*, 1997] corpus laughter comes about every 200 words. Laughter is an ambiguous social signal, and in addition to communicating joy and pleasure intuitively associated with humour it also can communicate embarrassment, be used to smooth and soften everyday interactions and also bear pragmatic functions such as marking irony or usage of a word in a specific sense [Poyatos, 1993; Mazzocconi, 2019; Ginzburg *et al.*, 2020].

For a spoken dialogue system, laughter is an important signal to account for due to its contribution to the naturalness of automated dialogue. Laughter can be used in chit-chat dialogue due to its potential to build rapport and establish a para-social bond between the user and artificial agent.

There have been attempts to produce laughs as a way to mimic human behaviour and align with it [Urbain *et al.*, 2010; El Haddad *et al.*, 2019], as well as laughing avatars mainly focussed on laughter as a reaction to jokes [Ochs and Pelachaud, 2013; Ding *et al.*, 2014]. In this paper we take a rather different approach. We start from examples of usage of laughter in real task-oriented dialogue and then propose ways how these behaviours can be reproduced in a dialogue system, and, more specifically, in its dialogue management component.

The example (1) below is an excerpt from a role-play dialogue collected by Howes *et al.* [2019] for their Directory Enquiries Corpus (DEC) [Bondarenko *et al.*, 2020]. Dialogue participants were playing the roles of a caller and an operator, respectively asking for the phone numbers of certain named businesses. Half of the dialogues happened in a noisy environment, with many mishearings and laughs induced. This paper addresses the following research question: how can these laughs be accounted for in a dialogue system, which implements a similar scenario?

(1) DEC:22_KL_loc2

56	Caller	*er the next one is er tanfield chambers*
57	Operator	*santias?*
58	Caller	*tanfield like t- T A N*
59	Operator	*sorry i don't hear you again please?*
60	Caller	*er T A N*
61	Operator	*C?*
62	Caller	*tanfield*
63	Operator	*A*
64	Operator	*N*
65	Caller	*yeah*
66	Caller	*and then field*
67	Operator	*and then seal?*
68	Caller	*chambers*
69	Operator	*<laugh> sorry i hear you quite poorly*
70	Operator	*let's try again*
71	Operator	*C?*
72	Caller	*yeah sorry the traffic is crazy around here*
73	Operator	*I know <laugh> don't worry*
74	Operator	*so C*
75	Operator	*A*
76	Caller	*er*
77	Caller	*tanfield T like thomas*

Let's look at the first laughter (line 69). We can see that the operator's question "and then seal?" (l.67) was not addressed and this piece of information was not grounded. "C?" (l.71) refers to the restart from the beginning (it was "Tanfield", but she has heard "C"). The negative feedback provided by the operator (l.69) entails extra effort from the caller—she needs to restart her request from the beginning—this obligation is somewhat intrusive and may require extra smoothing [Mazzocconi, 2019; Raclaw and Ford, 2017]. For our purposes, we will treat this laughter as accompanying negative feedback.

*Contact Author

For a dialogue system designer, this poses an empirical question, namely, would it be useful to soften negative feedback with laughter? For instance, the feedback associated with a local failure (e.g. speech recognition failure), such as "Sorry, I didn't understand" or "Sorry I didn't hear you". It may also be useful where negative feedback is the result of an external query, for example, when something is not found in the database, and can accompany a system request to start over, as in example (1).

The reaction to the apology also can be accompanied by laughter, as with the second laugh in (1) (l.73). We do not think that these days users often apologise to a dialogue system, as it is usually the dialogue system which is at fault, but this might be different for special cases of systems that aim at more naturalistic behaviour.

In this paper we consider laughter from the utilitarian perspective and attempt to determine which kinds of laughs can be relevant for dialogue systems. Next, we will look at laughter from the point of view of providing feedback, either positive or negative.

In Section 2 will start with a background on our approach to dialogue, dialogue management and laughter. Next, Section 3 presents a small typology of laughter types that we think should be accounted for in a task-oriented dialogue system. In Section 4 we describe our own dialogue management framework and in Section 5 we show a formal account for the aforementioned types of laughter. We conclude with a brief discussion of our findings and further laughter-related issues in Section 6.

2 Background

2.1 Dialogue

A key aspect of dialogue systems 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 [Allen *et al.*, 1995; Larsson, 2002; Jokinen, 2009] and statistical modelling of dialogue [Rieser and Lemon, 2011; Young *et al.*, 2010; Williams *et al.*, 2017]. 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. 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. In this paper we do so by employing our own implementation of the KoS theoretical dialogue framework [Ginzburg, 2012] which we discussed in [Maraev *et al.*, 2020]. In this work we extend our implementation with rudimentary support of grounding, therefore allowing the implementation to be further extended to support certain types of laughter.

In KoS (and many other dynamic approaches to meaning), language is treated as 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 least, shared assumptions/visual field, moves (= utterances, form and content), and questions under discussion.

In dialogue, especially in a dialogue with a machine which involves uncertainty of automatic speech recognition (ASR) and natural language understanding components (NLU), we can not assume perfect communication. While communicating, especially over an unreliable communication channel, humans give each other evidence that their contributions are understood to a certain extent, sufficient for current purposes. Clark [1996] and Allwood [1995] distinguish four *levels of action* related to different degrees of grounding. Here we list them according to the *action ladder* [Clark, 1996], from the hearer's perspective.

