UNDERSTANDING NATURAL LANGUAGE INSTRUCTIONS: THE CASE OF PURPOSE CLAUSES

Barbara Di Eugenio *
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA
dieugeni@linc.cis.upenn.edu

ABSTRACT

This paper presents an analysis of purpose clauses in the context of instruction understanding. Such analysis shows that goals affect the interpretation and / or execution of actions, lends support to the proposal of using generation and enablement to model relations between actions, and sheds light on some inference processes necessary to interpret purpose clauses.

INTRODUCTION

A speaker (S) gives instructions to a hearer (H) in order to affect H's behavior. Researchers including (Winograd, 1972), (Chapman, 1991), (Vere and Bickmore, 1990), (Cohen and Levesque, 1990), (Alterman *et al.*, 1991) have been and are addressing many complex facets of the problem of mapping Natural Language instructions onto an agent's behavior. However, an aspect that no one has really considered is *computing the objects* of the intentions H's adopts, namely, the actions to be performed. In general, researchers have equated such objects with logical forms extracted from the NL input. This is perhaps sufficient for simple positive imperatives, but more complex imperatives require that action descriptions be computed, not simply extracted, from the input instruction. To clarify my point, consider:

Ex. 1 *a) Place a plank between two ladders.*
b) Place a plank between two ladders to create a simple scaffold.

In both a) and b), the action to be executed is *place a plank between two ladders.* However, Ex. 1.a would be correctly interpreted by placing the plank *anywhere* between the two ladders: this shows that in b) H must be inferring the proper position for the plank from the expressed goal *to create a simple scaffold.* Therefore, the goal an action is meant to achieve constrains the interpretation and / or the execution of the action itself.

The infinitival sentence in Ex. 1.b is a *purpose clause*, which, as its name says, expresses the agent's purpose in performing a certain action. The analysis of purpose clauses is relevant to the problem of understanding Natural Language instructions, because:

1. Purpose clauses explicitly encode goals and their interpretation shows that the goals that H adopts guide his/her computation of the action(s) to perform.
2. Purpose clauses appear to express generation or enablement, supporting the proposal, made by (Allen, 1984), (Pollack, 1986), (Grosz and Sidner, 1990), (Balkanski, 1990), that these two relations are necessary to model actions.

After a general description of purpose clauses, I will concentrate on the relations between actions that they express, and on the inference processes that their interpretation requires. I see these inferences as instantiations of general *accommodation* processes necessary to interpret instructions, where the term *accommodation* is borrowed from (Lewis, 1979). I will conclude by describing the algorithm that implements the proposed inference processes.

PURPOSE CLAUSES

I am not the first one to analyze purpose clauses: however, they have received attention almost exclusively from a syntactic point of view – see for example (Jones, 1985), (Hegarty, 1990). Notice that I am not using the term *purpose clause* in the technical way it has been used in syntax, where it refers to infinitival *to* clauses adjoined to NPs. In contrast, the infinitival clauses I have concentrated on are adjoined to a matrix clause, and are termed *rational clauses* in syntax; in fact all the data I will discuss in this paper belong to a particular subclass of such clauses, subject-gap rational clauses.

As far as I know, very little attention has been paid to purpose clauses in the semantics literature: in (1990), Jackendoff briefly analyzes expressions of *purpose, goal, or rationale, normally encoded as an infinitival, in order*

*Mailing address: IRCS - 3401, Walnut St - Suite 400C - Philadelphia, PA, 19104 - USA.

to-phrase, or for-phrase. He represents them by means of a subordinating function FOR, which has the adjunct clause as an argument; in turn, FOR plus its argument is a restrictive modifier of the main clause. However, Jackendoff's semantic decomposition doesn't go beyond the construction of the logical form of a sentence, and he doesn't pursue the issue of what the relation between the actions described in the matrix and adjunct really is.

