Domain Inference in Incremental Interpretation

David DeVault and Matthew Stone

Computer Science and Cognitive Science
Rutgers University
{*ddevault,mdstone*}*@cs.rutgers.edu*

Abstract

Speakers in dialogue describe domain-specific actions, goals, conditions and plans using the general resources of their linguistic knowledge. Interlocutors must recognize these descriptive connections through inference—and they must do so incrementally, since they need interpretations of partial utterances to inform their on-line participation in conversational interaction. This paper explores techniques that dialogue systems can use to achieve incremental interpretation, even when domain reasoning is modular, genuinely nonlinguistic and potentially expensive. We base our discussion on an implemented prototype natural language interface, FIGLET, that combines plan-recognition with real and discrete constraint-satisfaction to interpret English instructions in a drawing domain. FIGLET provides a proof-of-concept of the feasibility of such incremental interpretation, a testbed for the quantitative evaluation of the tradeoffs involved, and a case study in the methodological challenges that remain for future work.

1 Introduction

We are experts at formulating instructions in natural language that tell our collaborators what we expect of them. But our instructions are infamous for the "common sense" they seem to presuppose. For example, when we formulate or carry out instructions, we seem to never even consider harmful or off-topic actions as possible interpretations, no matter how precisely they fit what we say. An increasing range of projects in AI aim to endow computer systems with similar expertise [1,2,6,15]. This work reveals that in constrained domains, the knowledge required to characterize domain actions is increasingly available, and systems can increasingly reason efficiently with this knowledge to draw conclusions about actions and their effects in context.

In this paper, we explore the computational infrastructure required for such applications of inference to incremental natural language interpretation. As a case study, we consider an implemented drawing application, described in Section 2, in which the user can use language to instruct the system to draw a caricature of an expressive face. In this application, instructions as expressed in language may be vague, ambiguous or both, and only domain reasoning about the inferred structure of the figure will resolve the underspecification. To support interactive dialogue, our interface needs to be able to carry out this reasoning incrementally, over provisional constituents.

Despite this dependence of utterance interpretation on incremental domain reasoning, we assume that language has no say in the form of domain representations or the models and processes of domain inference. Rather, we assume that domain reasoning is modular, genuinely nonlinguistic, and potentially expensive. The challenge is thus to integrate domain reasoning with linguistic interpretation in an efficient, modular and scalable architecture. We describe our solution in Section 3. Despite the potential pitfalls, we have achieved an effective integration of domain reasoning and interpretation, through our own analysis of the inference problems the system faces. As we argue in Section 4, the principal challenge for future work is to found such integration on general abstractions rather than meticulous analysis.

2 Interpretation and modular domain inference

2.1 Interpretation

The computational problem of interpretation that practical dialogue systems need to solve is to identify the intention motivating an utterance in context. By recognizing these intentions, dialogue systems can use utterance interpretations directly to inform their interactions with users. For example, Allen, Ferguson and Stent [1] describe and motivate a general architecture for natural language dialogue systems, which is designed to give a central place to the intentions behind utterances in collaboration. More generally, Rich, Sidner and Lesh [11] discuss how any practical interface can collaborate with its users by modeling their intentions, while Stone [14] characterizes linguistic interpretations for dialogue as a special case of general intention representations.

Concretely, consider the interaction in Figure 1, in which a user instructs our natural language interface, FIGLET, to draw an iconic expressive face. At each step, the system responds by performing an action that most closely reflects its assessment of what the user intended it to do. In pursuing such dialogues, users devise plans to achieve desired real-world results, and pursue those plans more or less systematically. By recognizing these plans, an interface can predict a limited set of candidate actions that the user might currently have in mind. See Lesh, Rich and Sidner [8] for specific models of the different strategies by which users act in interfaces.

The remaining task of interpretation is to use the action description provided in the user's utterance to identify one of the candidate actions precisely enough to carry out a cooperative response. In the drawing domain of Figure 1, for example, the repertoire of system actions includes adding new objects, moving them, resizing them, and changing their shape. These

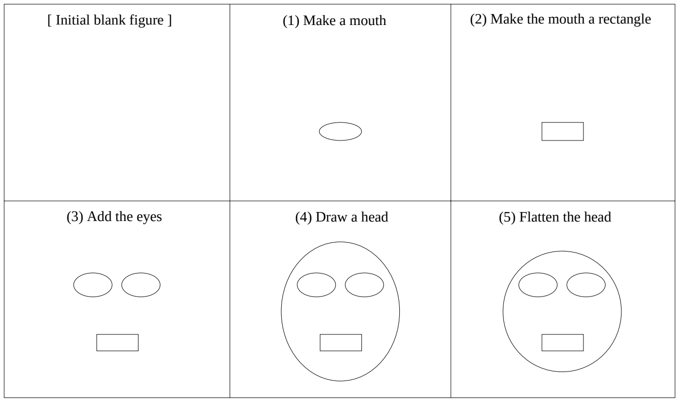

Fig. 1. Interacting with FIGLET.

actions allow users to build figure parts by introducing shapes and revising them. Users' domain plans organize these actions hierarchically into strategic patterns. For example, users characteristically complete the structures they begin before drawing elsewhere; and once they are satisfied with what they have, they proceed in natural sequence to a new part nearby. Users' proposed actions at any point are thus constrained by the drawing they have created and their focus of attention within it. The effect of user instructions is to select an action from the set of candidate actions at each point, and to help determine values for any free action parameters.

