On the Interpretation of Natural Language Instructions

Barbara Di Eugenio Michael White

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA, USA
{dieugeni, mwhite}@linc.cis.upenn.edu

Abstract

In this paper, we discuss the approach we take to the interpretation of instructions. Instructions describe actions related to each other and to other goals the agent may have; our claim is that the agent must actively compute the actions that s/he has to perform, not simply "extract" their descriptions from the input.

We will start by discussing some inferences that are necessary to understand instructions, and we will draw some conclusions about action representation formalisms and inference processes. We will discuss our approach, which includes an action representation formalism based on Conceptual Structures [Jac90], and the construction of the structure of the agent's intentions. We will conclude with an example that shows why such representations help us in analyzing instructions.

1 Making sense of instructions

Consider the following three instructions:

(1a) *Go into the other room to get the urn of coffee.*

(1b) *Before you pick it up, be sure to unplug it.*

(1c) *When you bring it back here, carry it carefully with both hands.*

Let's consider (1a). To understand this instruction, an agent must find the connection between the two actions α—*go into the other room*, and β—*get the urn of coffee*. The infinitival *to* alerts the agent to the fact that α contributes to achieving β. General knowledge about physically getting objects requires that the agent move to the place where the object is located; therefore, the agent will infer that the (most direct) connection between these actions has *go into the other room* fulfilling this requirement. However, this is not enough. An assumption needs to be made for such connection to go through, namely, that *the urn is in the other room*.

This example shows that to *make sense* of instructions, an agent must engage in the active computation of the action(s) to be executed, and cannot simply "extract" all such information from the input. This differentiates our work from others', as we will discuss shortly.

Another important point that arises from (1a) is that the relation *contributes* holding between α, described in the matrix clause, and β, described in the purpose clause[1], can be specified either as *generation* or *enablement*, as a study of naturally occurring *purpose clauses* [Di 92a] shows.

Generation was introduced by [Gol70]. Informally, if action α generates action β, we can say that β is executed by executing α. An example is *Turning on the light* by *flipping the switch*.

Enablement. Following [Pol86] and [Bal90], action α *enables* action β if and only if an occurrence of α brings about conditions necessary for the subsequent performance of β. In *Unscrew the protective plate to expose the box*, "unscrew the protective plate" enables "taking the plate off" which generates "exposing the box".

In [Pol86], it is shown that these two relations are necessary to model action descriptions conveyed by Natural Language. We would like to add one further observation: such relations allow us to draw conclusions about action execution too. This is quite useful since we *do* have to execute (ie., animate) the input instructions, as our work is taking place in the context of the *Animation from Natural Language (AnimNL)* project at the University of Pennsylvania [WBD*91].

As far as generation is concerned, while two actions are described, only α, the generator, needs to be performed; instead, if α enables β, after executing α, β still needs to be executed. In fact, if α enables β, α has to begin, but not necessarily end, before β.

[1] We are using the term *purpose clauses* to informally designate subordinate clauses — such as those introduced by *to* — that express the agent's purpose in executing the action described in the matrix clause. The usage of the term *purpose clause* in the syntactic literature is somewhat different—see [Jon85].

In both cases, the goal β also *constrains* the interpretation and / or execution of α. An example of this as regards generation is

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

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

We turn next to the second instruction (1b). Observe that the agent understands *pick up* to be part of the sequence that achieves *get the urn of coffee*. This is not warranted by the preposition *before:* if (1b) were *Before you ruin it, be sure it's unplugged*, the agent clearly shouldn't infer that *ruin it* is part of *getting the urn*! This shows that in *before* α, β, the action α is not necessarily part of achieving a certain goal, even if β is.

As far as (1c) goes, the agent has to understand that *bring it back here* is part of achieving *getting the urn*; that *carry it carefully with both hands* generates *bring it back here*, provided that *carry it carefully with both hands* is augmented with the destination *back here*. Notice that the action description *carry it carefully with both hands* is fairly complex, sporting two modifiers in addition to the traditional arguments of agent and patient.

2 Problems and Proposed Solutions

The following conclusions can be drawn from the observations in the previous section:

1. NL action descriptions are fairly complex, including modifiers of many different types—see also [WD90]. An action representation formalism must be able to deal with complex descriptions, such as *carry it carefully with both hands*; with descriptions at different levels of abstraction, such as *go* and *walk to*, or such as *cut the square in half* and *cut the square in half along the diagonal* in (2).

