

Specifying Generation of Referring Expressions by Example

Matthew Stone

Computer Science and Cognitive Science
Rutgers University
Piscataway NJ 08854-8019
mdstone@cs.rutgers.edu

Abstract

A module for generation of referring expressions (GRE) derives descriptions that identify specified entities in context. In the implemented system I describe here for specifying simple cases of GRE by example, system-builders pair entities with descriptions of them that would be satisfactory for a system to use in context. Automatic methods then construct a suitable knowledge base and context set for a knowledge-based GRE module for the system. These resources will always account for the sample descriptions the designer has supplied, but can also generalize to other possible referring expressions in other possible contexts in the application. I discuss the results in the perspective of knowledge-acquisition methodology for NLG for dialogue, draw contrasts with other uses of examples in NL technology, and use the results to argue for constrained models of the generation process founded on declarative links between resources and generator output.

Introduction

This paper is motivated by the perennial problem of knowledge acquisition for natural language generation (Scott, Power, & Evans 1998). The problem is that NLG requires rich linguistic knowledge that describes the form and meanings of utterances, and rich world knowledge that captures how to use these utterances appropriately. But this knowledge is rarely available before the NLG system is constructed. Acquiring such knowledge is a significant bottleneck. It will be especially problematic for dialogue systems, as they move from today's simple customer-service scenarios into more sophisticated domains such as help, tutoring, decision-support and entertainment.

This paper describes work in progress to conceptualize and map out new tools that might aid in knowledge acquisition for NLG in dialogue. Like (Scott, Power, & Evans 1998), I adopt the perspective that such tools will be especially powerful if they leverage natural language technology as part of their design and methodology. For example, such tools should allow system designers and developers whenever possible to create the resources that drive the system simply by providing illustrative natural language examples.

The methodological challenge is that NLG modules for dialogue have sophisticated requirements that go beyond

what NLG modules need to produce written documents. In dialogue, NLG modules must interface interactively with other dialogue system modules, they must achieve rich and flexible context-dependence, and more generally they must accommodate users on-the-fly instead of just producing one good, compromise result. Resources for NLG in dialogue must be correspondingly fine-grained. To specify such resources more effectively and more naturally, we need to characterize the needs for NLG in dialogue more precisely, we need to analyze the way NLG modules use their resources more systematically, and we need to respond by developing new and creative ways of developing modules and resources for NLG. This paper sketches some preliminary ideas for a program of research on knowledge acquisition for NLG in dialogue that addresses each of these issues.

I start by summarizing our experience with knowledge acquisition for dialogue from several end-to-end dialogue applications (Cassell *et al.* 1994; Stone *et al.* 2000; Cassell, Stone, & Yan 2000). We have found that the knowledge systems need to communicate in natural language is often orthogonal to their pre-existing domain knowledge, and that dialogue brings requirements of intentionality and compositionality that force this additional communication knowledge to be formalized explicitly.

In our experience, any tools that help in formalizing this new knowledge, no matter how simple, will be advantageous. As one such possibility, I consider specifying linguistic content for natural language generation by example. In specification by example, designers systematically provide sample utterances that illustrate desired ways for the system to articulate its domain-specific representations. I particularly consider utterances that function to identify an abstract entity from an underlying domain ontology. This means specifying generation of referring expressions by example. To interpret the specification, automatic tools reason *backwards* to a consistent body of communication knowledge which would support use of the sample utterance.

In presenting this idea, I emphasize that example specifications are programs, rather than observations. Treating examples as programs means judging inferred models of communication by whether they allow the system to behave in accordance with its specification, not by whether they match the actual states of speakers in attested utterances. Similarly, when programmed through example specifications,

NLG modules must be constructed for the transparency by which they achieve specified behavior, not necessarily for their sophistication, intelligence or cognitive plausibility.

I close by presenting a very provisional case study of specifying natural language generation by example: the generation of simple referring expressions specified for a fixed context. The interest in this case study is in exemplifying the kinds of results that should be forthcoming within this program of research. In particular, I offer tractable paired algorithms for interpretation and generation of such examples. Running the procedure to interpret a set of examples will either reconstruct provably correct communication knowledge that can generate the examples exactly as written, or will pinpoint specific flaws and model failures that prevent the examples from ever being generated. My implementation of these algorithms has been a short and straightforward logic programming exercise.

The problem of communication knowledge

I have been involved in a number of NLG projects aimed at constructing natural language utterances in context from deep, domain-specific representations. In the work reported in (Cassell *et al.* 1994), we used generative models, including a plan-based dialogue manager and grammar-based NLG, to produce and animate simple task-oriented dialogues in which two agents plan and carry out a real-world joint activity. In the work reported in (Stone *et al.* 2000), we generated verbal instructions in a physical maintenance domain from underlying action representations that could also be used to simulate, animate and validate maintenance procedures. In the work reported in (Cassell, Stone, & Yan 2000), we generated extended descriptions of objects and places, including words, intonation and coverbal gesture, as contributions to face-to-face dialogues with users in a real-estate domain.