In this characterization of interpretation, the central reasoning task is to match the shared contextual background against the linguistic descriptions formulated by the user. To specify these correspondences, we work at the knowledge level, as understood by Levesque [9]. We use logic as the interface between the linguistic processing and the domain reasoning required to interpret instructions. In particular, we use *constraints*, or logical conjunctions of open atomic formulas, to represent the contextual requirements that utterances impose; we view these constraints as presuppositions that speakers make in using the utterance. We assume matches take the form of instances that supply particular domain representations as suitable values for variables. Stone [13] motivates this framework in detail.

As a concrete example, (1a-c) records the presuppositions we assign to an utterance of *Make the face bigger*.

(1) a. $\mathit{simple}(A) \wedge \mathit{target}(A, X) \wedge \mathit{holds}(\mathit{now}, \mathit{fits\_plan}(A)) \wedge$
$\mathit{holds}(\mathit{result}(A, \mathit{now}), \mathit{visible}(X)) \wedge$

b. $\mathit{number}(X, 1) \wedge \mathit{holds}(\mathit{now}, \mathit{type}(X, \mathit{face})) \wedge$

c. $\mathit{holds}(\mathit{now}, \mathit{size}(X, \mathit{SO})) \wedge \mathit{region}(R, \mathit{bigger\_than}, \mathit{SO}) \wedge$
$\mathit{in\_region}(\mathit{SN}, R) \wedge \mathit{holds}(\mathit{result}(A, \mathit{now}), \mathit{size}(X, \mathit{SN}))$

We formulate these constraints in an expressive ontology. We have terms and variables for *actions*, such as A; for *situations*, such as *now* and $\mathit{result}(A, \mathit{now})$; for *objects*, such as X; and for quantitative points and intervals of varying dimensionality, such as S and SO (two-dimensional points recording size as width and height) and R (a region in size space). We can characterize entities in terms of the state of the visual display in different situations; for example $\mathit{holds}(\mathit{now}, \mathit{size}(X, \mathit{SO}))$ means that SO is the current ("old") size of X, and $\mathit{holds}(\mathit{result}(A, \mathit{now}), \mathit{size}(X, \mathit{SN}))$ means that SN will be the new, possibly different size of X once action A is carried out. We can characterize entities in terms of causal relationships in the domain; for example $\mathit{target}(A, X)$ means that action A directly affects X, and the constraints of (1c) together mean that carrying out action A causes X to have a new bigger size SN. And we can characterize entities in terms of the model of the user's attention and intentions; for example $\mathit{simple}(A)$ means that the action A is a natural domain action rather than an abstruse one, and $\mathit{holds}(\mathit{now}, \mathit{fits\_plan}(A))$ means that A is a possible action given the user's current plan state.

(1) is structured to show how the overall presupposition of the utterance factors compositionally into contributions from the syntactic components that make it up.[1] Overall, then, (1) characterizes natural and contextually appropriate actions that affect a single face X and that bring about a situation in which X is visible and has a size SN larger than its current size.

2.2 Using domain inference to interpret ambiguous and vague utterances

According to the model sketched in Section 2.1, calculating the interpretation of utterances involves calling on representations of the current visual context, the current user model, and general domain inference, in order to supply values to variables in presupposed constraints such as (1). Although all this information is essential for recognizing the user's intention, we find that deep domain inference—as embodied in constraints such as $\mathit{holds}(\mathit{now}, \mathit{fits\_plan}(A))$ and $\mathit{holds}(\mathit{result}(A, \mathit{now}), \mathit{size}(X, \mathit{SN}))$ that link interpretation to plan-recognition and causal inference—plays a surprisingly pervasive and powerful role in the process. We catalogue some of the effects of domain inference in our application using the examples in (2).

(2) a. Make the circle an x.
    b. Put a circle below the eyes.
    c. Lower the circle below the eyes.

Domain inference contributes to:

- *Identifying objects under ambiguous descriptions.* To interpret (2a), the system must find a figure part that is currently rendered as a circle but that could be rendered as an x instead. Not every circle in the figure works; for

[1] *Make* contributes (1a), requiring A to be a natural and contextually appropriate action that affects object X and brings about a situation in which X is visible; *the face* contributes (1b), requiring that the object X must currently be a single face; *bigger* contributes (1c), requiring that action A must result in a new size SN for X that falls in a region R including all measures larger than the current size SO of X.

example, you cannot draw an x for the circle that represents the silhouette of the whole head. This fact is part of the domain knowledge that speakers appear to assume in producing terse utterances like (2a). In interpretation, this constraint helps zero in on the object that the user had in mind, even when no noun phrase in the user's utterance completely identifies it.