1. **Acceptance** level determines whether the content of utterance was accepted or rejected by the hearer.
2. **Understanding** level specifies whether the utterance was understood by the hearer
3. **Perception** level determines whether the utterance was perceived by the hearer.
4. **Contact** level determines whether interlocutors have established a channel of communication.

The action ladder assumes that if the level above is complete, then all levels below are complete. For instance, if Bob asks "Do you like Paris" and Mary replies "Yes", then Bob's utterance is accepted (and also understood, perceived, and their contact has been established). If she asks "Paris?" then it might signal that Bob's utterance was perceived but not understood (and thus not accepted).

Larsson [2002] accounts for different levels of action within the IBiS2 dialogue management framework using a set of rules to update the common ground represented in the information state of the system. He uses "Interactive Communication Management" (ICM) moves [Allwood, 1995] as explicit signals concerned with communicating the updates to the common ground, and sequencing moves, e.g. restarting a dialogue.

2.2 Laughter

Our focus of attention towards laughter is motivated by its ubiquity in natural dialogue. In the British National Corpus, laughter is quite a frequent signal regardless of gender and age—the spoken dialogue part of the British National Corpus (UK English, unscripted interactions that were recorded by volunteers in various social settings, balanced for age, region and social class) contains approximately one occurrence of laughter every 14 utterances. In the Switchboard Dialogue Act corpus [Jurafsky *et al.*, 1997] (US English, one-on-one

interactions over a phone where participants that are not familiar with each other discuss a potentially controversial subject, such as gun control or school system) non-verbally vocalised dialogue acts (whole utterances that are marked as non-verbal) constitute 1.7% of all dialogue acts and 65% of them contain laughter. Laughter tokens make up 0.5% of all the tokens that occur in Switchboard Dialogue Act corpus.

Laughter production in conversation is not exclusively related to humour. But, perhaps unsurprisingly, the study of laughter has often been linked to the study of humour and the two terms are frequently used interchangeably. However, laughter does not occur only in response to humour or in order to frame it. Many studies, particularly in conversation analysis, have shown its crucial role in managing conversations at several levels: dynamics (turn-taking and topic-change), lexical (signalling problems of lexical retrieval or imprecision in the lexical choice), pragmatic (marking irony, disambiguating meaning, managing self-correction) and social (smoothing and softening difficult situations or showing (dis)affiliation) [Glenn, 2003; Jefferson, 1984; Mazzocconi, 2019; Petitjean and González-Martínez, 2015]

There have been several approaches to classify types of laughter [e.g., Poyatos, 1993; Vettin and Todt, 2004; Mazzocconi, 2019]. Mazzocconi [2019] claims that the most problematic issue with existing taxonomies is that they mix types of laughter functions with types of laughter triggers, so she roots her proposal on the function of laughter and the propositional content of *laughable*—the argument the laughter predicates about, an event or state referred to by an utterance or exophorically [Glenn, 2003]. In this paper we look at laughter not exclusively from a perspective of a taxonomy that can be used as a theoretical framework but from the utilitarian perspective, looking at which kinds of laughs can be relevant for dialogue systems.

Laughter as a way for an embodied conversational agent (ECA) to provide emotional response has gained some attention from the Affective Computing and other research communities. Becker-Asano and Ishiguro [2009] evaluated the role of laughter in the perception of social robots and indicated that the situational context, determined by linguistic and non-verbal cues (such as gaze) played an important role. Nijholt [2002] discusses the challenges of integrating humour into ECAs, and existing integration of smiling and laughter in embodied conversational agents (ECA) is typically is triggered by a joke told by a user or an agent [Ding *et al.*, 2014; Ochs and Pelachaud, 2013]. El Haddad *et al.* [2019] looked at the mimicry of smiles and laughs between the interlocutors, which also might be used as the basis for ECA's behaviour. Urbain *et al.* [2010] takes a similar perspective, equipping ECAs with a capability to join its conversational partner's laugh. In this work we take a contrasting approach, looking at pragmatic functions of some types of laughter, namely providing feedback and answering questions, and provide a formal account for such behaviour within a dialogue management framework.

3 Types of laughter

In this section we outline some types of laughter that can be of special interest to task-oriented dialogue systems and can be accounted for within our proposed framework.

3.1 Laughter as a component of grounding

As we have mentioned in Section 2, and in accord with Allwood [1995]; Clark [1996]; Larsson [2002] we consider four action levels that are involved in a dialogue. Here we discuss what can happen at each level of action—contact, perception, understanding and reaction—with respect to laughter.

Contact and perception levels

Troubles related to establishing and maintaining a stable communication channel can lead to laughter. One such example would be delays in communication, for instance over an unreliable network, which might lead to a person already speaking at the moment when the communication is only supposed to be established. Obvious examples of such cases are caused by signal jitter over video conference platforms like Zoom.

The lack of perception indicates things that haven't been heard correctly (cases similar to (1)). Also, it seems that interruptions or events related to that can be quite surprising and laughter can be a natural reaction to a surprise (see Section 6).

Understanding level

The lack of pragmatic understanding relates to the kinds of incongruities that are caused by the violation of the principle of conversational relevance. This is very useful for dialogue systems because they are prone to errors in this realm. It is often the case that incorrect NLU or ASR can lead to prioritising irrelevant results (for example, in cases of out-of-scope user queries), which can cause user's confusion and, therefore, laughter. This type of laughter can be treated as negative feedback.