The only other work that mentions purpose clauses in a computational setting is (Balkanski, 1991). However, she doesn't present any linguistic analysis of the data; as I will show, such analysis raises many interesting issues, such as [1]:

• It is fairly clear that S uses purpose clauses to explain to H the goal β to whose achievement the execution of α contributes. However, an important point that had been overlooked so far is that the goal β also *constrains* the interpretation of α, as I observed with respect to Ex. 1.b. Another example in point is:

Ex. 2 *Cut the square in half to create two triangles.*

The action to be performed is *cutting the square in half*. However, such action description is underspecified, in that there is an infinite number of ways of cutting a square in half: the goal *create two triangles* restricts the choice to *cutting the square along one of the two diagonals*.

• Purpose clauses relate action descriptions at different levels of abstraction, such as a physical action and an abstract process, or two physical actions, but at different levels of granularity:

Ex. 3 *Heat on stove to simmer.*

• As far as what is described in purpose clauses, I have been implying that both matrix and purpose clauses describe an action, α and β respectively. There are rare cases – in fact, I found only one – in which one of the two clauses describes a state σ:

Ex. 4 *To be successfully covered, a wood wall must be flat and smooth.*

I haven't found any instances in which both matrix and purpose clauses describe a state. Intuitively, this makes sense because S uses a purpose clause to inform H of the purpose of a given *action* [2].

• In most cases, the goal β describes a change in the world. However, in some cases

1. The change is not in the world, but in H's knowledge. By executing α, H can change the state of his knowledge with respect to a certain proposition or to the value of a certain entity.

   Ex. 5 *You may want to hang a coordinating border around the room at the top of the walls.* **To determine the amount of border,** *measure the width (in feet) of all walls to be covered and divide by three. Since borders are sold by the yard, this will give you the number of yards needed.*

   Many of such examples involve verbs such as *check, make sure* etc. followed by a *that*-complement describing a state ϕ. The use of such verbs has the pragmatic effect that not only does H check whether ϕ holds, but, if ϕ doesn't hold, s/he will also do something so that ϕ comes to hold.

   Ex. 6 *To attach the wires to the new switch, use the paper clip to move the spring type clip aside and slip the wire into place. Tug gently on each wire* **to make sure it's secure.**

2. The purpose clause may inform H that the world should *not* change, namely, that a given event should be prevented from happening:

   Ex. 7 *Tape raw edges of fabric to prevent threads from raveling as you work.*

• From a discourse processing point of view, interpreting a purpose clause may affect the discourse model, in particular by introducing new referents. This happens when the effect of α is to create a new object, and β identifies it. Verbs frequently used in this context are *create, make, form* etc.

Ex. 8 *Join the short ends of the hat band to form a circle.*

Similarly, in Ex. 2 the discourse referents for the triangles created by cutting the square in half, and in Ex. 5 the referent for *amount of border* are introduced.

RELATIONS BETWEEN ACTIONS

So far, I have mentioned that α *contributes* to achieving the goal β. The notion of *contribution* can be made more specific by examining naturally occurring purpose clauses. In the majority of cases, they express *generation*, and in the rest *enablement*. Also (Grosz and Sidner, 1990) use *contribute* as a relation between actions, and they define it as *a place holder for any relation ... that can hold between actions when one can be said to contribute (for example, by generating or enabling) to the performance of the other*. However, they don't justify this in terms of naturally occurring data. Balkanski (1991) does mention that purpose clauses express generation or enablement, but she doesn't provide evidence to support this claim.

GENERATION

Generation is a relation between actions that has been extensively studied, first in philosophy (Goldman, 1970) and then in discourse analysis (Allen, 1984), (Pollack, 1986), (Grosz and Sidner, 1990), (Balkanski, 1990). According to Goldman, intuitively generation is the relation between actions conveyed by the preposition *by* in English – *turning on the light* **by** *flipping the switch*.

[1]I collected one hundred and one consecutive instances of purpose clauses from a how-to-do book on installing wall coverings, and from two craft magazines.

[2]There are clearly other ways of describing that a state is the goal of a certain action, for example by means of *so/such that*, but I won't deal with such data here.

More formally, we can say that an action α conditionally generates another action β iff [3]:

1. α and β are simultaneous;
2. α is not part of doing β (as in the case of *playing a C note* as part of *playing a C triad* on a piano);
3. when α occurs, a set of conditions C hold, such that the joint occurrence of α and C imply the occurrence of β. In the case of the generation relation between *flipping the switch* and *turning on the light*, C will include that the wire, the switch and the bulb are working.