- *Selecting quantitative parameters for action in response to vague requests.* In (2b), domain reasoning allows the system to recognize that the circle here is meant to serve as the mouth of the figure. That gives a new constraint on where to draw the circle that narrows down the vague spatial placement indicated by the user, *below the eyes.*
- *Resolving syntactic ambiguity.* We can rule out an analysis with no domain-specific interpretation. For example, in (2c), the PP *below the eyes* may be analyzed as an NP modifier specifying the present location of the circle or as a VP modifier specifying the final location of the circle. The VP modifier reading may be discarded as inapplicable if every object either already lies below the eyes (the mouth) or cannot be moved there (the eyes, the silhouette). The importance of context for disambiguation is well known; Schuler [12] provides a recent case study. Here we observe further that on-the-fly domain *inference* may be needed to inform parsing decisions, in addition to a precomputed domain-specific contextual database.

2.3 Incremental interpretation

In this paper we are particularly concerned with applying the domain inference described in Section 2.2 *incrementally*, to compute interpretations for partial utterances. Our principal motivation is the desire to support more natural conversational interaction in FIGLET.

In real-time conversation, speakers frequently count on their interlocutors to understand partial utterances. Indeed, speakers frequently act as if their interlocutors should be able to actively collaborate in helping to complete them. To give just one example, (3a) presents a "trial noun phrase" that Clark and Wilkes-Gibbs [4, pp. 466–467] observed in the human-human dialogues they studied. The trial noun phrase is marked by rising intonation, and constitutes an explicit request for the hearer to indicate, *before* the speaker continues with the utterance, whether the noun phrase has been understood.

(3) a. S: Okay now, the small blue cap we talked about before?

b. J: Yeah.

c. S: Put that over the hole on the side of that tube...

Other similar phenomena include the use of expanded noun phrases, installment noun phrases, and proxy noun phrases; the formulation and interpretation of self-repair; and the concurrent feedback speakers expect with backchannel items like *okay* and *mm-hmm.* See Clark and Wilkes-Gibbs [4], Milward and Cooper [10] or Allen, Ferguson and Stent [1] for more detailed discussion of this fine-grained interactivity. These conversational phenomena suggest that a language understanding system aspiring to natural, real-time dialogue with human users will not only need domain inference at some stage of interpretation, but that it will need to call on domain inference on provisional constituents. Such problems can be approached naturally within an incremental interpretation strategy such as the one we develop in this paper.

In general, the interactivity of natural dialogue is only one of many motivations for incremental interpretation. Incrementality may improve system performance even when the system does not act incrementally. For example, since users formulate utterances incrementally, partial utterances may be available for a substantial amount of time—enough to get much of the work of interpretation done in some applications. In such cases, an incremental interpretation strategy may allow the system to respond more quickly, by minimizing the delay between the time the user finishes and the time the utterance is interpreted.

Similarly, in certain applications, bringing context to bear on parsing decisions may dramatically decrease total interpretation time by ruling out, at an early stage, many analyses which are linguistically possible but contextually unsupported. Extreme cases of syntactic ambiguity include sentences such as (4), attributed to Stabler by Milward and Cooper [10, (1)].

(4) I put the bouquet of flowers that you gave me for Mothers' Day in the vase that you gave me for my birthday on the chest of drawers that you gave me for Armistice Day.

Attachment ambiguities give this sentence 4,862 distinct syntactic analyses. Incremental interpretation is one strategy an application faced with widespread ambiguity could take, in an effort to defuse potential combinatorial interactions among ambiguities early on. See also Haddock [7]. [2]

Finally, of course, incrementality is an important aspect of human sentence processing; for example, Brown-Schmidt, Campana and Tanenhaus [3] offer new evidence for incrementality from their investigations of spontaneous dialogue. We do not suppose that human incrementality in itself argues that natural language systems should therefore be incremental, too. (This supposition is sometimes made, however; see Haddock's introductory discussion in [7], for example.) Nevertheless, a *cognitive model* of human sentence processing *will* have to account for this incrementality in computational terms, and our work may prove relevant to this scientific project.

For the present purpose of exploring the use of domain inference in incremental interpretation, it is not crucial what factor compels the adoption of an incremental interpretation strategy in a particular application.

[2] While FIGLET must be prepared to handle mildly ambiguous utterances, such as (2c), syntactic ambiguity does not at present necessitate an incremental interpretation strategy.

2.4 Modularity and Inference in Interpretation

Calculating the interpretations as described in Sections 2.1 and 2.2 might seem like a straightforward application for off-the-shelf constraint programming, as in Oz [5]. These techniques assume that the solution instances for individual constraints can be tabulated efficiently and supplied as input to the constraint solver. The solver then manages the combinatorial interactions that arise in reconciling the tables into consistent overall solutions.