One enduring lesson from these applications is that domain reasoning is usually best captured at a different level of granularity than appears in target application utterances. Sometimes domain reasoning requires detail that natural language suppresses. For example, domain talk might be peppered with the sorts of instructions in (1).

- (1) a Pick up the mug.
- b Pick up the book.
- c Pick up the box.

The actions required here will be very different: in one case, the actor must grab the object by the handle; in others, the actor must grab the object as a whole. Perhaps the actor should use one hand, perhaps two. A domain representation, suitable for validating or animating the underlying procedure, must mark the distinction (Douville, Levison, & Badler 1996). On the other hand, the instructions suppress all this detail. They simply highlight the change of state the action must achieve, and leave the actor of the instruction to work out the specific action to do for themselves. (On inference and indirection in instructions generally, see (Stone 2000).)

In other cases, coarser specifications may suffice for domain reasoning than natural language instructions require.

For example, contrast the instructions in (2).

- (2) a Slide the sleeve onto the elbow.
- b Reposition the sleeve at the joint.

These are complementary instructions; the second undoes the first. Not surprisingly then, the same representations of motion and change of state suffice for domain reasoning for both actions. However, the first instruction adds that the path of motion stays in contact with a surface, while the second instruction adds that the sleeve performs a specific function at the joint, and that the sleeve used to be at the joint at some earlier stage in the procedure.

This paper responds to a practical challenge of such mismatches. In building a natural language interface to domain information, you have to supply the system with a substantial body of new communication knowledge. You cannot simply translate an internal representation to the meaning of an utterance: that will leave out necessary information sometimes, and will include too much information other times. Rather you must spell out ways of creating natural descriptions that identify specified internal representations to interlocutors. For implementation, we can imagine the domain representations behind instructions like (1) and (2) as *terms* in some formal representation. These terms might be primitive, e.g., *a1*, or composite, e.g., *right-arm-move(handshape=grab1,target=(0.4,0.4,0.1))*, but in either case, these terms will have no meaningful linguistic structure. To identify these terms, we will have to have *predicates* that we can use to characterize them, as in (3).

- (3) a *slide(a1)*
- b *pickup(right-arm-move(handshape=grab1, target=(0.4,0.4,0.1)))*

Formulating an instruction for a user will be aggregating a selection of these predicates to form a grammatical utterance that characterizes the term representing the intended action uniquely from among the possibilities in the domain. In other words, it is a problem of generating a referring expression (GRE).

GRE is a well-studied and important subtask of NLG which has attracted increasing attention, particularly since the paradigmatic work of (Dale & Reiter 1995). The input to GRE is an entity, as characterized by a database of information about it that is presumed to be part of the common ground. The output is the semantics and perhaps also the syntax of a linguistic description that will allow the hearer to distinguish the entity uniquely from its salient alternatives. To build a GRE module requires identifying the context set of individuals that are explicit or implicit in application discourse, formalizing the relevant properties of all these individuals, and specifying how to compose these properties into linguistic descriptions. The GRE module thus epitomizes the resource-intensive character of NLG, and highlights the gap between NLG output and the formal machinery of knowledge bases and grammars that underpins it.

Indeed, the knowledge acquisition problem here is as clear as it is severe. The system's knowledge of linguistic predicates and the terms they apply to is communication knowledge. It is separate from domain knowledge. And so,

even after the system has a strong and useful domain representation, this communication knowledge would need to be specified fully from scratch before the system can communicate with its users.

Compositionality and intentionality as requirements for dialogue

The discussion so far has been motivated by experience with NLG for dialogue, but has not been specific to dialogue. Indeed, it is important to recognize that this abstract take on the problem does not much constrain the architecture of the generator or the techniques that the generator will use. For example, it is compatible with a shallow generator that primarily uses canned text or templates. Such a shallow generator still needs a link between domain objects and the expressions that it uses to describe domain objects, which we can formalize as a complex predicate characterizing domain objects, as in (4).

(4) *slide-sleeve-onto-elbow(a1)*

And even in such a shallow generator, we still ought to record the system's commitment that its audience can recognize the link between its canned descriptions and the domain objects they are intended to identify or express.

However, there are substantial constraints on NLG that come when we attempt to use the generator as part of a dialogue system. For example, the generator needs to be sensitive to context. It should be able to identify the same domain object with many different expressions, depending on the state of the dialogue.

- (5) a Q: What do I do next?
A: Slide the sleeve onto the elbow.
b Q: What do I do with the sleeve?
A: Slide it onto the elbow.
c Q: How do I get the sleeve onto the elbow?
A: Slide it there.