Although generation doesn't hold between α and β if α is part of a sequence of actions $\mathcal{A}$ to do β, generation may hold between the whole sequence $\mathcal{A}$ and β.

Generation is a pervasive relation between action descriptions in naturally occurring data. However, it appears from my corpus that *by* clauses are used less frequently than purpose clauses to express generation [4]: about 95% of my 101 purpose clauses express generation, while in the same corpus there are only 27 *by* clauses. It does look like generation in instructional text is mainly expressed by means of purpose clauses. They may express either a direct generation relation between α and β, or an indirect generation relation between α and β, where by *indirect generation* I mean that α belongs to a sequence of actions $\mathcal{A}$ which generates β.

ENABLEMENT

Following first Pollack (1986) and then Balkanski (1990), *enablement* holds between two actions α and β if and only if an occurrence of α brings about a set of conditions that are necessary (but not necessarily sufficient) for the subsequent performance of β.

Only about 5% of my examples express enablement:

Ex. 9 *Unscrew the protective plate to expose the box.*

Unscrew the protective plate enables *taking the plate off* which generates *exposing the box.*

GENERATION AND ENABLEMENT IN MODELING ACTIONS

That purpose clauses do express generation and enablement is a welcome finding: these two relations have been proposed as necessary to model actions (Allen, 1984), (Pollack, 1986), (Grosz and Sidner, 1990), (Balkanski, 1990), but this proposal has not been justified by offering an extensive analysis of whether and how these relations are expressed in NL.

A further motivation for using *generation* and *enablement* in modeling actions is that they allow us to draw conclusions about action execution as well – a particularly useful consequence given that my work is taking place in the framework of the *Animation from Natural Language – AnimNL* project (Badler *et al.*, 1990; Webber *et al.*, 1991) in which the input instructions do have to be executed, namely, animated.

As has already been observed by other researchers, if α generates β, two actions are described, but only α, the generator, needs to be performed. In Ex. 2, there is no *creating* action per se that has to be executed: the physical action to be performed is *cutting*, constrained by the goal as explained above.

In contrast to generation, if α enables β, after executing α, β still needs to be executed: α has to temporally precede β, in the sense that α has to begin, but not necessarily end, before β. In Ex. 10, *hold* has to continue for the whole duration of *fill*:

Ex. 10 *Hold the cup under the spigot to fill it with coffee.*

Notice that, in the same way that the generatee affects the execution of the generator, so the enabled action affects the execution of the enabling action. Consider the difference in the interpretation of *to* in *go to the mirror*, depending upon whether the action to be enabled is *seeing oneself* or *carrying the mirror somewhere else*.

INFERENCE PROCESSES

So far, I have been talking about the purpose clause constraining the interpretation of the matrix clause. I will now provide some details on how such constraints are computed. The inferences that I have identified so far as necessary to interpret purpose clauses can be described as

1. Computing a more specific action description.
2. Computing assumptions that have to hold for a certain relation between actions to hold.

Computing more specific action descriptions. In Ex. 2 – *Cut the square in half to create two triangles* – it is necessary to find a more specific action α_1 which will achieve the goal specified by the purpose clause, as shown in Fig. 1.

For Ex. 2 we have β = *create two triangles*, α = *cut the square in half*, α_1 = *cut the square in half along the diagonal*. The reader will notice that the inputs to accommodation are linguistic expressions, while its outputs are predicate – argument structures: I have used the latter in Fig. 1 to indicate that accommodation infers relations between *action types*. However, as I will show later, the representation I adopt is not based on predicate – argument structures. Also notice that I am using Greek symbols for both linguistic expressions and action types: the context should be sufficient to disambiguate which one is meant.

Computing assumptions. Let's consider:

[3]Goldman distinguishes among four kinds of generation relations: subsequent work has been mainly influenced by *conditional* generation.

[4]Generation can also be expressed with a simple free adjunct; however, this use of free adjuncts is not very common – see (Webber and Di Eugenio, 1990).