Such constraint solutions can even be maintained and updated in tandem with grammatical analysis, as described by Schuler [12]. On this strategy, upon encountering the word *make*, the presuppositions it contributes, (1a), are immediately solved, yielding a set $\{(A_i, X_i)\}$ where each (A_i, X_i) is a pair of domain representations that satisfies the presuppositions. Subsequent words in the utterance trigger their own, independent search problems. In building up the syntactic structure of the complete utterance, incompatible solutions are weeded out, so that when the complete syntactic structure of the utterance is finally derived, only complete utterance interpretations remain.

In fact, however, such techniques cannot be applied to find instances for interpretive constraints such as (1). In practice, these relations are impossible to tabulate. Our rich ontology induces relations of high arity over large domains. Moreover, these relations are frequently intensional or hypothetical, and hence cannot be closely circumscribed by the representation of the figure. The semantics of *bigger* in (1c) illustrates the difficulties. In general, *bigger* describes the sizes of two objects in two situations:

(5) $\mathit{holds}(\mathit{TO}, \mathit{size}(\mathit{XO}, \mathit{SO})) \wedge \mathit{region}(R, \mathit{bigger\_than}, \mathit{SO}) \wedge$
$\mathit{in\_region}(\mathit{SN}, R) \wedge \mathit{holds}(\mathit{TN}, \mathit{size}(\mathit{XN}, \mathit{SN}))$

The first object XO provides its size SO in situation TO as a reference, and this size is compared with the size of the second object XN in the second situation TN. This is a common pattern shared by relational vocabulary, including prepositions (such as *in*) and relational spatial adjectives (such as *left*). Based on the syntactic (and pragmatic) context in which these words are used, these words could describe one or two specific objects, and could describe neither, either, or both sizes hypothetically. For example, the syntax of *Make the face bigger* determines that $\mathit{XO} = \mathit{XN}$, that TO is *now* but TN is hypothetical. The syntax of *Move the bigger face up*, on the other hand, allows XO and XN to differ but requires that $\mathit{TO} = \mathit{TN} =$ now. In this case there are two different faces, one of which is currently bigger than the other. We have to reason in terms of varying objects and situations to get the semantics of instructions right.

It is impossible to match such semantics arbitrarily against the context, even in the simplest applications. For example, in a typical situation, FIGLET can perform about 35 primitive actions (move the eyes, move the mouth, resize the face, etc.), and there are 127 distinct nonempty subsets of individuals on the screen. To tabulate instances for a word like *bigger* as formalized in (5) would require considering some 1,225 real and hypothetical pairs of situations, and 16,000 pairs of referents in each. Even the component relations in (5) get unwieldy in these simple contexts; for example, they may involve computing binary relations over thousands of hypothetical sizes, indexed functionally by object and situation. In our implementation, we have been simply unable to tabulate lexical interpretations within our available time and memory.

We have chosen instead to implement domain inference by simulation. Relationships of the form $\mathit{holds}(\mathit{result}(A, \mathit{now}), P)$ are assessed by operational specification transforming the current state as dictated by action A, and checking whether P holds in the resulting representation. Thus, although we support a knowledge-level analysis of inference in interpretation in terms of constraint-satisfaction, we emphasize that for implementation, inference in interpretation is a matter of *problem-solving*. The understanding process must therefore formulate specific, constrained tasks for domain reasoning at appropriate stages in interpretation.

3 Implementing incremental interpretation

3.1 Incremental understanding

Language understanding in FIGLET is mediated by a bottom-up chart parser written in Prolog. As usual, chart edges indicate the presence of recognized partial constituents within the input sequence. In addition, edges now interface with the domain problem-solving required for understanding. Edges include a set of candidate interpretations; each candidate is represented along with the status of the ongoing domain problem-solving associated with it. Specifically, as in Schuler's work [12], candidate interpretations include an instantiation of discourse anaphors to discourse referents that meet presupposed constraints; this summarizes information about objects from completed problem-solving. Moreover, our interpretations include lists of real constraints, represented symbolically. The real constraints formalize the metric and spatial relationships that have been inferred for this interpretation from domain problem-solving. Finally, interpretations include lists of *delayed* presuppositions. The delayed presuppositions must ultimately be derived by domain inference to construct a completed interpretation, but may not yet be sufficiently specified for this inference to be tractable.

We present pseudocode for our understanding algorithm in Figure 2. Formally, we use the notation $c : i \to j$ to indicate the chart edge of syntactic category c from word i to word j. The edge stores a set of tuples $\{(v, r, p)\}$ where v is an assignment of values to variables, r is a list of real constraints and p is a list of unsolved constraints for domain problem-solving. When using a packed interpretation chart [12], we can further associate each tuple with a backward index recording how the tuple has been derived.