2. NL instructions include a wide variety of constructions, such as *purpose clauses* and *temporal clauses*. Instruction interpretation systems must be able to deal with complex imperatives and with the relations between actions that they express.

3. An instruction interpretation system cannot assume that the descriptions of the actions to be performed are equivalent to the logical forms computed by the parser: such logical forms have to be constrained in various ways, e.g. by computing assumptions, as in (1a), or more specific action descriptions, as in (2)[2]. Notice that these constraints derive from the interaction between the actions to be executed and the goals the agent adopts. It is essential that this interaction is taken into account by such systems.

Work done in the past on understanding instructions has generally concentrated on simple positive commands, and has failed to address some of the desiderata listed above: [VB90] limits the interaction between new and preexisting goals to inserting the new goals in the list of goals if their execution does not violate preexisting constraints, otherwise they are rejected. [Cha91] proposes a model of instruction interpretation which seems useful at the level of the basic skills an agent is endowed with, but in which there is no internal structure to actions, and no distinction between the agent's actions and goals. [AZC91] instead does assume a rich relation between instructions and preexisting goal(s). However, instructions are not continually integrated into the plan the agent is developing; instead they are used as a resource when the stored knowledge about plans cannot be adapted to the situation at hand.

Turning now to our proposal, our approach to these problems includes

1. An **action representation formalism** based on Jackendoff's Conceptual Structures [Jac90].

2. An **action KB** that contains simple plans that represent common sense knowledge about actions.

3. A **plan graph** that represents the structure of the agent's intentions.

3 Action representation

We have chosen to use Jackendoff's Conceptual Structures [Jac90] for two reasons. First, as our point of departure is NL, there are the obvious benefits of using a linguistically motivated representational theory, e.g. easing the burden upon the parser to produce such representations [Whi92]. Second, there is significant mileage to be gained from using a decompositional theory of meaning, insofar as the primitives effectively capture important generalizations. In this section we introduce the notation and some minor modifications to the theory as presented in [Jac90]. We use *Go into the other room* as a representative example.

In Jackendoff's theory, an entity may be of ontological type Thing, Place, Path, Event, State, Manner or Property. The conceptual structure for a room is shown in (3a) below:

(3a) [Thing ROOM]

(3b) [Thing KITCHEN]

Square brackets indicate an entity of type Thing meeting the enclosed featural description. Small caps indicate atoms in conceptual structure, which serve as links to other systems of representation; for example, the conceptual structure for a kitchen (3b) differs from that of a

[2] In this paper we will only discuss the former type of constraint computation; the latter is discussed in [Di 92b].

to a body that generates a header. The *annotations* on the body specify the relations between the subactions; such relations include partial temporal ordering, enablement, and possibly others.

From the planning tradition, we retain the notions of *qualifiers* and *effects*. Qualifiers are conditions that make an action relevant: for example, *unplug* x is relevant only if x is plugged.

Notice the importance of using a representation such as Jackendoff's: it helps us capture the common characteristics of different actions, e.g. *get* and *carry*. The semantic representation for *carry* would also match the generic *move*-action template, and would add to it a qualification such as

(10) $[_{\text{Manner}}$ WITH($[_{\text{Thing}}$ HANDS])]

Having such a representation is also useful for computing qualifiers and effects in a systematic way: they can be precompiled from the representation itself. For example, for every action including a component δ such as

$$\left[\text{GO}_{\text{Sp}}\left(j, \left[\begin{array}{l}\text{FROM}([\text{AT}(j)]) \\ \text{TO}(l)\end{array}\right]\right)\right]_{\delta}$$

we know that after δ, j must be at l, therefore we can include this in the effects of the action. Given the further restriction that j cannot be in two places at once, we may infer that j cannot be at l now, and thus precompute the qualifier[8].

4 The plan graph

The *plan graph* represents the structure of the intentions that the agent adopts as a response to the instructions. It keeps track of the goals the agent is pursuing, of the hierarchical relations between the goals and the actions whose execution achieves such goals, and of various relations between the actions. It also helps interpret the instructions that follow. In (1), establishing the initial goal *get the urn of coffee* provides the context in which the two following instructions have to be interpreted—a similar strategy is adopted for example by [Kau90]. In Fig. 2, we show the complete structure built after interpreting (1).