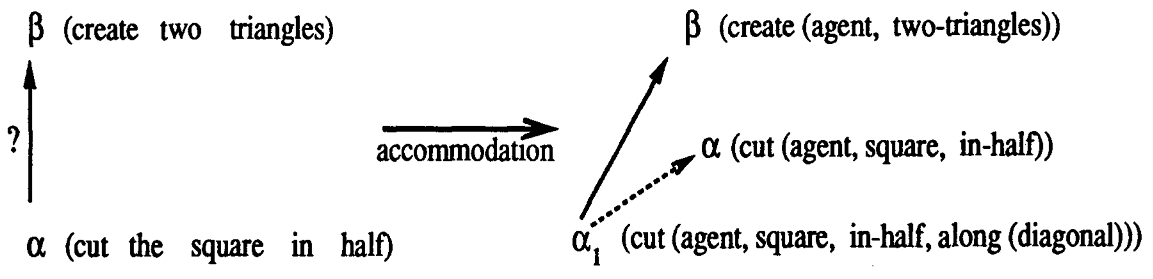
β (create two triangles)
?
α (cut the square in half)
accommodation
β (create (agent, two-triangles))
α (cut (agent, square, in-half))
α1 (cut (agent, square, in-half, along (diagonal)))

Figure 1: Schematic depiction of the first kind of accommodation

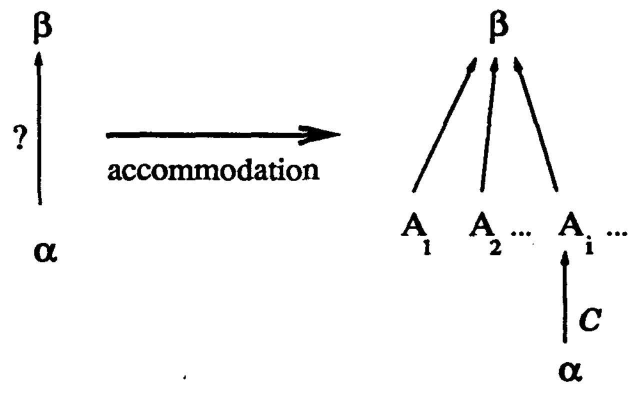

Figure 2: Schematic depiction of the second kind of accommodation

Ex. 11 *Go into the other room to get the urn of coffee.*

Presumably, H doesn't have a particular plan that deals with *getting an urn of coffee.* S/he will have a generic plan about *get x*, which s/he will adapt to the instructions S gives him [5]. In particular, H has to find the connection between *go into the other room* and *get the urn of coffee.* This connection requires reasoning about the effects of *go* with respect to the plan *get x*; notice that the (most direct) connection between these two actions requires the assumption that the referent of *the urn of coffee* is *in the other room.* Schematically, one could represent this kind of inference as in Fig. 2 – β is the goal, α the instruction to accommodate, A_k the actions belonging to the plan to achieve β, C the necessary assumptions.

It could happen that these two kinds of inference need to be combined: however, no example I have found so far requires it.

INTERPRETING *Do α to do β*

In this section, I will describe the algorithm that implements the two kinds of accommodation described in the previous section. Before doing that, I will make some remarks on the action representation I adopt and on the structure of the intentions – the *plan graph* – that my algorithm contributes to building.

[5] Actually H may have more than one single plan for *get x*, in which case *go into the other room* may in fact help to select the plan the instructor has in mind.

Action representation. To represent action types, I use an hybrid system (Brachman *et al.*, 1983), whose primitives are taken from Jackendoff's Conceptual Structures (1990); relations between action types are represented in another module of the system, the action library.

I'd like to spend a few words justifying the choice of an hybrid system: this choice is neither casual, nor determined by the characteristics of the *AnimNL* project.

Generally, in systems that deal with NL instructions, action types are represented as predicate – argument structures; the crucial assumption is then made that the logical form of an input instruction will exactly match one of these definitions. However, there is an infinite number of NL descriptions that correspond to a basic predicate – argument structure: just think of all the possible modifiers that can be added to a basic sentence containing only a verb and its arguments. Therefore it is necessary to have a flexible knowledge representation system that can help us understand the relation between the input description and the stored one. I claim that hybrid KR systems provide such flexibility, given their virtual lattice structure and the classification algorithm operating on the lattice: in the last section of this paper I will provide an example supporting my claim.

