Pragmatic overloading in Natural Language instructions

Barbara Di Eugenio
Computational Linguistics
Department of Philosophy
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
dieugeni@andrew.cmu.edu

Bonnie Lynn Webber
Computer and Information Science
Moore School
University of Pennsylvania
200 S. 33rd Street
Philadelphia, PA 19104
bonnie@central.cis.upenn.edu

Abstract

It has long been noted that Natural Language utterances can communicate more than their conventional meaning (Grice, 1975). It has also been noted that behaving appropriately in response to instructions given in Natural Language requires understanding more than their conventional meaning (Suppes and Crangle, 1988; Webber and Di Eugenio, 1990; Webber et al., 1992). This paper addresses one mechanism by which speakers convey, and hearers derive, such additional aspects of meaning – a mechanism we call *pragmatic overloading*. In pragmatic overloading, a clause interpreted as conveying directly or indirectly the *goal* β of an action α which is described by some other clause, forms the basis of constrained inference that leads to additional information about the action α. The paper demonstrates *pragmatic overloading* through a variety of clausal adjuncts. We then discuss a framework that supports many of the inferences that pragmatic overloading gives rise to. This framework integrates a lexical semantics representation á la Jackendoff (1990) with a knowledge representation system, CLASSIC (Brachman et al., 1991), based on description logic. We give examples of its use, before concluding with a discussion of future work.

1 Introduction

It has long been noted that Natural Language utterances can communicate more than their conventional meaning (Grice, 1975). It has also been noted that behaving appropriately in response to Natural Language instructions requires that more than their conventional meaning be understood (Suppes and Crangle, 1988; Webber and Di Eugenio, 1990; Webber et al., 1992). For example, consider what a hearer[1] must know to carry out each of the following instructions:

(1a) *Carry the beakers carefully.*

(1b) *Go to the mirror and fix your hair.*

In Ex. (1a), a hearer must derive from "carefully", constraints on his behavior that will keep the beakers from breaking and their contents from spilling. In Ex. (1b), the hearer must derive from "to the mirror" and "fix your hair", a location in front of the mirror that will enable him to see his hair clearly.

It is not that equivalent inferences can not be drawn from corresponding declarative sentences:

(2a) *Mary carried the beakers carefully.*

(2b) *Fred went to the mirror and fixed his hair.*

And it is not that other inferences cannot be drawn from Ex. (2a) and (2b) either. It is that in the case of instructions to behave appropriately, particular behavior-related inferences **must** be drawn if the information is not otherwise provided.[2] It is here that the core linguistic-pragmatic notion of relevance (Grice, 1975; Wilson and Sperber, 1986) comes into play: it is because they are needed to behave appropriately that these inferences are relevant.

In this paper we discuss a particular constrained mechanism by which speakers convey, and hearers derive, additional aspects of meaning. We call this mechanism *pragmatic overloading*. In pragmatic overloading, a clause interpreted as conveying directly or indirectly the *goal* β of an action α described by another clause forms the basis of constrained inference that leads to additional information about that action. In Sec. 3, we demonstrate *pragmatic overloading* through a variety of clausal adjuncts that perform double duty: explicitly, they convey one specific relation $\mathcal{R}$ between the action described in the main clause α and the action β or state κ described in the adjunct; implicitly, they refine the interpretation of α, producing a more specific action description α', or they refine the interpretation of $\mathcal{R}$.

The reader should note that we are not describing a fully implemented system. A prototype system, *AnimNL* for *Animation and Natural Language* (Webber et al., 1992; Webber et al., 1995), does exist (Sec. 2 and Sec. 4), and embodies our attempt to use instructions to guide the task-related behavior of animated human figures. The algorithm described in Sec. 4 is implemented within this system, but it is able to handle only some of the input constructions discussed in Sec. 3 that give rise to pragmatic overloading. However, we claim that the representation and reasoning we propose in Sec. 4 provide solid theoretical and practical foundations for further development of the system.

The paper is organized as follows. In Sec. 2 we briefly describe *AnimNL* and *Jack*®, the animation system on which *AnimNL* is based. Sec. 3 describes *pragmatic overloading*, differentiating it from other pragmatic inferences, and provides the empirical evidence for it. Sec. 4 then discusses a Knowledge Representation framework that integrates a lexical semantics representation (Jackendoff, 1990) with a system based on description logic (Brachman et al., 1991). We demonstrate that those two components provide an elegant way of supporting many of the inferences that pragmatic overloading gives rise to. Finally, we conclude with a note on how the work described here is now proceeding.

A comment on notation: given that we will discuss adjuncts, we will reserve the symbol α to the action described in the main clause, and the symbols β and κ to the object described in the adjunct, β for an action, κ for a state. $\mathcal{R}$ is the relation holding between α and β or κ.