This accounts for the examples (2) and (3) below. [Larsson, 2002] subdivides this level into three categories for the negative feedback (context-dependent, context-independent and pragmatic). The examples (2) and (3) above would relate to the pragmatic level of misunderstanding.

(2) from the dialogue between a virtual assistant (Diana) and a person with ASD (Mark):

Mark	*Diana, what is money?*
Diana	*I am Diana, a virtual interlocutor.*
Audience	*(laugh)*

(3) constructed example

Brian	*Would you like tea or coffee?*
Katie	*yes*
Brian	*(laughs)*

A dialogue system can also be unsure about what has been understood. In such cases, the system should demonstrate a lower degree of commitment to what has been said as a part of a display of understanding. For example, in the case of the feedback regarding the user input, when the system repeats the input after the user, it can be useful to include laughter in verbatim repeats, which would mean: yes, I heard (understood) this, but I might be wrong. This can also be useful for a system's actions taken based on low confidence results.

Reaction (consider for acceptance) level

On this level what has being understood can be either accepted or rejected for the current purpose. Acceptance laughter can typically be related to a reaction to humour, which is out of the scope of the current paper, or apology (see next section).

Ginzburg *et al.* [2020] consider some uses of standalone laughter as cases of negative response to a polar question (4) or a signal of disbelief in a previously uttered assertion (5).

(4) From Ginzburg *et al.* [2020], context: Bayern München goalkeeper Manuel Neuer faces the press after his team's (*Dreierkette*—three-in-the-back) defence has proved highly problematic in the game just played (which they won 3-2 against Paderborn).

Journalist: (smile)	*Dreierkette auch 'ne Option? (Is the three-at-the-back also an option?)*
Manuel Neuer:	*fuh fuh fuh (brief laugh)*

(5) From Ginzburg *et al.* [2020] (biblical example rephrased as a dialogue)

God:	*You will at age 99 with your aged wife Sarah have a son.*
Abraham:	*(laughs)*

→ I don't think I will at age 99 have a son

In Section 5 we show how this kind of laughter as negative response like (4) can be handled by the dialogue manager.

3.2 Laughter and intrusion

In natural dialogue, an intrusion is frequently associated with laughter. In the Switchboard Dialogue Act corpus (SWDA) [Jurafsky *et al.*, 1997] an Apology dialogue act is more related to laughter, as compared to other dialogue acts. In Figure 1 we show how many dialogue acts are associated with utterances[1] containing laughter, for the current dialogue act and for preceding and following utterances, depending on the speaker. In addition to an apology, we show its adjacency counterpart (second element of the utterance pair produced by the other speaker [Schegloff and Sacks, 1973])—Downplayer—realised, for instance, by utterances like "Don't worry" or "It's alright".

In (6), the caller reacts with compassionate laughter to the apology given by the operator. This is a similar instance of laughter to one seen in (1): the second laugh shows that the same reaction, as in (6) can be expected from the operator.

(6) DEC:16_HG_loc2

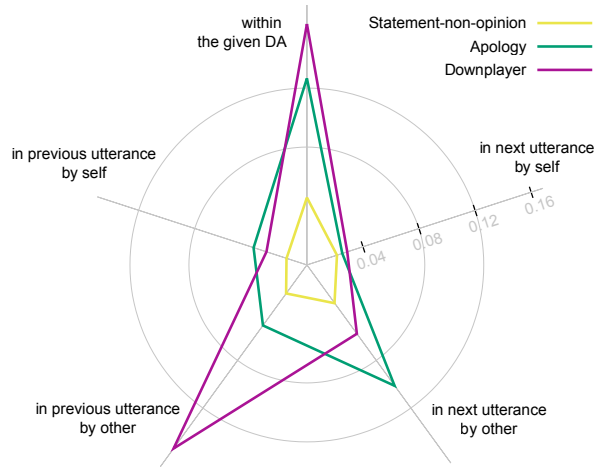

Figure 1: Comparison of the most common dialogue act in SWDA—"Statement-Non-Opinion" (33.27% of all utterances) with the dialogue acts "Apology" (0.04%) and "Downplayer" (0.05%). The proportion of utterances that contain laughter are shown in association with each dialogue act.

162	Operator	*still not finding it*
163	Operator	*having problems with this one*
164	Caller	*okay*
165	Caller	*er maybe i can find*
166	Caller	*er the place myself but thank you very much for the information*
167	Operator	*no problem sorry for not finding the the last one*
168	Caller	*<laugh>*
169	Caller	*no worries*
170	Caller	*thank you*

We also observe that laughter can clearly accompany the asking for a favour by the same speaker. In example (7) the operator asks the caller if they can start from the beginning, which can be treated as an intrusion of some sort, therefore asking for a favour and the apology is accompanied by laughter.

(7) DEC:24_LK_loc2

59	Caller	*B as in bicycle*
60	Operator	*yeah*
61	Caller	*then you have R*
62	Caller	*I*
63	Operator	*R*
64	Caller	*G*
65	Operator	*I*
66	Operator	*okay sorry no- now i lost the track okay can we it start from the beginning <laugh> sorry*
67	Caller	*okay*
68	Caller	*yes we can*
69	Operator	*maybe you can just say the uh say words*
70	Caller	*yeah no no problem*

[1] In SWDA each utterance is typically mapped to a single dialogue act.