Space doesn't allow me to deal with the reason why Conceptual Structures are relevant, namely, that they are useful to compute assumptions. For further details, the interested reader is referred to (Di Eugenio, 1992; Di Eugenio and White, 1992).

Just a reminder to the reader that hybrid systems have two components: the *terminological* box, or T-Box, where concepts are defined, and on which the classification algorithm works by computing subsumption relations between different concepts. The algorithm is crucial for adding new concepts to the KB: it computes the subsumption relations between the new concept and all the other concepts in the lattice, so that it can "position" the new concept in the right place in the lattice. The other component of an hybrid system is the *assertional* box, or A-box, where assertions are stored, and which is equipped with a theorem-prover.

In my case, the T-Box contains knowledge about action types, while assertions about individual actions – instances of the types – are contained in the A-Box: such individuals correspond to the action descriptions contained in the input instructions [6].

The *action library* contains simple plans relating actions; *simple plans* are either generation or enablement relations between pairs: the first member of the pair is either a single action or a sequence of action, and the second member is an action. In case the first member of the pair is an individual action, I will talk about *direct* generation or enablement. For the moment, generation and enablement are represented in a way very similar to (Balkanski, 1990).

The plan graph represents the structure of the intentions derived from the input instructions. It is composed of nodes that contain descriptions of actions, and arcs that denote relations between them. A node contains the Conceptual Structures representation of an action, augmented with the consequent state achieved after the execution of that action. The arcs represent, among others: temporal relations; generation; enablement.

The plan graph is built by an interpretation algorithm that works by keeping track of *active nodes*, which for the moment include the goal currently in focus and the nodes just added to the graph; it is manipulated by various inference processes, such as plan expansion, and plan recognition.

My algorithm is described in Fig. 3 [7]. Clearly the inferences I describe are possible only because I rely on the other *AnimNL* modules for 1) parsing the input and providing a logical form expressed in terms of Conceptual Structures primitives; 2) managing the discourse model, solving anaphora, performing temporal inferences etc (Webber *et al.*, 1991).

AN EXAMPLE OF THE ALGORITHM

I will conclude by showing how step 4a in Fig. 3 takes advantage of the classification algorithm with which hybrid systems are equipped.

Consider the T-Box, or better said, the portion of T-Box shown in Fig. 4 [8].

Given Ex. 2 – *Cut the square in half to create two triangles* – as input, the individual action description *cut (the) square in half* will be asserted in the A-Box and recognized as an instance of α – the shaded concept *cut (a) square in half* – which is a descendant of *cut* and an abstraction of ω – *cut (a) square in half along the diagonal*, as shown in Fig. 5 [9]. Notice that this does *not* imply that the concept *cut (a) square in half* is known beforehand: the classification process is able to recognize it as a virtual concept and to find the right place for it in the lattice [10]. Given that α is ancestor of ω, and that ω generates β – *create two triangles*, the fact that the action to be performed is actually ω and not α can be inferred. This implements step 4(a)ii.

The classification process can also help to deal with cases in which α is in conflict with ω – step 4(a)iv. If α were *cut (a) square along a perpendicular axis*, a conflict with ω – *cut (a) square in half along the diagonal* – would be recognized. Given the T-Box in fig. 4, the classification process would result in α being a sister to ω: my algorithm would try to unify them, but this would not be possible, because the role fillers of *location* on α and ω cannot be unified, being *along(perpendicular-axis)* and *along(diagonal)* respectively. I haven't addressed the issue yet of which strategies to adopt in case such a conflict is detected.

Another point left for future work is what to do when step 2 yields more than one simple plan.

The knowledge representation system I am using is BACK (Peltason *et al.*, 1989); the algorithm is being implemented in QUINTUS PROLOG.

[6]Notice that these individuals are simply instances of generic concepts, and not necessarily action tokens, namely, nothing is asserted with regard to their happening in the world.

[7]As I mentioned earlier in the paper, the Greek symbols refer both to input descriptions and to action types.