A node in a plan graph contains the Conceptual Structure representation of an action, augmented with the consequent state achieved after the execution of that action[9]. The arcs represent relations between actions; among them, those relevant to our example are: *temporal*, such as *precedes* in Fig. 2; *enablement*; *generation*, and its generalization *substep*, used when α belongs to a sequence of more than one action that generates β.

[8] Jackendoff suggests something analogous with his inference rules, which have yet to be formalized.

[9] In Fig. 2 the labels on the nodes are only mnemonics, and do not represent their real contents.

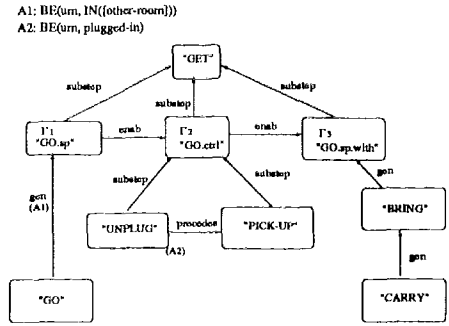

Figure 2: The plan graph.

There may also be assumptions associated with a plan graph. If an assumption is derived from the qualifiers associated with an action, it is associated with the node describing that action—A2 in Fig. 2; if it is derived while inferring a relation between two actions, it is associated with the corresponding arc—A1.

The plan graph is built by an interpretation algorithm that takes as its input the logical form constructed by the parser. The algorithm works by keeping track of the *active nodes*, which include the goal currently in focus, and the nodes just added to the tree. The topmost level of the algorithm invokes different procedures, according to the particular syntactic construction at hand – e.g. the construction *Do* α *to do* β will trigger the hypothesis that α either generates or enables β [Di 92b]. These procedures retrieve the plan(s) associated with the goal currently in focus, and then expand such plans in a hierarchical fashion.

These procedures embody various inference processes, that can be characterized either as *planning*—e.g. plan expansion, subgoaling— or as *plan inference*—e.g. inferring assumptions, inferring the more abstract goal some actions are supposed to achieve. Space doesn't allow us to go into further details about the algorithm or the inference processes; rather, in the next section we will give an example of how assumptions are computed.

5 Making an Assumption

We will now show how the assumption that the urn is to be found in the other room is made while processing (1a), *Go into the other room to get the urn of coffee*.

The process begins with the following representation constructed by the parser, where the FOR-function (derived from the *to*-phrase) encodes the *contributes* relation holding between the *go*-action α and the *get*-action β:

room only in its choice of constant, leaving the determination of their similarities and differences to a system of representation better suited to the task[3].

To distinguish instances of a type, we follow [ZV91] in requiring every conceptual structure to have an index:

(4) $[_{\text{Thing ROOM}}]_1$

Conceptual structures may also contain complex features generated by conceptual functions over other conceptual structures. For example, the conceptual function IN: Thing $\rightarrow$ Place may be used to represent the location *in the room* as shown in (5a) below. Likewise, the function TO: Place $\rightarrow$ Path describes a path that ends in the specified place, as shown in (5b) — (5c) is an equivalent representation of (5b), where the index l stands for the entire constituent[4]:

(5a) $[_{\text{Place}}\ \text{IN}([_{\text{Thing ROOM}}]_k)]_l$

(5b) $[_{\text{Path}}\ \text{TO}([_{\text{Place}}\ \text{IN}([_{\text{Thing ROOM}}]_k)]_l)]_m$

(5c) $[_{\text{Path}}\ \text{TO}(l)]_m$

To complete our clause[5], it remains only to add the conceptual function GO: Thing $\times$ Path $\rightarrow$ Event:

(6) $[_{\text{Event}}\ \text{GO}([_{\text{Thing}}\ ]_i, m)]$
$[_{\text{Path}}\ \text{TO}([\text{IN}([_{\text{OTHER-ROOM}}])])]_m$

As there is no subject in our clause, the constituent i (pragmatically, the AGENT) in (6) is left unspecified.