4 Dialogue manager architecture

We believe that it is crucial to use formal tools which are most appropriate for the task: one should be able to express the rules of various genres of dialogue in a concise way, free, to any possible extent, of irrelevant technical details. 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. Very often, dialogue-management rules update subsets (propositions) of the information state independently from the rest. A suitable and flexible way to represent such updates is as function types in linear logic. The domain of the function is the subset of propositions to update, and the co-domain is the (new) set of propositions which it replaces.

By using well-known techniques which correspond well with the intuition of information-state based dialogue management, we are able to provide a fully working prototype of the components of our framework:

1. a 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])
3. several examples which use the above to construct potential applications of the system.

4.1 Linear rules and proof search

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 using linear logic [Girard, 1995] instead. For our purpose, the crucial feature of linear logic is that hypotheses may be used *only once*.

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.

In linear logic, normally 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. However, 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.) Scenarios can be realised as a sequence of actual actions and updates. 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 extra-logical updates of the information state (through a model of the perceptory subsystem).

In our implementation, we treat the information state as a multiset of *linear hypotheses* that can be queried. Because they are linear, these hypotheses can also be removed from the state. In particular, we have a fixed set of rules (they remain available even after being used). Each such rule manipulates a part of the information state (captured by its premisses) and leaves everything else in the state alone.

Our dialogue manager (DM) models the information-state of only one participant. Regardless, this participant can record its own beliefs about the state of other participants. In general, the core of the 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 examples of such rules in Section 4.4.

4.2 Questions and answers

In this paper, the essential components of the representation of a question are a type A, and a predicate P over A. Using a typed intuitionistic logic, we write:

$$A : \mathit{Type} \qquad\qquad P : A \to \mathit{Prop}$$

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 qualifying the predicate in question, for example, "maybe", "on Tuesdays", etc. (Table 1, last two rows). This is formalised by letting $A = \mathit{Prop} \to \mathit{Prop}$.

4.3 Representation of questions 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 to a question.

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:

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

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

$$\mathit{processShort} : (a : \mathit{Type}) \to (x : a) \to (p : \mathit{Prop}) \to \mathit{ShortAnswer}\ a\ x \multimap \mathit{QUD}\ (Q\ a\ x\ p) \multimap p$$

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.

Above we use Π type binders to declare (meta)variables (written here $(a : Type) \to$, $(x : a) \to$, etc.). This terminology will make sense to readers familiar with dependent types. For others, such binders can be thought of as universal quantification ($\forall a, \forall x$, etc.), the difference is that the type of the bound variable is specified.[2]

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:

$$processAssert : (a : Type) \to (x : a) \to (p : Prop) \to Assert\ p \multimap QUD\ (Q\ a\ x\ p) \multimap p$$

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 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 of the 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.

4.4 Dialogue management

In this section we integrate our question/answering framework within more complete dialogue manager (DM). We stress that this DM models the information-state of only one participant. Regardless, this participant can record its own beliefs about the state of other participants. In general, the core of the 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 the generic processing of answers).

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:

$Greet$	$spkr\ addr$
$CounterGreet$	$spkr\ addr$
Ask	$question\ spkr\ addr$
$ShortAnswer$	$vtype\ v\ spkr\ addr$
$Assert$	$p\ spkr\ addr$

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 $Heard : Move \to Prop$ 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 counter greeting, a rule will place the proposition $(CounterGreet\ \ A\ \ S)$ in the $Cons$-list $Agenda$ part of 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 recorded 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

[2]The reader worried about any theoretical difficulty regarding mixing linear and dependent types is directed to Atkey [2018] and Abel and Bernardy [2020].

actively react to, in a 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.

Here we will provide a few examples of the rules which are implemented in our system, and we refer our reader to [Maraev *et al.*, 2020] for more detailed description.

Examples

We can show how basic move-adjacency can be defined in the example of a counter greeting preconditioned by a greeting from the other party:[3]

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

Another important rule accounts for pushing the content of any received Ask move on top of the stack of questions under discussion (QUD).

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

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 $PendingUserFact$.[4] The following rule[5] is an extended dialogue management version of the rule previously introduced in Section 4.3.

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

Then, other rules will take into account the $PendingUserFact\ \ p$ in a system-specific way. In the simplest case, the system may treat p as a true proposition. (In this paper we will consider meta-level pending user facts instead.)

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

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

If the system has a fact p in its database it can produce an answer or a domain-specific clarification request depending on whether the fact is unique and concrete or not (defined by operators $\to_!$ and $\to_?$ respectively, see Maraev *et al.*, 2020 for further details).

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

4.5 Extending the dialogue manager with grounding strategies

In this subsection we provide a sketch of basic grounding strategies and moves related to them, which will be further used to model laughter.