[8]The reader may find that the representation in Fig. 4 is not very perspicuous, as it mixes linguistic expressions, such as *along(diagonal)*, with conceptual knowledge about entities. Actually, roles and concepts are expressed in terms of Conceptual Structures primitives, which provide a uniform way of representing knowledge apparently belonging to different types. However, a T-Box expressed in terms of Conceptual Structures becomes very complex, so in Fig. 4 I adopted a more readable representation.

[9]The agent role does not appear on *cut square in half* in the A-Box for the sake of readability.

[10]In fact, such concept is not really added to the lattice.

Input: the Conceptual Structures logical forms for α and β, the current plan graph, and the list of active nodes.

1. Add to A-Box individuals corresponding to the two logical forms. Set flag ACCOM if they don't exactly match known concepts.
2. Retrieve from the action library the simple plan(s) associated with β – generation relations in which β is the generatee, enablement relations in which β is the enablee.
3. If ACCOM is not set
   (a) If there is a *direct* generation or enablement relation between α and β, augment plan graph with the structure derived from it, after calling `compute-assumptions`.
   (b) If there is no such direct relation, recursively look for possible connections between α and the components γ_i of sequences that either generate or enable β.
   Augment plan graph, after calling `compute-assumptions`.
4. If ACCOM is set,
   (a) If there is ω such that ω directly generates or enables β, check whether
      i. ω is an ancestor of α: take α as the intended action.
      ii. ω is a descendant of α: take ω as the intended action.
      iii. If ω and α are not ancestors of each other, but they can be unified – all the information they provide is compatible, as in the case of *cut square in half along diagonal* and *cut square carefully* – then their unification $\omega \sqcup \alpha$ is the action to be executed.
      iv. If ω and α are not ancestors of each other, and provide conflicting information – such as *cut square along diagonal* and *cut square along perpendicular axis* – then signal *failure*.
   (b) If there is no such ω, look for possible connections between α and the components γ_i of sequences that either generate or enable β, as in step 3b. Given that α is not known to the system, apply the inferences described in 4a to α and γ_i.