The suggestion in (5) is that in all cases the NLG module is given the same goal, of identifying the action represented by term *a1*, and that the NLG system links up with the changing context to achieve the variation in the system's responses. Note, by the way, that rendering these different examples in text as in (5) underestimates the difference among them, since they also have to be realized with different intonation and perhaps even with different embodied coverbal signals. In order to achieve such flexibility, generators for dialogue need compositional analyses of their utterances, so they can streamline the formulations of any subconstituent to exploit a connection to the context. (Krahmer & Theune 2002) represent a good recent investigation of the inherent context-sensitivity of general approaches to GRE.

In dialogue, the system as a whole also needs to be able to follow up on what the generator has produced. Consider the fragments in (6), envisaged as alternative ways a dialogue could continue after the system utters *slide the sleeve onto the elbow*.

- (6) a Q: What do you mean sleeve?
A: That tube around the pipe at the joint.

- b Q: What do you mean elbow?
A: That place where the pipe hooks and bends to the left of the joint.
c Q: What do you mean slide?
A: Just push the sleeve gently over along the pipe.
d Q: What do you mean onto?
A: The sleeve can hang there safely out of the way while you complete the repair.

In each case, the question seems to call for further information not about the general meaning of the system's terms, but rather about the particular way in which the generator intended the term in its utterance. To respond appropriately to such questions, the system must record and pursue the intended connections between the words in its prior utterance and its domain-specific ontology of objects, actions, and states. In particular, the system could supply the clarifications here by constructing alternative referring expressions that supply new information about those same referents. Again, in order to achieve such flexibility, generators for dialogue have to have a compositional analysis of their utterances which allows them to pursue alternative formulations of any subconstituent. For a more systematic exploration of the connections between fragments in dialogue and the compositional interpretation of utterances, see (Ginzburg & Cooper 2001; Cooper & Ginzburg 2002).

Finally, dialogue differs from written language in that the generator is not just conveying information to an abstract user, but must actively work at each stage of a conversation to meet the expectations of a specific partner. A classic example of this is entrainment on vocabulary (Brennan & Clark 1996; Brennan 1996). When redescribing the same objects, participants in a conversation are expected to reuse the same vocabulary. Changes in vocabulary are more difficult to understand and can even give rise to conversational implicatures. Contrast:

- (7) a Q: Do I slide the sleeve all the way to the bend?
A: Yes, slide it onto the bend.
b Q: Do I slide the sleeve all the way to the bend?
A: Yes, slide it onto the elbow.

Again, in order to achieve such flexibility, generators for dialogue have to have a compositional analysis of their utterances which allows them to assess and meet their partner's expectations, at any subconstituent.

Introducing NLG by example

Although dialogue demands a sophisticated formal model of communication, we start as designers typically knowing just what our system should say. We might know, for example, that a good way to describe term *a1* as the answer to the question *what next*, might be *Slide the sleeve onto the elbow*. And that a good description for term *a3* as the answer to the question *what next*, might be *Reposition the sleeve at the joint*. What are the prospects for designing NLG systems explicitly in these terms?

I think the prospects are quite good indeed. One way to see this is simply to work backwards from a simple description, and see how routinely the knowledge base to generate it could be constructed. Suppose, then, that we want

to describe d_1 as *the big black dog*. Let's assume that *dog* expresses the property dog' , *black* expresses the property $black'$ and *big* expresses the property big' . Then to support the properties that we attribute to d_1 , we must at least have the mutual knowledge in (8).

- (8) a $dog'(d_1)$
 b $black'(d_1)$
 c $big'(d_1)$

But that can't be all our knowledge base contains. Otherwise we would say *it* or *the dog*. The generator must have alternatives to d_1 in mind. For example, maybe there is another dog d_2 that is big but not black, and a third dog d_3 that is black but not big, as in (9).

- (9) a $dog'(d_2)$ $dog'(d_3)$
 b $black'(d_3)$
 c $big'(d_2)$

(In (9), I assume a model of coordination in reference resolution in which participants rely on positive information when possible. So if we don't say that d_2 is big, then by default we won't consider d_2 as a referent for a description that features the property *big*.)

Now the generator has a motivation for every element in its description of d_1 : without *black*, the referent might be d_1 or d_2 ; without *big*, the referent might be d_1 or d_3 ; and without *dog*, the generator wouldn't have an English noun phrase expressing the right properties. So we can predict that the generator realistically should and would be able to use *the big black dog* to identify d_1 in this context.

Obviously, we need to be able to generalize these ideas beyond individual cases in order to develop specification by example as a serious methodology for NLG. Before proceeding, there are two important points to make about how such methodology may differ from more familiar methods in NLG, and empirical natural language technology generally. The examples here are programs, not corpora; and the decision-making that generates them must be simple rather than sophisticated. I explain below.