As usual, we begin by accessing constituents from the chart and putting

```
To populate edge $c : i \rightarrow j$ {
    Search over $k$ between $i$ and $j$
        Search over $(v_1, r_1, p_1) \in c' : i \rightarrow k$
            Search over $(v_2, r_2, p_2) \in c'' : k \rightarrow j$
                $\sigma \leftarrow \textsc{combine}(c', c'', c)$ – or fail
                $(v_3, r_3, p_3) \leftarrow (v_1\sigma + v_2\sigma, r_1\sigma + r_2\sigma, p_1\sigma + p_2\sigma)$ – or fail
                Search over $(v_4, r_4, p) \in \textsc{solve}(v_3, r_3, p_3)$
                    $(v, r) \leftarrow \textsc{simplify}(v_4, r_4, p, c)$
                    Add $(v, r, p)$ to $c : i \rightarrow j$
    }
```

Fig. 2. Algorithm for constructing chart entries with incremental interpretation and flexible domain inference.

them together by syntactic operations. Syntax is implemented by the function COMBINE, which determines whether adjacent constituents of category c' and c'' can be put together to make a constituent of category c. When it succeeds, COMBINE will return a substitution σ which must be applied to relate syntax and semantics in the overall constituent c to that of its subconstituents. This substitution may not be possible, because σ may equate variables that have been assigned incompatible values in subconstituents (as specified explicitly in the variable assignments or implicitly in the real constraints).

We encapsulate the interface between interpretation and domain inference by a function SOLVE; SOLVE(v, r, p) calls a domain-specific problem-solver as appropriate to make progress on the open problem-solving p given the existing assignment v and constraints r. Since this problem-solving may derive alternative candidate interpretations, SOLVE returns a *set* of new tuples (v', r', p') where p has been partially solved; each amounts to a new assignment v' that extends v, a new list of constraints r' that extends r, and a subproblem p' that represents the unsolved part of the domain problem p. Finally, we note that it may be possible to simplify the representation of a candidate interpretation before adding it back into the chart. For example, we can eliminate a variable (and its associated real constraints) if the variable cannot be constrained by further syntactic modification, is not subject to outstanding domain problem-solving, and is not required to express real constraints on other variables.

The important feature of our implementation is that the SOLVE procedure provides an interface where domain reasoning can be staged at the point in interpretation where it can be applied most effectively. As we have argued in Section 2, this is important because interlocutors in dialogue expect some incrementality, but complete word-by-word incrementality will prove prohibitively expensive. Thus, in practice, it is the complexity of domain inference that motivates the algorithm. Nevertheless, it is instructive to consider the overall complexity of understanding, abstracting away from domain inference. Suppose we adopt Schuler's assumptions [12], that domain reasoning is completed at the lexical level and yields discrete alternatives. Then each edge will have empty r and p; we can therefore simplify assignments v by pruning them down to the semantic arguments of the headword of each constituent—a fixed length. Thus, each chart entry's size is independent of the length of the sentence, and the algorithm has asymptotic space and time complexity $O(n^3)$ as a function of the input length n. By contrast, suppose that SOLVE *always* delays the constraints associated with incomplete utterances. Then each interpretation accumulates a semantic analysis of the entire constituent as its problem p. In the worst case, since utterance interpretations record assignments to $O(n)$ variables, the number of explicitly-represented alternative interpretations grows exponentially. Furthermore, since we must call on problem-solving for each semantic analysis independently, the time required for interpretation grows hyperexponentially. This may be prohibitive. But if it is, the problem is inherent in the model of domain inference. Delaying incomplete problems means that the only way to do semantic disambiguation is with complete logical forms; searching these logical forms exhaustively will create combinatorial problems even with a packed chart.

3.2 Inference strategy

In FIGLET, we implemented SOLVE by using a programmer-specified regime to manage problems for domain inference during interpretation. This regime takes the form of *delay rules* and *proof-order rules*, which determine which constraints can be solved and what order to solve them in.

Proof-order rules attempt to restrict the size of the explored search space by ordering the proofs of a lexical item's presuppositions in an attempt to minimize the branching that occurs as each presupposition is considered. The left-to-right orders of the presuppositions in (1a-c) reflect the proof-order rules currently defined in FIGLET. In finding solutions to (1a), for example, FIGLET first looks for ways to satisfy $simple(A)$, then for ways to satisfy $target(A, X)$ as well, and so on. Thus, only simple actions are checked against the plan, and the visibility of objects is only considered in situations resulting from actions that fit the plan, and only for objects targeted by actions that fit the plan.

An alternate, less effective proof-order might solve the presuppositions in the reverse order—right-to-left as they are written in (1a). In this case search would begin by finding all subsets of visible objects in situations resulting from known actions, proceed by weeding out actions that don't fit the plan and subsets of objects not targeted by actions that fit the plan, and then finally conclude by filtering out non-simple actions. This alternate proof-order is less efficient *in typical* FIGLET *scenarios* because it requires more reasoning about irrelevant entities: there are generally many irrelevant situations, many sets of visible objects not targeted by actions that fit the plan, and many actions that fit the plan that are not simple.[3] We have tuned our proof-order rules

[3] If a simple action is not found during interpretation, we relax our notion of simplicity and reinterpret. In this way, more abstruse actions (e.g., move the right eye and resize the

to work well in typical FIGLET scenarios.

While proof-order rules provide control over domain inference within the interpretation of a single lexical item's presuppositions, we have found a further need to define a problem-solving strategy *across* the lexical items that make up an utterance. We use delay rules for this purpose. Delay rules identify, in terms of the current state of instantiation of certain variables, presuppositions which are particularly expensive to solve. The expectation is that subsequent interpretation of other, better constrained presuppositions will eventually make solving the delayed presuppositions less expensive.