Dialogue systems deal with confidence scores from ASR and NLU components, which reflects the uncertainty in user queries. For simplicity we will represent the confidence score t in on the basis of three confidence threshold levels ($T_1 < T_2$), where RED would correspond to $t < T_1$, $YELLOW$ to $T_1 < t < T_2$, and $GREEN$ to $T_2 < t$. Colour-coded confidence scores would accompany user moves, e.g. the Ask move such as "What time is it?" can be represented as follows:

$$Ask\ (Q\ U\ Time\ t0\ (IsTime\ t0))\ U\ S\ YELLOW$$

Here we illustrate the possibility of extending the system with Interactive Communication Management (ICM) moves and grounding strategies, replicating Larsson's [2002] account for grounding and feedback. ICM moves are used for coordination of the common ground in dialogue, which expresses, for instance, explicit signals for integrating the incoming information and updating the common ground (dialogue gameboard in our implementation). The basic type for the ICM move is the following:

$$ICM\ level\ polarity\ content$$

where $level$ corresponds to the level of grounding (contact, perception, understanding, acceptance), $polarity$ is either positive or negative, and the optional value $content$ corresponds to a component of the common ground in question. For instance, the move $(ICM\ Per\ Neg\ None)$ would correspond to the utterance "I didn't understand what you said" or "Pardon", and the move $(ICM\ Und\ Pos\ q)$ can be realised as the utterance "You are asking me what time is it" if the QUD q corresponds to the question from Ask move exemplified above.

Next, we modify our basic $pushQUD$ rule defined in Section 4.4 to support different system behaviours depending on the confidence score. In the $GREEN$ case, question from the user Ask move is being integrated into QUD, and ICM move displaying positive acceptance feedback, i.e. "okay", $(ICM\ Acc\ Pos\ None)$ is being put on the $Agenda$. In the $YELLOW$ case, system should additionally report about positive understanding, e.g. "You want to know about time", so it adds $(ICM\ Und\ Pos\ q)$ move on the $Agenda$.

[3] Taking a linear argument and producing it again is a common pattern, which can be spelled out $A \multimap (A \otimes P)$. From here on we use the syntactic sugar $A \multimap P$ for it.

[4] 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.

[5] Note the use of the single colon (:) for metavariables and the double colon for information-state hypotheses (::).

$$
\begin{array}{l}
pushQUDGreen : (q : Question) \rightarrow \\
\quad (qs : List\ Question) \rightarrow (x\ y : DP) \rightarrow \\
\quad Pending\ (Ask\ q\ x\ y\ GREEN) \multimap Agenda\ as \multimap \\
\quad QUD\ qs \multimap \\
\quad\quad [\_ :: QUD\ (Cons\ q\ qs); \\
\quad\quad\ \_ :: Agenda\ (Cons\ (ICM\ Acc\ Pos\ None)\ as);\ ]
\end{array}
$$

$$
\begin{array}{l}
pushQUDYellow : (q : Question) \rightarrow \\
\quad (qs : List\ Question) \rightarrow (x\ y : DP) \rightarrow \\
\quad Pending\ (Ask\ q\ x\ y\ YELLOW) \multimap Agenda\ as \multimap \\
\quad QUD\ qs \multimap \\
\quad\quad [\_ :: QUD\ (Cons\ q\ qs); \\
\quad\quad\ \_ :: Agenda\ (Cons\ (ICM\ Acc\ Pos\ None) \\
\quad\quad\ (Cons\ (ICM\ Und\ Pos\ q)\ as));\ ]
\end{array}
$$

For RED confidence score, the system issues an interrogative ICM query, such as "I understood you're asking me about the time, is that correct?". In this case a special type of QUD is introduced, namely a question about whether question q is correctly understood.

$$
\begin{array}{l}
icmINTConfirm : (q : Question) \rightarrow (x\ y : DP) \rightarrow \\
\quad Pending\ (Ask\ q\ x\ y\ RED) \multimap Agenda\ as \multimap \\
\quad QUD\ qs \multimap \\
\quad [\_ :: QUD\ (Cons\ (Q\ Bool\ x \\
\quad\quad (\textbf{if}\ x\ \textbf{then}\ UND\ q \\
\quad\quad\ \textbf{else}\ UNDN\ q))\ qs); \\
\quad\ \_ :: Agenda\ (Cons\ (ICM\ Und\ Int\ q)\ as)]
\end{array}
$$

Processing answers related to such a type of QUD will be done as usual. For instance, a short "yes" or "no" will be treated here as a boolean, and depending on the answer the context will contain either $PendingUserFact\ (UND\ q)$ or $PendingUserFact\ (UNDN\ q)$.

In this sketch implementation, we do not care about confidence scores for these answers, leaving it underspecified, but further, more specific dialogue rules are possible.

Regardless of the particular answer, once the ICM question is answered, it is removed from the QUD stack, so that to of the QUD stack is restored to the originally asked question. In our system, this is taken care of by the generic handling of $ShortAnswer$s. Thus, in the case of a positive answer to such a query, there is nothing particular to do.

In the negative case, the ICM move about the understanding that the question was not q is issued.

$$
\begin{array}{l}
icmINTneg : (q : Question) \rightarrow (x\ y : DP) \rightarrow \\
\quad (c : Confidence) \rightarrow \\
\quad PendingUserFact\ (UNDN\ q) \multimap \\
\quad Agenda\ as \multimap \\
\quad Agenda\ (Cons \\
\quad\quad (ICM\ Und\ Pos\ (QuestionIsNot\ q))\ as)
\end{array}
$$

How ICM moves are converted to natural language utterances, depending on q, is a natural language generation (NLG) issue. For instance,

$$
\begin{array}{l}
ICM\ Und\ Pos \\
\quad (QuestionIsNot \\
\quad\quad (Q\ U\ Time\ t0\ (IsTime\ t0)))
\end{array}
$$

can become the (rather tedious) utterance "So, you are not asking me what time it is", whereas more sophisticated queries with more arguments can be resolved in shorter utterance depending on the arguments that are made ground. For instance, in a context of interaction at a food kiosk:

$$
\begin{array}{l}
ICM\ Und\ Pos \\
\quad (QuestionIsNot \\
\quad\quad (Q\ U\ (Prop \rightarrow Prop)\ m0\ (m0\ WantOlives))
\end{array}
$$

could become a simple "Sorry, let's forget olives.".