Examples: Programs not corpora

It is important to think of a collection of examples as a specification, not as a corpus. As a specification, the examples will cover the requirements of the system comprehensively; they cannot necessarily be obtained from representative interactions, and at the same time need not even originate from dialogue. The collection represents someone's guess about what is good for the system, rather than what is natural or what people or previous systems would tend to do. The specification takes the form of natural language examples not because we expect that an NLG system will be directly assembled from naturally-occurring resources, but because designers can consult their intuitions and draw on their existing talents to create collections of natural language utterances in a way that they cannot do in creating predicate-calculus knowledge bases or formal grammars. For more on the difference between a specification for NLG and a corpus, see (Reiter & Sripada 2002).

Potential applications of NLG in entertainment highlight the difference between corpus resources and specification examples. Consider creating a virtual conversational character as part of an immersive interactive experience. If the project requires designers simply to script the character's dialogue, they can draw on the kinds of abilities screenwriters and actors already cultivate to convey the character's personality, values, background and relationships—to bring the character to life. By contrast, if the project requires us to encode that same information explicitly in terms of atomic knowledge bases and compositional grammars for the agent's utterances, the effort will rapidly become counterintuitive and unmanageable. But of course, creating a character must remain a creative endeavor: there will be no predefined corpus for the character and perhaps even no way in principle to collect one!

One consequence of viewing examples as programs is that questions about what is "really going on" in specification examples are misplaced. Coming up with an analysis of specification examples in terms of communicative goals is a way of modeling those examples. In characterizing the examples in formal terms, we aim simply derive new functionality from them. For example, if we explain a modifier by postulating an alternative entity in an abstract domain model, the formal model simply ensures that the NLG system includes that modifier in unrestricted contexts, but leaves out the modifier in certain kinds of more specific contexts. There's no reason to ask whether that alternative entity is "really" part of the domain; actually, neither the system nor the user will ever talk explicitly about it. Similarly, there's no reason to ask whether the corresponding sample utterance "really" carries the communicative goal of distinguishing something from this alternative. The utterance was not produced meaningfully to express some intrinsic communicative goals, the way an utterance in a corpus might be; the utterance is simply part of a program. Actually in this case, the system does not so much construct the utterance to act in accordance with its goals, as reconstruct its communicative goals in order to act in accordance with the utterance. All we can expect or ask is that knowledge, grammar, goals and utterances are consistent.

A further consequence of viewing examples as programs is that it may prove necessary to "debug" a collection of examples. There may be errors, oversights and inconsistencies in the interactions that designers anticipate for the system, just as there are in any large creative project. There is a need for *verification and validation* of the examples to check for their coherence (Barr & Gonzalez 2000; Barr & Klavans 2001). Deriving more formal resources to model the language in the collection is part of this verification and validation process. It can offer an inexpensive diagnostic that specified linguistic behavior portrays a world of individuals in a way that is consistent and that interlocutors can be expected to recognize. This check would fail for some sets of examples. For instance, a specification that said d_1 could be described as *the black dog* and d_2 could be described as *the dog* would not lead to a consistent model. In this specification *the dog* must be ambiguous. Of course, there may be many contexts or situations for this application

in which the generator ultimately should be able to produce *the dog*. But generator's dialogue models will already be able to adapt its contributions to context this way. Here, the description occurs as part of a program that derives the system's resources for NLG, and, as such, the abbreviation probably reflects the designer's oversight in failing to take into account the two entities d_1 and d_2 that are relevant here. The appropriate response is to revise the NLG examples.

Invertible NLG: Economical NLG

In constructing communicative resources that allow a generator to produce specified examples, we are really predicting and controlling the decisions that the NLG system makes. In fact of course, the decisions that a generator makes vary depending on the algorithms that the generator implements. We will only be able to predict and control the decisions of a generator if the generator acts in a very straightforward way. In this section, I return to the example of (8) and (9) to make this intuitive point in a precise way. Dale and Reiter's computational deconstruction of Grice's Maxim of Brevity (1995), with its well-known and useful taxonomy of decision-making in generation, provides a helpful guide to the discussion.

The example of (8) and (9) implicitly appeals to an aspect of Reiter's *local brevity* interpretation of Grice's maxim. In particular, we used the condition of *no unnecessary components*: a GRE system respects this condition if no element in its utterance can be removed to leave a briefer complete utterance that refers uniquely. (This interpolates the common idea that GRE derives syntactic derivation structures rather than mere semantics, e.g., (Stone & Doran 1997).) This condition is quite a weak constraint on the generator; many different generation algorithms would respect it. I will actually assume something stronger: that semantically, the generator is *indifferent* between all uniquely identifying descriptions that satisfy this condition, and is capable of generating any of them nondeterministically. I will call this *economical GRE*. This in fact uniquely determines what is computed in GRE. *To derive a specification for GRE by example thus means that we construct a knowledge base and grammar for which economical GRE can generate each sample description to identify its target referent.*

Why assume economical GRE in particular? Consider the other possibilities. Dale and Reiter in fact argue for the *incremental heuristic* interpretation of brevity: a generator ensures that at each stage when the generator adds a property to a description, the property appears to rule out some open alternatives. The knowledge based presented in (8) and (9) goes beyond what is necessary for such a generator. If the generator builds the description using *the big dog* as an intermediate stage, it is not necessary for d_2 or d_3 to be black. d_2 is ruled out at this stage because it is not big. Since d_3 is also a *big dog*, the generator will continue, and can add the property *black* at the next stage to rule out d_3 . Under the incremental heuristic, the generator then outputs its result *the big black dog* uncritically; it does not take into account the fact that *the black dog* uniquely identifies d_1 in this case. Conversely if the generator builds *the black dog* as an inter-

mediate stage, it is not necessary for d_2 or d_3 to be big.