Most of the expense of incremental interpretation in FIGLET is associated with simulating actions, which involves consulting domain knowledge about how to actually carry out each action (for example, what happens to the parts of the face when the entire face is resized) and invoking a linear real constraint solver in order to model the interaction between real constraints contributed linguistically and real constraints encoded as background knowledge. Action variables, such as A in (1a), are instantiated to fully grounded domain representations in two steps. First, a proof of $\mathit{fits\_plan}(A)$ binds A to a schematic term representing a primitive action or short sequence of primitive actions targeted at particular individuals. Later, the schematic term may become fully grounded through a simulation of the action. (For brevity, the instantiation associated with simulation is omitted from Figure 3.) The delay rules are designed to minimize the number of times this latter step occurs. They are:

(6) a. Delay $\mathit{fits\_plan}(A)$ if A's target is uninstantiated.

b. Delay simulation of A if A is not yet schematized by $\mathit{fits\_plan}(A)$.

c. Delay $\mathit{holds}(\mathit{result}(A, \mathit{now}), P)$ if A is not yet simulated.

The combined effect of the delay rules is to postpone simulation-based reasoning about presuppositions of the form $\mathit{holds}(\mathit{result}(A, \mathit{now}), P)$, which generally characterize the desired effects of actions, until the targets of potential actions have been identified, and until off-topic actions have been filtered out.

Let's reconsider the interpretation of (1a-c) in light of these delay rules. The chart for *Make the face bigger* is illustrated in Figure 3. In interpreting *Make*, $\mathit{simple}(A)$ and $\mathit{target}(A, X)$ are proved immediately. They each schematize A, but note that their proofs are context-independent and carried out in constant time. For example, a proof of $\mathit{target}(A, X)$ provides only an abstract link between A and X; X remains uninstantiated until some later proof binds it to some domain representation. By contrast, $\mathit{holds}(\mathit{now}, \mathit{fits\_plan}(A))$ and $\mathit{holds}(\mathit{result}(A, \mathit{now}), \mathit{visible}(X))$ are delayed by delay rules (6a) and (6c), respectively. In interpreting *face*, $\mathit{number}(X, 1)$ is proved, schematizing X to some singleton set. Since S is not instantiated until *the face* is constructed, $\mathit{holds}(S, \mathit{type}(X, \mathit{face}))$ is delayed until then by delay rule (6c). In interpret-

mouth) are possible, but simpler actions are preferred. Both simple and non-simple actions are considered to fit the plan at all times.

Edge span	*Make*
Presups	$\mathit{simple}(A) \wedge \mathit{target}(A, X) \wedge \mathit{holds}(\mathit{now}, \mathit{fits\_plan}(A)) \wedge \mathit{holds}(\mathit{result}(A, \mathit{now}), \mathit{visible}(X))$
Asserts	$\mathit{do}(A)$
Interps	$(\{A \leftarrow \mathit{change}(X, \_)\}, \{\}, < \mathit{holds}(\mathit{now}, \mathit{fits\_plan}(A)), \mathit{holds}(\mathit{result}(A, \mathit{now}), \mathit{visible}(X)) >)$

Edge span	*the*
Presups	
Asserts	
Interps	

Edge span	*face*
Presups	$\mathit{number}(X, 1) \wedge \mathit{holds}(S, \mathit{type}(X, \mathit{face}))$
Asserts	
Interps	$(\{X \leftarrow \{\_\}\}, \{\}, < \mathit{holds}(S, \mathit{type}(X, \mathit{face})) >)$

Edge span	*bigger*
Presups	$\mathit{holds}(TO, \mathit{size}(XO, SO)) \wedge \mathit{region}(R, \mathit{bigger\_than}, SO) \wedge \mathit{in\_region}(SN, R) \wedge \mathit{holds}(TN, \mathit{size}(XN, SN))$
Asserts	
Interps	$(\{SO \leftarrow (WO, HO), R \leftarrow ((WO, HO), (1, 1)), SN \leftarrow (WN, HN)\},$ $\{WN > WO, WN < 1, HN > HO, HN < 1\},$ $< \mathit{holds}(TO, \mathit{size}(XO, SO)), \mathit{holds}(TN, \mathit{size}(XN, SN)) >)$

Edge span	*the face*
Presups	$\mathit{number}(X, 1) \wedge \mathit{holds}(\mathit{now}, \mathit{type}(X, \mathit{face}))$
Asserts	
Interps	$(\{X \leftarrow \{face3\}\}, \{\}, <>)$