To distinguish *Walk into the other room* from (6), we include an indication of manner[6]:

(7) $\begin{bmatrix} \text{GO}(i, m) \\ [_{\text{Manner}}\ \text{WALKING}] \end{bmatrix}$

Finally, semantic fields, such as Spatial and Possessional, are intended to capture the similarities between sentences like *Jack went into the other room* and *The gift went to Bill*, as shown in (8) below:

(8a) $[\text{GO}_{\text{Sp}}([\text{JACK}], [\text{TO}([\text{IN}([_{\text{OTHER-ROOM}}])])])]$

(8b) $[\text{GO}_{\text{Poss}}([\text{GIFT}], [\text{TO}([\text{AT}([\text{BILL}])])])]$

The idea is that verbs like *go* leave the semantic field underspecified, whereas verbs like *donate* specify a particular field. In addition to these semantic fields, we propose to add a new one called Control. It is intended to represent the functional notion of *having control over* some object. For example, in sports, the meanings of *having the ball*, *keeping the ball*, and *getting the ball* embody this notion, and are clearly quite distinct from their Spatial and Possessional counterparts; (9) represents *Jack got the ball*:

(9) $[\text{GO}_{\text{Ctrl}}([\text{BALL}], [\text{TO}([\text{AT}([\text{JACK}])])])]$

[3]In our case, the action representation formalism is grounded in the animation system serving as the back-end to the *AnimNL* project.

[4]We will often adopt the representation in (5c), and leave out indices and ontological types, in order to lessen the typographical burden of representing large conceptual structures.

[5]Ignoring, of course, the meaning of *other* for now.

[6]Though this is clearly intended, Jackendoff never explicitly represents such a distinction.

Header
$[\text{CAUSE}([_{\text{AGENT}}]_i, [\text{GO}_{\text{Sp}}(j, k)])]$ $\left[\begin{array}{l} \text{FROM}([\text{AT}(j)]) \\ \text{TO}(l) \end{array}\right]_k$
Body
- $[\text{GO}_{\text{Sp}}([i, [\text{TO}([\text{AT}(j)])])]_{\gamma_1}$ - $[\text{CAUSE}(i, [\text{GO}_{\text{Ctrl}}(j, [\text{TO}([\text{AT}(i)])])])]_{\gamma_2}$ - $\left[\begin{array}{l} \text{GO}_{\text{Sp}}(i, k) \\ [\text{WITH}(j)] \end{array}\right]_{\gamma_3}$ - Annotations -   - γ_1 *enables* γ_2 *enables* γ_3
Qualifiers
- $[\text{NOT BE}_{\text{Sp}}(j, l)]$
Effects
- $[\text{BE}_{\text{Sp}}(j, l)]$

Figure 1: A *Move Something Somewhere* Action.

3.1 The action KB

The action KB contains simple plans that represent common sense knowledge about actions, and whose components are expressed in terms of Jackendoff's semantic primitives. To discuss the characteristics of these plans, we will refer to the *move*-action KB entry shown in Fig. 1, which might be described as follows: go to where j is, get control over it, then take it to l[7].

Actions have a *header* and a *body*. This terminology is reminiscent of planning operators; however we express the relations between these components in terms of *enablement* and *generation*—e.g. the body *generates* its header.

The representation does not employ preconditions, because it is very difficult to draw the line between what is a precondition and what is part of the body of an action. One could say that *having control over* the object to be moved is a precondition for a *move*-action. However, if the object is heavy, the agent will start exerting force to lift it, and then carry it to the other location. It is not obvious whether the lifting action is still part of achieving the precondition, or already part of the body. Therefore, we don't have preconditions, but only actions which are substeps in executing another action, that is, they may belong

[7]This do-it-yourself method is but one way to move something from where it is to somewhere else. Other methods would be listed separately in the action KB.

$$\left[\begin{array}{l} \mathrm{GO_{Sp}}([\textsc{agent}]_i, [\mathrm{TO}([\mathrm{IN}([\textsc{other-room}])])]) \\ \mathrm{FOR}(\beta) \end{array}\right]_\alpha$$

$$[\mathrm{CAUSE}(i, [\mathrm{GO_{Sp}}([\textsc{urn-of-coffee}]_j, k)])]_\beta$$

$$\left[\begin{array}{l} \mathrm{FROM}([\mathrm{AT}(j)]) \\ \mathrm{TO}(l) \end{array}\right]_k$$