Note how, on the incremental heuristic, the order of choice that the generator follows interacts with the world knowledge that the generator needs to have. On the incremental heuristic, the order of these choices can arise in two ways. The generator may entertain properties in a lexically-specified order, and in this case the order represents an aspect of the generator's grammar. In this case knowledge base and grammar interact. Alternatively, the generator may pick properties heuristically, for example using a greedy heuristic to pick the property that rules out the greatest number of alternatives first. In this case, knowledge base and control interact, and it may be necessary to populate the knowledge base with alternatives in quantity in order to motivate specific choices for the greedy generator. Neither of the two cases is very attractive.

Another interpretation that Dale and Reiter consider is *full brevity*: a generator produces a description that uses as few properties as possible. When used to reconstruct the context behind specific examples, full brevity differs from economical GRE only when multiple descriptions of the same object are in play, as in (10).

- (10)a d_1 is *the big black shaggy dog*
 b d_1 is *the affable purebred racing dog*

A full-brevity generator will produce (10) only when shorter descriptions such as *the black racing dog* do not refer uniquely. In contrast, economical GRE does not treat *the black racing dog* as a competitor to (10) because it involves a new combination of properties. The full-brevity generator requires correspondingly more alternatives in the knowledge base. (Reiter's complete definition of *local brevity* includes a related condition that would also require more alternative entities.) Suppose that a target referent r is known to have N properties, but is identified in examples using sets of L of them. Then a full-brevity generator requires an alternative to witness each subset of $L - 1$ of the N properties of r . Otherwise that description would identify r uniquely and would be preferred over the specified descriptions. This is of course exponentially many alternatives in the worst case. So here, it is the combinatorics of the approach to generation that makes the approach unattractive. (The impossibility of handling examples where different references to the same object of include different numbers of properties is another serious drawback.)

For building GRE systems using examples as specification, we have strong reasons to assume, as a working hypothesis, the economical generator that includes no unnecessary components but is otherwise indifferent to its alternatives. We have seen why the economy is appropriate, but why the indifference? The reason is that to assume more semantic structure to the generator would also mean complicating the semantic knowledge bases and grammars we reconstruct. Our discussion of the greedy heuristic hints why this is: we need to provide facts that tweak the decisions the generator makes, an awkward and potentially open-ended process. Instead, we leave open the possibility that the generator has other preferences that are induced from other analyses of potentially different data. It makes much more sense

to flesh out such preferences on their own terms—perhaps using pragmatic notions, or using perhaps statistical decision theory—than to attempt to reconstruct such preferences by adapting a logical knowledge base.

An initial case study

The remainder of this paper offers a case study looking within this framework at the generation of *simple referring expressions* specified for a *fixed context*. A simple referring expression is a definite singular noun phrase in which each descriptive element attributes a grammatically-specified property. As usual, I will assume that referring expressions such as those in (11) are simple.

- (11)a The pencil.
 b The tiny mischievous chihuahua.
 c The big heavy Louis-XV oak armoire.

On the other hand, the noun phrases in (12) are not:

- (12)a The pencils.
 b That tiny mischievous chihuahua.
 c The oak armoire in the master bedroom.

(12a) is plural; it requires interpretation by maximization rather than identification. (12b) involves an indexical descriptive element whose contribution is fixed by the context not the grammar. And (12c) is a relational description in which some elements, such as *bedroom*, describe not the intended referent but another related entity.

In taking the context as fixed, we assume that the specification consists of a set of descriptions paired with their target referent. We assume that each example expression is specified to take its reference from a single *context set* C of salient individuals, including of course all the referents of the specified descriptions as well as possibly others. The goal is to construct a knowledge base, grammar and a single context set C such that economical GRE can generate each sample description to identify its target referent against C . (This does not mean that C must be the only context set in the dialogue system, of course.)

Definitions and Assumptions

We can idealize a simple referring expression as a set D of grammatically-specified properties $P_1 \dots P_n$. We idealize away from grammatical distinctions by assuming that exactly one property P_1 corresponds to the syntactic head of the description and the others correspond to syntactic modifiers of the head; under this assumption the set D determines a unique syntactic derivation structure; write D_M for the modifiers of D . We further assume that the status of descriptors as head and modifiers is part of the specification. (See (Stone *et al.* 2001) for an exploration of the syntax-semantics interface in GRE, under similar assumptions.)