Edge span	*Make the face*
Presups	$\mathit{simple}(A) \wedge \mathit{target}(A, X) \wedge \mathit{holds}(\mathit{now}, \mathit{fits\_plan}(A)) \wedge \mathit{holds}(\mathit{result}(A, \mathit{now}), \mathit{visible}(X)) \wedge$ $\mathit{number}(X, 1) \wedge \mathit{holds}(\mathit{now}, \mathit{type}(X, \mathit{face}))$
Asserts	$\mathit{do}(A)$
Interps	$(\{A \leftarrow \mathit{change}(X, \mathit{location}((XN, YN))), X \leftarrow \{face3\}\}, \{\mathcal{R}_{Loc}(XN, YN)\}, <>),$ $(\{A \leftarrow \mathit{change}(X, \mathit{size}((WN, HN))), X \leftarrow \{face3\}\}, \{\mathcal{R}_{Size}(WN, HN)\}, <>),$ $(\{A \leftarrow \mathit{change}(X, \mathit{shape}(SN)), X \leftarrow \{face3\}\}, \{\mathcal{R}_{Shape}(SN)\}, <>),$ $(\{A \leftarrow \mathit{change}(X, \mathit{color}(CN)), X \leftarrow \{face3\}\}, \{\mathcal{R}_{Color}(CN)\}, <>),$

Edge span	*Make the face bigger*
Presups	$\mathit{simple}(A) \wedge \mathit{target}(A, X) \wedge \mathit{holds}(\mathit{now}, \mathit{fits\_plan}(A)) \wedge \mathit{holds}(\mathit{result}(A, \mathit{now}), \mathit{visible}(X)) \wedge$ $\mathit{number}(X, 1) \wedge \mathit{holds}(\mathit{now}, \mathit{type}(X, \mathit{face})) \wedge$ $\mathit{holds}(\mathit{now}, \mathit{size}(X, SO)) \wedge \mathit{region}(R, \mathit{bigger\_than}, SO) \wedge \mathit{in\_region}(SN, R) \wedge$ $\mathit{holds}(\mathit{result}(A, \mathit{now}), \mathit{size}(X, SN))$
Asserts	$\mathit{do}(A)$
Interps	$(\{A \leftarrow \mathit{change}(X, \mathit{size}((WN, HN))), X \leftarrow \{face3\},$ $SO \leftarrow (0.6, 0.7), R \leftarrow ((0.6, 0.7), (1, 1)), SN \leftarrow (WN, HN)\},$ $\{WN > 0.6, WN < 1, HN > 0.7, HN < 1, \mathcal{R}_{Size}(WN, HN)\},$ $<>)$

Fig. 3. The chart constructed during incremental parsing of *Make the face bigger*. $\mathcal{R}_{Property}$ indicates constraints on values for *Property* inferred from domain problem-solving. Note that presuppositions and assertions need not be stored with each edge; here they highlight the information from which edge interpretations have been computed.

ing *bigger*, the region constraints are proved immediately and yield real constraints. The other presuppositions of *bigger* are delayed according to delay rule (6c).

Subsequently, as larger syntactic structures are constructed, all the delayed presuppositions are eventually proved for certain restricted values of the variables they contain. For example, when *Make* and *the face* are combined, in constructing the incomplete sentential constituent *Make the face*, the

	O=H/D=H	**O=R/D=H**	**O=H/D=R**	**O=R/D=R**
CI, mean	631	1240	4007	7343
CI, std	0	185	2099	2684
time, mean	3.6	3.8	57.5	89.5
time, std	0.5	0.4	20.9	29.3

Fig. 4. For four different problem-solving strategies, the total number of domain constraint instances (CI) returned to the language understanding module, and the elapsed time spent processing the complete interaction, in seconds.

delayed $holds(now, \mathit{fits\_plan}(A))$ and $holds(result(A, now), visible(X))$ presuppositions are proved. Proving the latter presupposition causes the simulation of several actions at this step. However, in conjunction with delay rule (6b), only actions fitting the plan and targeting faces are ever simulated.

3.3 Performance analysis

To quantify the effect of problem-solving strategy on interpretation, we constructed four versions of the system. All versions of the system use the same constraints and the same domain inference mechanism, so all versions come up with the same interpretations. The only difference is performance. The systems differ in whether they used our handbuilt proof order (**O=H**) or random proof order (**O=R**, meaning that the order of constraints in lexical presuppositions is scrambled when a word edge is initially added to the chart). And they differ in whether they implement our handbuilt delay rules (**D=H**) or make random decisions to delay constraints for domain inference (**D=R**, which we tuned to delay with probability 0.44, after finding that the handbuilt delay mechanism triggered delay 44% of the time).

We presented each version of the system with the instruction sequence of Figure 1 ten times. The results are presented in Figure 4. The number of constraint instances solved shows clear differences across conditions (one-way ANOVA, p<0.0001) and between condition pairs (by two-tailed t-tests, p<0.05 or less). The delay rules enforce rather strong constraints on what domain problems are posed, and cut the domain constraint instances required for interpretation roughly by a factor of six. The further halving achieved by our handbuilt proof order is comparatively small.

The performance data likewise show clear differences across conditions (one-way ANOVA, p<0.0001). Indeed, we found a strong overall correlation between elapsed interaction time and constraint instances solved (r=0.88, p<0.0001). Notice that processing the five instructions of Figure 1 required nearly 90 seconds on the random incremental interpretation strategy (**O=R/D=R**), while our handbuilt system achieved the same results in under 4 seconds. Such data show that supporting real-time interactions in domains where inference is expensive demands a flexible inference strategy.