Given the presence of the *to* phrase, we know that α may be part of a sequence of actions that generate β. To pursue this hypothesis, we begin by looking up β in the action KB. β matches the general *move*-action shown in Fig. 1 if the object to be moved j is bound to *the urn of coffee*:

$$j \quad = \quad [\textsc{urn-of-coffee}]$$

Next we try to match α with some subaction γ of β. α matches the first action γ_1 in β if we take $[\mathrm{AT}(j)]$ and $[\mathrm{IN}([\textsc{other-room}])]$ to be the same place. This is tantamount to making the following assumption:

(11) $[\mathrm{BE_{Sp}}(j, [\mathrm{IN}([\textsc{other-room}])])]$

Once the instruction is understood in this way, the two actions may be incorporated into the plan graph as shown in Fig. 2.

One should mention that assumption (11) could of course be wrong, say if there were a note in the next room saying *ha ha, it's not really in this room but the next*.

Notice that even if there is already an urn of coffee in the current room, the instruction *Go into the other room to get the urn of coffee* is still understood to refer to an urn in the other room. This contrasts sharply with *Go into the other room to* **wash out** *the urn of coffee*, where the most likely urn is the currently visible one. In the current framework, this difference would be captured in the following way. Unlike in the case of the *get*-action, the *go*-action matches the following subaction of *wash-out*:

$$[\mathrm{GO_{Sp}}([i, [\mathrm{TO}([\mathrm{AT}(\textsc{washing-materials})])])]_\gamma$$

Therefore, assumption (11) will not be derived, permitting the possibility of the urn being in the current room.

6 Summary and Future Research

We have presented an approach to action representation and instruction interpretation which we feel is more flexible than previously proposed formalisms: it allows us to use terms at different levels of specificity, and to perform the complex inferences that NL instructions require.

Future research includes exploring how to integrate a hierarchical organization of entities, actions and plans with the action KB.

The system is being implemented in Quintus Prolog, with substantial progress having been made in particular on the parser [Whi92], and on the action KB.

Acknowledgements

This research was supported by the following grants: DARPA no. N00014-90-J-1863, ARO no. DAAL 03-89-C-0031, NSF no. IRI 90-16592, and Ben Franklin no. 91S.3078C-1. We would like to thank all the members of the AnimNL group, and in particular Bonnie Webber, Libby Levison and Chris Geib, for very stimulating and helpful discussions.

References

[AZC91] Richard Alterman, Roland Zito-Wolf, and Tamitha Carpenter. *Interaction, Comprehension, and Instruction Usage*. Technical Report CS-91-161, Brandeis University, 1991.

[Bal90] Cecile Balkanski. *Modelling act-type relations in collaborative activity*. Technical Report TR-23-90, Center for Research in Computing Technology, Harvard University, 1990.

[Cha91] David Chapman. *Vision, Instruction and Action*. Cambridge: MIT Press, 1991.

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

[Di 92b] Barbara Di Eugenio. Understanding Natural Language Instructions: the Case of Purpose Clauses. In *Proceedings ACL 92*, 1992.

[Gol70] Alvin Goldman. *A Theory of Human Action*. Princeton University Press, 1970.

[Jac90] Ray Jackendoff. *Semantic Structures. Current Studies in Linguistics Series*, The MIT Press, 1990.

[Jon85] Charles Jones. Agent, patient, and control into purpose clauses. In *Chicago Linguistic Society, 21*, 1985.

[Kau90] Henry Kautz. A circumscriptive theory of plan recognition. In J. Morgan, P. Cohen, and M. Pollack, editors, *Intentions in Communication*, MIT Press, 1990.

[Pol86] Martha Pollack. *Inferring domain plans in question-answering*. PhD thesis, University of Pennsylvania, 1986.

[VB90] Steven Vere and Timothy Bickmore. A basic agent. *Computational Intelligence*, 6:41–60, 1990.

[WBD*91] 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.

[WD90] Bonnie Webber and Barbara Di Eugenio. Free Adjuncts in Natural Language Instructions. In *Proceedings COLING 90*, pages 395–400, 1990.

[Whi92] Michael White. Conceptual Structures and CCG: Linking Theory and Incorporated Argument Adjuncts. 1992. COLING '92.

[ZV91] Joost Zwarts and Henk Verkuyl. An Algebra of Conceptual Structure; an investigation into Jackendoff's Conceptual Semantics. 1992. To appear in *Linguistics and Philosophy*.