As is standard, we assume that objects are identified by properties that are mutually known as determined by a knowledge base (KB) of information; write $[cr]p$ when proposition p is part of the conversational record according to the KB. Then the interpretation of D , $\mathbf{I}(D)$, with context set C is defined as in (13):

$$(13) \{x \in C : [cr]P_1(x) \wedge \dots \wedge [cr]P_n(x)\}$$

This is just the set of elements from the context set known to satisfy all the properties in the description! A simple referring expression refers uniquely to entity r in our given context set C when $\mathbf{I}(D) = \{r\}$.

The generation of simple referring expressions is a benchmark problem in the NLG literature; see (Dale & Reiter 1995; Krahmer & Theune 2002; van Deemter 2002). A common practical and psychologically-motivated algorithm is to incorporate properties incrementally into the description, while making progress, until the description is sufficient to identify the intended referent, then (for economy) checking that no descriptors are unnecessary. (van Deemter 2002) also shows that the incremental approach is *complete*, in that it can never fail to derive an identifying grammatical simple referring expression for a target when one exists.

Our objective is to construct specifications for generation of simple referring expressions by example. The input is a set S of pairs of the form $\langle D, r \rangle$ where D is a description and r is its intended referent. For convenience, we define the *referents of* S , R_S as $\{x : \langle D, x \rangle \in S\}$. We define the *characterization of* x in S , $char_S(x)$ as $\{P : \langle D, x \rangle \in S \wedge P \in D\}$.

When confronted with such a pair $\langle D, r \rangle$, we must ask what things would have to be like for us to refer to r as D . It turns out that there are two key concepts.

- *Support*. We must have the information that r fits the description. Formally, the KB supports the attribution of a property P_i to r if $[cr]P_i(r)$; the KB supports the attribution of description D to r if it supports each $P_i \in D$.

If the description is not supported it could not be understood or generated—cf. (8).

- *Motivation*. We must have a reason to use each of the properties we use to describe r . Formally, the KB motivates a property P_i in describing r with D in C if D uniquely identifies r in C but $D - \{P_i\}$ does not. The KB motivates a description D in describing r in C if it motivates each property $P_i \in D_M$ (i.e., other than the head P_1).

If a descriptor is not motivated, the generator might just as well have left the descriptor out—cf. 9.

Economical GRE can produce a description D to identify r in C just in case the KB supports the attribution of D to r and motivates D in describing r in C .

Our results require a further assumption: the possibility of *alternatives*. Let A be the set of all properties that figure in descriptions in a collection of examples. Then an individual x is an *alternative* to y if condition (14) holds.

$$(14) \forall P \in A : [cr]P(x) \Rightarrow [cr]P(y)$$

If x is an alternative to y then x cannot be distinguished from y using properties from A ; y satisfies the same ones. The significance of such an alternative is that we know that adding x to a context set that contains y cannot cause descriptions that refer to *other* entities to become ambiguous. If these descriptions fit x , they fit y too. Of course, we will want to distinguish y from x using properties from A , since descriptions of y should not be ambiguous either! More generally, we would expect that x could be distinguished from y using

other properties. So we imagine that there is some property $P \in A$ such that y has P whereas x is different— x has some other unspecified property $P' \notin A$ instead of P . It sounds more mysterious than it is—in the example of (8) and (9), both d_2 and d_3 are alternatives to d_1 .

When No Solution Exists

Specification by example is a *programming* methodology, so the system must be capable of reporting its failures to the system-builder, and helping the system-builder adapt the examples to fix failures. Thus, it is significant in this paradigm that we can characterize precisely and declaratively why a body of examples S might fail to yield a knowledge base and context set. There are two reasons: failure of support and failure of motivation.

(15) S has a *failure of support* if
 $\exists x \in R_S. \exists \langle D, y \rangle \in S. x \neq y \wedge D \subseteq \text{char}_S(x)$.

If S has a failure of support, some description D in S must be ambiguous between two things x and y . When this happens, there is no KB and C for which economical GRE can generate the descriptions of S . To see this, observe that a knowledge base for S must contain $[cr]P(x)$ for $P \in \text{char}_S(x)$. Otherwise the corresponding description of x is not supported. And of course we must have $x \in C$. But then $\{x, y\} \subseteq \mathbf{I}(D)$ so D does not refer uniquely and cannot be generated.

(16) S has a *failure of motivation* if
 $\exists \langle D, x \rangle \in S. \exists \langle D', x \rangle \in S. D \subset D'$.

If S has a failure of motivation, we cannot populate the context set with alternatives to x to justify D' while keeping D unambiguous. Formally, if $\langle D', x \rangle$ is motivated, then for any $P \in D'$, there is some $y \in C$ such that $x \neq y$ and $y \in \mathbf{I}(D' - \{P\})$. Consider in particular any $P \in D' - D$. Then we must have $D \subseteq D' - P$ so $\{x, y\} \subseteq \mathbf{I}(D)$ so D does not refer uniquely and cannot be generated.