5 Formal treatment of certain types of laughter

5.1 Laughter as a rejection signal

Laughter as a reaction to interrogative feedback in the case of low confidence ASR/NLU result can be illustrated by the following dialogue.

(8)

U:	*I would like to order a vegan bean burger.*	Ask q
S:	*I understood you'd like to order a beef burger. Is that correct?*	ICM Und Int q
U:	*HAHAHA*	ShortAnswer Bool False

Here we can treat laughter as a short negative answer, similar to "No". In the case of interrogative ICM move, such an answer can be processed using the $icmINTneg$ rule defined above.

This can be treated as a recovery strategy for different system outputs not desired by dialogue system designers. This approach can be extended to other cases of user feedback, for instance, to cover the cases with higher confidence score where the system produces $ICM\ Und\ Pos\ q$ move, but this is out of the scope of the current paper.

Returning to the more sophisticated (4), it can be handled by our generic rules for integrating QUDs ($pushQUD$). For that we need to consider polar questions as expecting an answer of $Prop \rightarrow Prop$ type (see Table 1). Recalling the example:

(4)

Journalist: (smile)	*Dreierkette auch 'ne Option? (Is the three-in-the-back also an option?)*
Manuel Neuer:	*fuh fuh fuh (brief laugh)*

and a type for question:

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

In this case,

$$
\begin{array}{l}
A = Prop \rightarrow Prop \\
P = \lambda m.m\ IsOptionDreierkette
\end{array}
$$

The brief laughter by Manuel Neuer can be represented as:

$$
\begin{array}{l}
[\![\ fuhfuhfuh\ ]\!] = ShortAnswer \\
\quad (Prop \rightarrow Prop)\ (\lambda x.Laughable\ x)
\end{array}
$$

where the modification of the proposition, resulting in $(Laughable\ \ IsOptionDreierkette)$ has a very basic meaning: this proposition is the *laughable*, without being more

specific about the laughter function. One can also consider being more specific, simply treating laughter as a negation ($ShortAnswer\ (Prop \rightarrow Prop)\ (\lambda x.Not\ x)$), but in general laughter has a more nuanced meaning.

5.2 Laughter which accompanies feedback

Laughter can act as a part of ICM moves' realisation performed by natural language generation (NLG) component. It seems to us that, in particular, ICM moves the use of laughter can be considered "safe". For instance, ICM move of the form ($ICM\ Und\ Pos\ (QuestionIsNot\ (Q\ U\ (Prop \rightarrow Prop)\ m0\ (m0\ WantOlives)))$) can be realised as a natural language utterance like "Okay, let's forget olives, hehe", whereas laughter is used as a smoothing device to mitigate the awkwardness of system failure. Larsson [2002] often included an apology "Sorry" in some of the ICM moves, e.g. "Sorry, I didn't understand that". With some possible caveats, we can sometimes include slight laughter in such moves, especially if a system is getting a bit repetitive and produces ($ICM\ Und\ Neg$) too often. Considering the evidence for laughter often accompanying apology (as a separate dialogue act) presented in Section 3.2, this can mimic natural behaviour in dialogue.

6 Discussion and future work

In this paper we have shown how some types of laughter can be accounted for in task-oriented spoken dialogue system. We proposed our own proof-theoretic architecture of a dialogue manager based on KoS framework and extended it with some grounding strategies. Based on this, we have shown how certain types of laughter, can be processed within the dialogue manager and natural language generator, namely: laughter as a negative feedback, laughter as a negative answer to a polar question and laughter as a signal accompanying system feedback.

In the following subsections we discuss several issues related to laughter in spoken dialogue systems, but only merely touching the main subject of the paper.

6.1 Humour

We start with humour, which is usually considered in relation to jokes generated by dialogue system, but here we present more subtle incongruities related to humour in task-oriented dialogue.

(9) DEC:28_NM_loc2

17	Caller	*okay so it starts with a*
18	Caller	*L*
19	Operator	*L?*
20	Caller	*as in london*
21	Operator	*yes*
22	Caller	*A as in america*
23	Operator	*america*
24	Caller	*er U*
25	Caller	*as in er ((pause: 1.2s))*
26	Caller	*er under*
27	Caller	*<laugh>*
28	Operator	*under yes*

In (9) the caller experiences issues with coming up with phonetic spellings for certain words. The first laugh (line 27) deserves attention, as it seems that it reflects on both pleasant incongruity and social one (smoothing), according to the taxonomy of [Mazzocconi, 2019]. The pleasant incongruity is due to the fact that the phonetic spelling of "U" as in "under" is incongruous with the preceding ones: a preposition vs. proper nouns. The way to spell things phonetically is typically culturally specific, with the most typical cases of cities or countries. Stereotypes and conversational conventions can be expressed with the formal notions of *enthymemes* and *topoi*, following the work of Breitholtz [2020] on reasoning in conversation. Breitholtz and Maraev [2019] used these notions to analyse conversational humour as well as canned jokes, and we find it potentially helpful to be integrated into our framework in order to account for humour in dialogue systems. Dybala *et al.* [2010] emphasises the importance of the "two-stage" approach to humour in dialogue systems, where the system tracks the emotional state of the user, produces humour as a reaction to certain states and analyses user's further emotional reaction.