Figure 3: The algorithm for *Do α to do β*

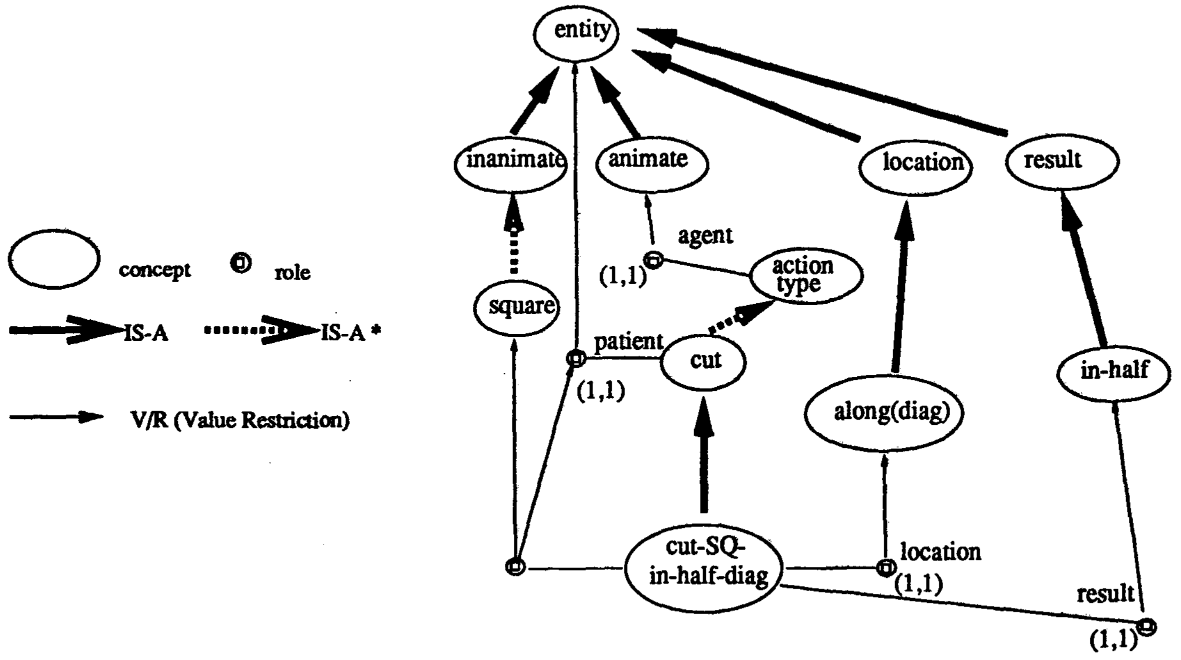

Figure 4: A portion of the action hierarchy

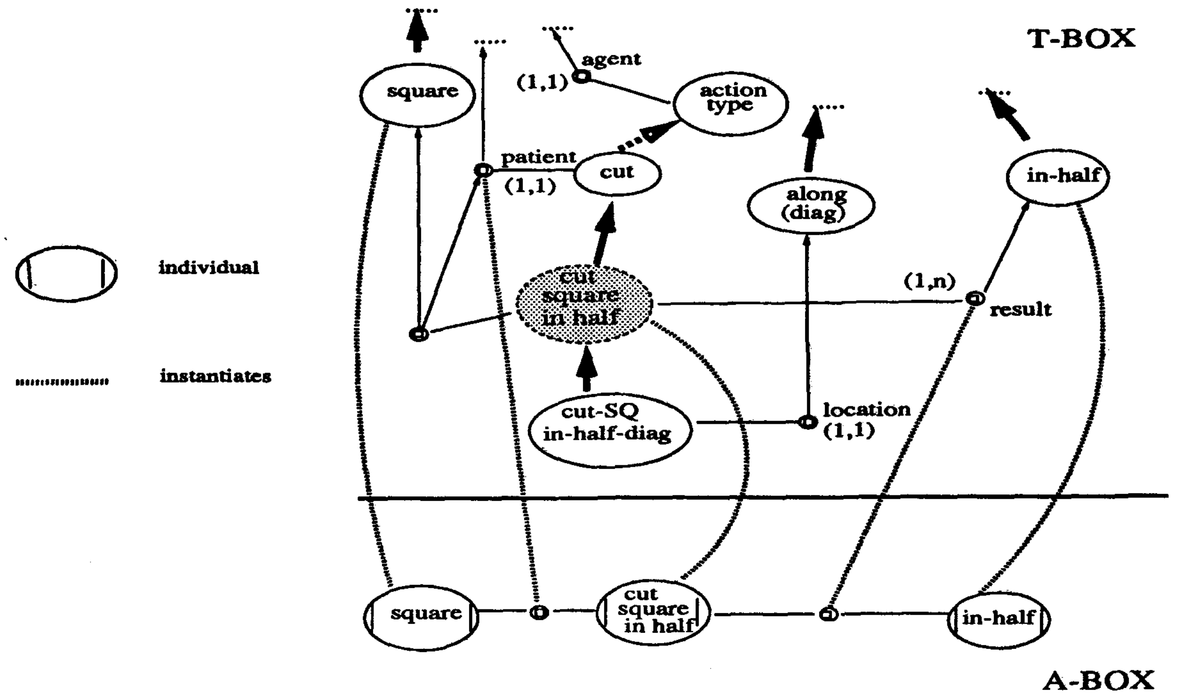

Figure 5: Dealing with less specific action descriptions

CONCLUSIONS

I have shown that the analysis of purpose clauses lends support to the proposal of using generation and enablement to model actions, and that the interpretation of purpose clauses originates specific inferences: I have illustrated two of them, that can be seen as examples of *accommodation* processes (Lewis, 1979), and that show how the hearer's inference processes are directed by the *goal(s)* s/he is adopting.

Future work includes fully developing the action representation formalism, and the algorithm, especially the part regarding computing assumptions.

ACKNOWLEDGEMENTS

For financial support I acknowledge DARPA grant no. N0014-90-J-1863 and ARO grant no. DAALO3-89-C0031PR1. Thanks to Bonnie Webber for support, insights and countless discussions, and to all the members of the AnimNL group, in particular to Mike White. Finally, thanks to the Dipartimento di Informatica - Universita' di Torino - Italy for making their computing environment available to me, and in particular thanks to Felice Cardone, Luca Console, Leonardo Lesmo, and Vincenzo Lombardo, who helped me through a last minute computer crash.