Say that S is *flawed* if S has a failure of support or a failure of motivation. We have just proved that if S is flawed, there is no KB and C for which economical GRE can generate the descriptions of S .

Constructing Possible Solutions

Suppose instead that S is not flawed. To reconstruct an input for S we introduce the notation $f(x, P, D)$; $f(x, P, D)$ names a free witness assumed as an alternative to x that satisfies all the properties in D except P . Now consider C defined as in (17a) and KB as defined in (17b).

(17)a $R_S \cup \{f(x, P, D) : \langle D, x \rangle \in S \wedge P \in D_M\}$
 b $\{[cr]P(x) : x \in R_S \wedge P \in \text{char}_S(x)\} \cup$
 $\{[cr]P'(f(x, P, D)) : f(x, P, D) \in C \wedge$
 $P' \in D - \{P\}\}$

The reader can check that (8) and (9) meet this definition with $d_2 = f(d_1, \text{black}', \{\text{dog}', \text{big}', \text{black}'\})$ and $d_3 = f(d_1, \text{big}', \{\text{dog}', \text{big}', \text{black}'\})$.

We will show that in this context economical GRE generates the descriptions of S . We need to show that for each $\langle D, r \rangle \in S$, economical GRE can produce D to identify r in C on the basis of this KB. Clearly, the KB supports

the attribution of D to r , so $r \in \mathbf{I}(D)$. Moreover, for each property $P_i \in D_M$, $D - \{P_i\}$ does not uniquely identify r : $f(r, P_i, D) \in \mathbf{I}(D - \{P_i\})$. So to show that the KB motivates D in describing r in C it is enough to show that $\{r\} = \mathbf{I}(D)$. We proceed by contradiction, and assume $x \in \mathbf{I}(D)$ with $x \neq r$. There are three cases:

- $x \in R_S$. But then $\exists x \in R_S. \exists \langle D, r \rangle \in S. x \neq r \wedge D \subseteq \text{char}_S(x)$. That is, S has a failure of support—by assumption, impossible.
- $x = f(y, P', D')$ with $y \neq r$. In this case x is an alternative to $y \in R_S$. Since $x \in \mathbf{I}(D)$, $y \in \mathbf{I}(D)$. This is the same case which we have just shown to be impossible.
- $x = f(r, P', D')$. But what we know of x is only that x satisfies $D' - \{P'\}$. That means $D \subseteq D' - \{P'\}$ and thus $D \subset D'$. So S has a failure of motivation. Again, this is impossible by assumption.

Complexity and Optimization

Let m be the length of the longest description in S and let k be the number of descriptions in S . Inspection of (17) shows that C has $O(mk)$ objects and KB has $O(m^2k)$ facts. On the reasonable assumption that we have recourse to an open-ended number of properties that make independent partitions of the universe, we would expect m to grow as $O(\log k)$. So we would expect in practice that C grows no faster than $O(k \log k)$ and that KB grows no faster than $O(k \log^2 k)$.

Indeed, we might also expect that collections of descriptions are coherent, in that the requirements of the referential task are sufficient to motivate the descriptions in the collection. In particular, we need not add any $f(r, P, D)$ to C when there is another $x \in R_S$ such that $f(r, P, D)$ is an alternative to x and $x \neq r$. In such a case, x itself is a witness that the KB motivates property P in the describing r as D in C .

Related work

This paper is the first to investigate GRE from the standpoint of specification-by-example. In previous work, sample utterances have been used to induce linguistic resources for NLG (Vargas & Mellish 2001; Pan & Weng 2002), to induce knowledge bases for NLG (Scott, Power, & Evans 1998), and to describe preferences among alternatives in generation (Langkilde 2000; Bangalore & Rambow 2000; Ratnaparkhi 2000; Walker, Rambow, & Rogati 2001). Although GRE shares a motivation for specification-by-example with other aspects of NLG, GRE brings special requirements because successful GRE so crucially depends on the interpretation of language in context. This dependence means that a solution to GRE must be cast in the terms of a knowledge-rich model rather than the heuristic terms that may be easier to apply to examples (Vargas & Mellish 2001; Pan & Weng 2002). An important aspect of this model is to consider not merely the facts that support the description, as in (Scott, Power, & Evans 1998), but also the potential alternatives that implicitly motivate the description. However, the model merely predicts candidate descriptions; it does not rank them. This makes the techniques and results orthogonal to the majority of work in inducing NLG from data, which aims instead to rank such alternative candidates.

Conclusions and Future Work

This paper has explored techniques that may alleviate the knowledge acquisition problem for NLG in dialogue by allowing designers to build systems from higher-level specifications. In particular, I have introduced the idea of specifying GRE behavior from sample utterances. This is best understood as setting up a problem of Gricean reasoning. The problem is to frame GRE tasks for the system in such a way that it can make its choices in a predictable way to match the specified examples.