6.2 Surprise

Intuitively, laughter is related to events that are not expected in interaction. One of the ways to establish some degree of natural behaviour for a dialogue system would be to react sincerely to these kinds of surprising events. A possible measure for a system's surprisal is how confused it is by the user input. A natural measure for this from information theory is *perplexity*, a probability-based metric. For N words in an evaluation set $W = w_1 w_2 \dots w_N$, the average perplexity per word is computed as follows:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_1 \dots w_{i-1})}} \qquad (1)$$

Given a language model, we can employ a threshold defined by perplexity which the system can use to act as being surprised, e.g. by saying "Ha-ha, I did not expect this!"

Similarly, perplexity can be inferred from tracking a dialogue state in a Dialogue State Tracking task [Mrkšić *et al.*, 2017], which is a common task in statistical approaches to dialogue system. Or, following Noble and Maraev [2021], the RNN trained on a large dialogue corpus as a representation of dialogue context can be used to calculate perplexity.

Laughter as a reaction of surprise can relate to the levels of feedback, for example, a user surprised by a pragmatically incoherent system's reply can laugh (Section 5.1). But here surprise is taken in isolation, as a measure on its own right.

6.3 Awkwardness and time-saving

In (9), "under" is produced after a long pause (l.25) and therefore indicates awkwardness in producing the phonetic spelling made the operator wait—therefore making the situation uncomfortable to the caller, so laughter was used for smoothing it.

In the follow-up excerpt (10) from the same dialogue, user's awkwardness continues and she accompanies it with

laughter. Firstly, she laughs (l.139) demonstrating that she has given up finding any phonetic spelling for "K", releasing the turn and allowing the operator to carry on. Her second laugh smooths her slight embarrassment after the situation was resolved by the operator.

(10) DEC:28_NM_loc2

134	Caller	*O for oslo*
135	Operator	*O for oslo*
136	Caller	*again O for oslo*
137	Operator	*O for oslo*
138	Caller	*and K for er ((pause: 1.6s))*
139	Caller	*<laugh>*
140	Operator	*as in king?*
141	Caller	*k- king <laugh> yeah*
142	Operator	*yes*
143	Caller	*thank you*
144	Operator	*that's it?*
145	Caller	*that's it*

We can hypothesise that in a dialogue system these examples can be handled as follows. For a system, there are operations which the developer knows are going to take time due to technical constraints, but are expected to be immediate by the user. In this case, a system can produce a similar behaviour to the one in (9) (l.25–27): "er… (pause) [comes up with an answer] <laugh>". A system can detect the patterns of filled pause + <laugh> from the user and treat them as turn-release cues. It can be a signal of either that there is something that confused the user, or that she genuinely could not come up with an answer due to certain difficulties. The downplayer dialogue act (e.g. "don't worry") or laughter in response also can be appropriate as system feedback in such a situation. We consider these ideas as a subject for further empirical investigations.

Laughter related to smoothing retrieval difficulties can be indicative. Consider the case of language tutoring. In the Anki "flashcard" app, the system provides users with a word in one language on the front side of the card and the user should provide a translation. The user then gets the correct response from the back of the card and evaluates her own response (was this card Hard, Good or Easy to recall). If we consider making a similar conversational app, indications of retrieval issues—filled pauses ("er em…") and follow-up smoothing by laughter—can lead to the decision to flag this card as "Hard" and provide corresponding feedback (11).

(11)

S	What is the Swedish for donkey?
U	er em … åsna?.. <laugh>
S	Yes, that was tough, but it is correct! (system marks the card as "Hard")

6.4 Approaches to evaluation

Each of the aforementioned improvements has to be a subject for evaluation within the dialogue system. We expect to see that these improvements will be reflected in the following evaluation criteria.

Some of the improvements would fall into an objective checklist-style criteria, like being able to understand laughter as negative feedback, or as a signal of surprise. The same goes for system's laughter as an appropriate reaction to conversational humour.

Another portion of the features can be evaluated only subjectively, for example, it is a question of user preference whether it is okay for a system to accompany asking for a favour (e.g. "Let's start over!") with laughter. For this purpose, we can employ subjective evaluation methods such as more task-oriented SASSI [Hone and Graham, 2000] or the more chatterbot-oriented methodology proposed by Dybala *et al.* [2009], which was used for humour-equipped chatbots. We optimistically expect that characteristics such as naturality and likeability would increase and annoyance would decrease.

Acknowledgments

The research reported in this paper was supported by grant 2014-39 from the Swedish Research Council, which funds the Centre for Linguistic Theory and Studies in Probability (CLASP) in the Department of Philosophy, Linguistics, and Theory of Science at the University of Gothenburg. In addition, we would like to thank Staffan Larsson, Jonathan Ginzburg and our anonymous reviewers for their useful comments.

References

Andreas Abel and Jean-Philippe Bernardy. A unified view of modalities in type systems. *Proceedings of the ACM on Programming Languages*, 4(ICFP), 2020.

James F Allen, Lenhart K Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang, Tsuneaki Kato, Marc Light, Nathaniel Martin, Bradford Miller, Massimo Poesio, et al. The TRAINS project: A case study in building a conversational planning agent. *Journal of Experimental & Theoretical Artificial Intelligence*, 7(1):7–48, 1995.