References

(Allen, 1984) James Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

(Alterman *et al.*, 1991) Richard Alterman, Roland Zito-Wolf, and Tamitha Carpenter. *Interaction, Comprehension, and Instruction Usage.* Technical Report CS-91-161, Dept. of Computer Science, Center for Complex Systems, Brandeis University, 1991.

(Badler *et al.*, 1990) Norman Badler, Bonnie Webber, Jeff Esakov, and Jugal Kalita. Animation from instructions. In Badler, Barsky, and Zeltzer, editors, *Making them Move*, MIT Press, 1990.

(Balkanski, 1990) Cecile Balkanski. *Modelling act-type relations in collaborative activity.* Technical Report TR-23-90, Center for Research in Computing Technology, Harvard University, 1990.

(Balkanski, 1991) Cecile Balkanski. Logical form of complex sentences in task-oriented dialogues. In *Proceedings of the 29th Annual Meeting of the ACL, Student Session*, 1991.

(Brachman *et al.*, 1983) R. Brachman, R.Fikes, and H. Levesque. *KRYPTON: A Functional Approach to Knowledge Representation.* Technical Report FLAIR 16, Fairchild Laboratories for Artificial Intelligence, Palo Alto, California, 1983.

(Chapman, 1991) David Chapman. *Vision, Instruction and Action.* Cambridge: MIT Press, 1991.

(Cohen and Levesque, 1990) Philip Cohen and Hector Levesque. Rational Interaction as the Basis for Communication. In J. Morgan, P. Cohen, and M. Pollack, editors, *Intentions in Communication*, MIT Press, 1990.

(Di Eugenio, 1992) Barbara Di Eugenio. *Goals and Actions in Natural Language Instructions.* Technical Report MS-CIS-92-07, University of Pennsylvania, 1992.

(Di Eugenio and White, 1992) Barbara Di Eugenio and Michael White. On the Interpretation of Natural Language Instructions. 1992. COLING 92.

(Goldman, 1970) Alvin Goldman. *A Theory of Human Action.* Princeton University Press, 1970.

(Grosz and Sidner, 1990) Barbara Grosz and Candace Sidner. Plans for Discourse. In J. Morgan, P. Cohen, and M. Pollack, editors, *Intentions in Communication*, MIT Press, 1990.

(Hegarty, 1990) Michael Hegarty. Secondary Predication and Null Operators in English. 1990. Manuscript.

(Jackendoff, 1990) Ray Jackendoff. *Semantic Structures. Current Studies in Linguistics Series*, The MIT Press, 1990.

(Jones, 1985) Charles Jones. Agent, patient, and control into purpose clauses. In *Chicago Linguistic Society, 21*, 1985.

(Lewis, 1979) David Lewis. Scorekeeping in a language game. *Journal of Philosophical Language*, 8:339–359, 1979.

(Peltason *et al.*, 1989) C. Peltason, A. Schmiedel, C. Kindermann, and J. Quantz. *The BACK System Revisited.* Technical Report KIT 75, Technische Universitaet Berlin, 1989.

(Pollack, 1986) Martha Pollack. *Inferring domain plans in question-answering.* PhD thesis, University of Pennsylvania, 1986.

(Vere and Bickmore, 1990) Steven Vere and Timothy Bickmore. A basic agent. *Computational Intelligence*, 6:41–60, 1990.

(Webber and Di Eugenio, 1990) Bonnie Webber and Barbara Di Eugenio. Free Adjuncts in Natural Language Instructions. In *Proceedings Thirteenth International Conference on Computational Linguistics, COLING 90*, pages 395–400, 1990.

(Webber *et al.*, 1991) Bonnie Webber, Norman Badler, Barbara Di Eugenio, Libby Levison, and Michael White. Instructing Animated Agents. In *Proc. US-Japan Workshop on Integrated Systems in Multi-Media Environments. Las Cruces, NM*, 1991.

(Winograd, 1972) Terry Winograd. *Understanding Natural Language.* Academic Press, 1972.