4 A methodological challenge

In this paper, we have argued that deep domain reasoning, including causal inference and plan recognition, should guide natural language understanding, and we have described an effective implementation of this idea. Our implementation casts domain inference as a modular and potentially expensive problem-solving process. This domain inference proceeds flexibly, and provisional interpretations contain records of the developing problem-solving state. Domain inference informs interpretive choices whenever sufficient constraints become available to yield meaningful conclusions. But otherwise, when constraints are insufficient, we prefer to avoid domain inference altogether.

Our approach to implementation has been to specify the problem-solving interface between language understanding and domain reasoning by hand. Although this is a flexible approach that offers fine-grained control, we feel that such programming is too expensive to pursue in most cases. The strategy requires painstaking effort from programmers, who must come to understand what problems are tightly constrained, what problems are underconstrained, and what problems are simply infeasible. Yet the results are fragile to changes in domain representation and reasoning. Our experience is that scalable inference should rest on general abstractions, not meticulous analysis.

Two such abstractions suggest themselves as prospects for future work. One is linguistic. The patterns of description and constituency that a speaker has chosen to realize their thought may provide a close guide to the inferential effort of the system to recognize it. Concretely, if a speaker wants you to identify some discourse entity which is explicitly or implicitly referenced in an utterance, you can reasonably expect her to provide you with a specific constituent that can be used to select a small selection of candidate referents, if not the exact referent itself. Before such a constituent completely arrives, you could delay problem-solving involving that referent. The challenge here is to extend this intuitively appealing strategy to the rich ontology we actually find in linking meaning and interpretation.

Another possibility is empirical. The system could optimize its inferential strategy based on a learned model of the costs and outcomes of inference from prior linguistic experience. For example, chart edges could be assigned a score, computed from readily available, relatively shallow features, in advance of inference, indicating both how likely the edge is to yield the right interpretation, and how much work it would be for the system to derive that interpretation. Inference mechanisms could then work on the best-scoring constituents first, and thereby focus their effort on small and useful problems. The challenge here is to find models of the costs and outcomes of inference that generalize effectively and require only modest quantities of training data.

Since each of these approaches has its potential strengths and weaknesses, we hope in future work to implement and investigate both, providing a broader and more detailed guide to future implementations.

References

[1] Allen, J., G. Ferguson and A. Stent, *An architecture for more realistic conversational systems*, in: *Proceedings of Intelligent User Interfaces (IUI)*, 2001.

[2] Bos, J. and T. Oka, *An inference-based approach to dialogue system design*, in: *COLING*, 2002, pp. 113–119.

[3] Brown-Schmidt, S., E. Campana and M. K. Tanenhaus, *Reference resolution in the wild: On-line circumscription of referential domains in a natural interactive problem-solving task*, in: *Proceedings of the Cognitive Science Society*, 2002, pp. 148–153.

[4] Clark, H. H. and D. Wilkes-Gibbs, *Referring as a collaborative process*, in: P. R. Cohen, J. Morgan and M. E. Pollack, editors, *Intentions in Communication*, MIT, 1990 pp. 463–493.

[5] Duchier, D., C. Gardent and J. Niehren, "Concurrent Constraint Programming in Oz for Natural Language Processing," Programming Systems Lab, Universität des Saarlandes, Germany, 1998.

[6] Gabsdil, M., A. Koller and K. Striegnitz, *Natural language and inference in a computer game*, in: *COLING*, 2002.

[7] Haddock, N. J., *Computational models of incremental semantic interpretation*, Language and Cognitive Processes **4** (1989), pp. 337–368.

[8] Lesh, N., C. Rich and C. L. Sidner, *Collaborating with focused and unfocused users under imperfect communication*, in: *International Conference on User Modeling (UM)*, 2001, pp. 63–74.

[9] Levesque, H. J., *Foundations of a functional approach to knowledge representation*, Artificial Intelligence **23** (1984), pp. 155–212.

[10] Milward, D. and R. Cooper, *Incremental interpretation: Applications, theory, and relationship to dynamic semantics*, in: *COLING*, 1994, pp. 748–754.

[11] Rich, C., C. L. Sidner and N. Lesh, *COLLAGEN: applying collaborative discourse theory to human-computer interaction*, AI Magazine **22** (2001), pp. 15–25.

[12] Schuler, W., *Computational properties of environment-based disambiguation*, in: *Proceedings of ACL*, 2001, pp. 466–473.

[13] Stone, M., *Knowledge representation for language engineering*, in: A. Farghaly, editor, *A Handbook for Language Engineers*, CSLI, 2003 .

[14] Stone, M., *Linguistic representation and Gricean inference*, in: *International Workshop on Computational Semantics*, 2003, pp. 5–21.

[15] Yates, A., O. Etzioni and D. Weld, *A reliable natural language interface to household appliances*, in: *Proceedings of Intelligent User Interfaces (IUI)*, 2003.