Jens Allwood. An activity based approach to pragmatics. 1995.

Robert Atkey. Syntax and semantics of quantitative type theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK*, pages 56–65, 2018.

Christian Becker-Asano and Hiroshi Ishiguro. Laughter in social robotics-no laughing matter. In *Intl. Workshop on Social Intelligence Design*, pages 287–300. Citeseer, 2009.

Anastasia Bondarenko, Christine Howes, and Staffan Larsson. Directory enquiries corpus, Feb 2020.

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

Ellen Breitholtz and Vladislav Maraev. How to put an elephant in the title: Modeling humorous incongruity with topoi. In *Proceedings of the 23rd Workshop on the Semantics and Pragmatics of Dialogue - Full Papers*, London, United Kingdom, September 2019. SEMDIAL.

Ellen Breitholtz. *Enthymemes and Topoi in Dialogue: The Use of Common Sense Reasoning in Conversation*. Brill, Leiden, The Netherlands, 2020.

Herbert H Clark. *Using language*. Cambridge university press, 1996.

Yu Ding, Ken Prepin, Jing Huang, Catherine Pelachaud, and Thierry Artières. Laughter animation synthesis. In *Proc. AAMS 2014*, pages 773–780. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

Lucas Dixon, Alan Smaill, and Tracy Tsang. Plans, actions and dialogues using linear logic. *Journal of Logic, Language and Information*, 18(2):251–289, 2009.

Pawel Dybala, Michal Ptaszynski, Rafal Rzepka, and Kenji Araki. Subjective, but ot worthless-on-linguistic features of chatterbot evaluations. In *6th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, page 87. Citeseer, 2009.

Pawel Dybala, Michal Ptaszynski, Rafal Rzepka, and Kenji Araki. Extending the chain: humor and emotions in human computer interaction. *International Journal of Computational Linguistics Research*, 1(3):116–125, 2010.

Kevin El Haddad, Sandeep Nallan Chakravarthula, and James Kennedy. Smile and laugh dynamics in naturalistic dyadic interactions: Intensity levels, sequences and roles. In *2019 International Conference on Multimodal Interaction*, pages 259–263, 2019.

Jonathan Ginzburg, Chiara Mazzocconi, and Ye Tian. Laughter as language. *Glossa: a journal of general linguistics*, 5(1), 2020.

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

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

Phillip Glenn. *Laughter in Interaction*. Cambridge University Press, Cambridge, UK, 2003.

Kate S Hone and Robert Graham. Towards a tool for the subjective assessment of speech system interfaces (sassi). 2000.

Christine Howes, Anastasia Bondarenko, and Staffan Larsson. Good call! Grounding in a Directory Enquiries Corpus. In *Proceedings of the 23rd Workshop on the Semantics and Pragmatics of Dialogue*, London, United Kingdom, sep 2019. SEMDIAL.

Gail Jefferson. On the organization of laughter in talk about troubles. In *Structures of Social Action: Studies in Conversation Analysis*, pages 346–369. 1984.

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

D Jurafsky, E Shriberg, and D Biasca. Switchboard dialog act corpus. *International Computer Science Inst. Berkeley CA, Tech. Rep*, 1997.

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

Vladislav Maraev, Jean-Philippe Bernardy, and Jonathan Ginzburg. Dialogue management with linear logic: the role of metavariables in questions and clarifications. *Traitement Automatique des Langues (TAL)*, 61(3):43–67, 2020.

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

Chiara Mazzocconi. *Laughter in interaction: semantics, pragmatics and child development*. PhD thesis, Université de Paris, 2019.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, 2017.

Anton Nijholt. Embodied agents: A new impetus to humor research. In *The April Fools Day Workshop on Computational Humour*, volume 20, pages 101–111. In: Proc. Twente Workshop on Language Technology, 2002.

Bill Noble and Vladislav Maraev. Large-scale text pretraining helps with dialogue act recognition, but not without fine-tuning. In *Proceedings of the 14th International Conference on Computational Semantics - Short Papers*, Groningen, Netherlands, 2021.

Magalie Ochs and Catherine Pelachaud. Socially aware virtual characters: The social signal of smiles [social sciences]. *IEEE Signal Processing Magazine*, 30(2):128–132, Mar 2013.

Cécile Petitjean and Esther González-Martínez. Laughing and smiling to manage trouble in french-language classroom interaction. *Classroom Discourse*, 6(2):89–106, 2015.

Fernando Poyatos. *Paralanguage: A linguistic and interdisciplinary approach to interactive speech and sounds*, volume 92. John Benjamins Publishing, 1993.

Joshua Raclaw and Cecilia E Ford. Laughter and the management of divergent positions in peer review interactions. *Journal of Pragmatics*, 113:1–15, 2017.

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

Emanuel A Schegloff and Harvey Sacks. Opening up closings. *Semiotica*, 8(4):289–327, 1973.

Jérôme Urbain, Radoslaw Niewiadomski, Elisabetta Bevacqua, Thierry Dutoit, Alexis Moinet, Catherine Pelachaud, Benjamin Picart, Joëlle Tilmanne, and Johannes Wagner. Avlaughtercycle. *J. Multimodal User Interfaces*, 4(1):47–58, 2010.

Julia Vettin and Dietmar Todt. Laughter in conversation: Features of occurrence and acoustic structure. *Journal of Nonverbal Behavior*, 28(2):93–115, 2004.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*, 2017.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.