To characterize choice in GRE, I draw on Reiter's discussion of *local brevity* (Reiter 1990; Dale & Reiter 1995); I assume that the generator does not include unnecessary components in descriptions, but that all other identifying descriptions are legitimate alternative candidates that will be ranked independently by pragmatic considerations. In this case, we can ignore search control, and use separate information to reconstruct the generation grammar and to reconstruct the generation knowledge base. Although not all sets of examples can be reconstructed under this assumption, we can identify those that cannot be precisely and declaratively; these conditions seem to single out specifications that are genuinely problematic cases. What's more, when we can derive grammar and knowledge bases, they are relatively compact. Further research is underway to extend the procedure to examples of richer discourse, adopting a broader knowledge-based perspective of NLG as goal-directed grammatical reasoning (Stone *et al.* 2001).

Acknowledgments

Thanks for comments from the anonymous reviewers, from Valerie Barr, Owen Rambow, Paul Tepper, Kees van Deemter and Bonnie Webber, and the NLG research group of ITRI, University of Brighton.

References

- Bangalore, S., and Rambow, O. 2000. Exploiting a probabilistic hierarchical model for generation. In *International Conference on Computational Linguistics (COLING)*.
- Barr, V., and Gonzalez, A. 2000. Validation and verification of intelligent systems—what are they and how are they different? *Journal of Experimental and Theoretical Artificial Intelligence* 12(4).
- Barr, V., and Klavans, J. 2001. Verification and validation of language processing systems: Is it evaluation? In *ACL 2001 Workshop on Evaluation Methodologies for Language and Dialogue Systems*, 34–40.
- Brennan, S. E., and Clark, H. H. 1996. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology: Learning, Memory and Cognition* 22(6):1482–1493.
- Brennan, S. E. 1996. Lexical entrainment in spontaneous dialog. In *International Symposium on Spoken Dialogue*, 41–44.
- Cassell, J.; Pelachaud, C.; Badler, N.; Steedman, M.; Achorn, B.; Becket, T.; Douville, B.; Prevost, S.; and Stone, M. 1994. Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *SIGGRAPH*, 413–420.
- Cassell, J.; Stone, M.; and Yan, H. 2000. Coordination and context-dependence in the generation of embodied conversation. In *First International Conference on Natural Language Generation*, 171–178.
- Cooper, R., and Ginzburg, J. 2002. Using dependent record types in clarification ellipsis. In *EDILOG 2002*, 45–52.
- Dale, R., and Reiter, E. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science* 18:233–263.
- Douville, B.; Levison, L.; and Badler, N. 1996. Task level object grasping for simulated agents. *Presence* 5(4):416–430.
- Ginzburg, J., and Cooper, R. 2001. Clarification, ellipsis and the nature of contextual updates in dialogue. King's College London and Göteborg University.
- Krahmer, E., and Theune, M. 2002. Efficient context-sensitive generation of referring expressions. In van Deemter, K., and Kibble, R., eds., *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*. CSLI. 223–265.
- Langkilde, I. 2000. Forest-based statistical sentence generation. In *Applied Natural Language Processing Conference*, 170–177.
- Pan, S., and Weng, W. 2002. Designing a speech corpus for instance-based spoken language generation. In *Second International Natural Language Generation Conference*, 49–56.
- Ratnaparkhi, A. 2000. Trainable methods for surface natural language generation. In *First meeting of the North American Chapter of the Association for Computational Linguistics*, 194–201.
- Reiter, E., and Sripada, S. 2002. Should corpora texts be gold standards for NLG? In *Second International Natural Language Generation Conference*, 97–104.
- Reiter, E. 1990. The computational complexity of avoiding conversational implicatures. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, 97–104.
- Scott, D.; Power, R.; and Evans, R. 1998. Generation as a solution to its own problem. In *Proceedings of the Ninth International Workshop on Natural language Generation*, 256–265.
- Stone, M., and Doran, C. 1997. Sentence planning as description using tree-adjointing grammar. In *Proceedings of ACL*, 198–205.
- Stone, M.; Bleam, T.; Doran, C.; and Palmer, M. 2000. Lexicalized grammar and the description of motion events. In *TAG+: Workshop on Tree-Adjoining Grammar and Related Formalisms*, 199–206.
- Stone, M.; Doran, C.; Webber, B.; Bleam, T.; and Palmer, M. 2001. Microplanning with communicative intentions: The SPUD system. *Under review*.
- Stone, M. 2000. Towards a computational account of knowledge, action and inference in instructions. *Journal of Language and Computation* 1:231–246.
- van Deemter, K. 2002. Generating referring expressions: boolean extensions of the incremental algorithm. *Computational Linguistics* 28(1):37–52.
- Vargas, S., and Mellish, C. 2001. Instance-based natural language generation. In *2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, 1–8.
- Walker, M. A.; Rambow, O.; and Rogati, M. 2001. SPoT: A trainable sentence planner. In *2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, 17–24.