2 Animation from NL instructions

The work described here has been done in connection with an ongoing project at the University of Pennsylvania called *AnimNL* for *Animation and Natural Language*, aimed at enabling users to guide the task-related behavior of animated human figures through NL instructions (Webber et al., 1992; Badler, Phillips, and Webber, 1993; Badler et al., 1993; Webber et al., 1995; Webber, 1995). *AnimNL* builds upon *Jack*®, an animation system developed by the University of Pennsylvania's *Center for Human Modeling and Simulation.* In *Jack*, animation follows from model-based simulation of virtual agents acting in an environment.

The agents of primary interest are *Jack*'s biomechanically reasonable and anthropometrically scaled human models — see Fig. 1, which shows Jack at a soda fountain. In task-related behavior, the movements of these

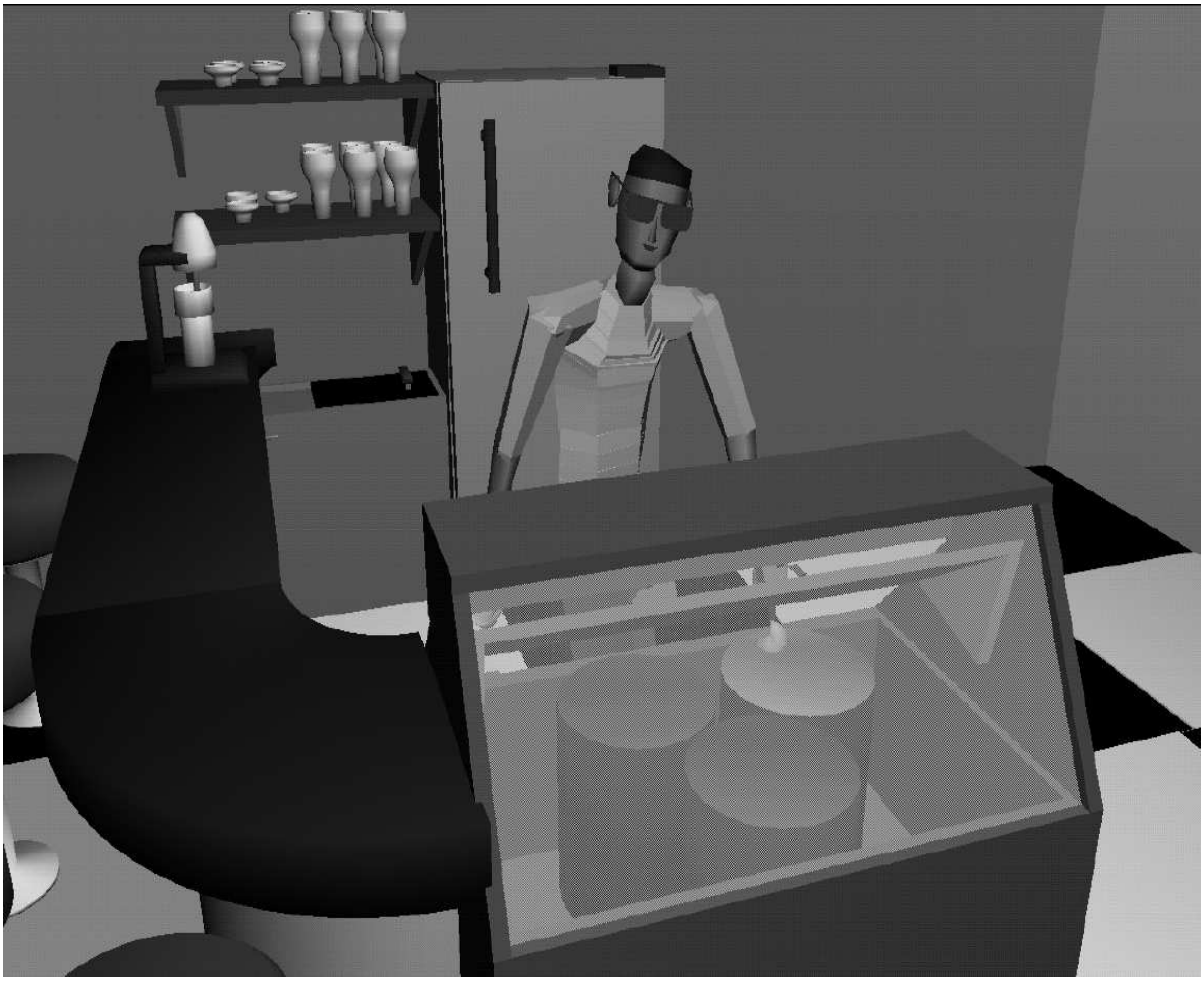

Figure 1: *Jack* at a soda fountain

animated human figures result from the interaction of at least four different factors:

- a growing repertoire of built-in behaviors such as walking, obstacle avoidance, stepping, turning, grasping, strength-guided lifting, etc. (Badler, Phillips, and Webber, 1993), that remove responsibility from either Natural Language understanding or high-level planning to control all behavior from "above". Each of these behaviors is environmentally reactive in the sense that incremental computations during simulation are able to adjust an agent's performance to the situation *without further involvement of the higher level processes* (Becket and Badler, 1993; Badler et al., 1995) unless an exceptional failure condition is signaled;
- intentions and expectations an agent adopts in response to understanding its given instructions, or *apropos* its current stage in a task (Webber et al., 1995);
- knowledge given to an agent of how to act in different environments in order to try to satisfy the intentions it adopts (Geib, 1995);

- limitations established on agent "perception": visual "perception" is limited to objects within a bounded space that are not otherwise obstructed. Currently, research is being carried out on a model of "perception" that distinguishes between what is available to an agent through foveal vision versus peripheral vision (Chopra, 1994), which will support more realistic focus and shifts of attention during task performance and other environmental changes.

Jack is described in more detail in (Badler, Phillips, and Webber, 1993; Becket and Badler, 1993). We will come back to *AnimNL* in Sec. 4.2.

3 Pragmatic Overloading

The term *overloading* has been used before in other contexts. In programming languages, an operator is said to be *overloaded* if it can be used for more than one purpose. For example, in many programming languages, "+" is overloaded, indicating both integer and real addition. In the context of AI planning, Pollack (1991) has used the term *overloading* to refer to cases where a single intention to act is used to wholly or partially satisfy several of an agent's goals simultaneously.

Our use of *overloading* is closer to Pollack's. It differs in that we are concerned with a communicative domain, that involves two agents, a speaker and a hearer. The speaker, knowing that some necessary information regarding *how* to act can be derived from knowing the *purpose* of acting, and believing that the hearer will recognize that the information is both necessary and missing, relies on the hearer deriving the missing information from her (the speaker's) expression of purpose. Thus we say that her expression of purpose is *pragmatically overloaded* – both conveying purpose and leading the hearer to infer additional constraints on his behavior.

As an example of *pragmatic overloading*, consider the instruction:

(3) *Hold the cup under the spigot to fill it with coffee.*

The infinitival adjunct here is an instance of a *Purpose Clause*. It both describes a goal β = *fill the cup with coffee*, and conveys a relation $\mathcal{R}$ between β and the action described in the main clause, α = *hold the cup under the spigot*. For the moment, following (Grosz and Sidner, 1990), we will refer to $\mathcal{R}$ as *contribute*. Notice that α does not specify how the cup should be held under the spigot – horizontally or vertically, and if vertically, with the concavity pointing upwards or downwards. But also notice that β can lead the hearer to appropriate constraints on that choice: to fill a cup with coffee, the cup must be held vertically, with its concavity pointing upwards. This constraint does not simply derive from default knowledge, but crucially depends on the *purpose* α is performed for. This is clear from Ex. (4), where the given purpose does not constrain cup orientation in the same way:

(4) *Hold the cup under the faucet to wash it.*

We claim that β can lead to additional constraints on α as a joint consequence of the fact that *contribute* holds between α and β, and the fact that the instructor intends the agent to recognize that such relation holds. The *Purpose Clause* construction may then perform two functions: explicitly, it conveys the *contribute* relation $\mathcal{R}$; implicitly, it may constrain the interpretation of α or $\mathcal{R}$. We therefore say it is *pragmatically overloaded*.

Clauses other than *Purpose Clauses* can demonstrate pragmatic overloading as well. In instructions, an *until* clause specifies the condition κ an agent should monitor for, as a signal to terminate the process denoted by the main clause α. For example, consider the *move* action in Ex. (5):

(5) *Have your helper move the tape sideways until the 4-foot mark on the tape coincides with the 5-foot mark on the ruler.*

Whether to move towards the left side or towards the right is not stated explicitly. Rather, the *until* clause suggests the direction (i.e., the sideways direction that will bring the marks into alignment), as well as specifying when the *move* action should terminate. Pragmatic overloading thus conveys a more refined version α' of the action description α given in the main clause.

The additional information conveyed by β can also enable the agent to refine the *contribute* relation $\mathcal{R}$ between α and β. The particular refinement we have noticed is the addition of expectations that the agent will assume hold in order for α to contribute to β. Consider:

(6) *Open the box and hand me the yellow block.*

The expectation here concerns the location of the referent of the NP *the yellow block*, that the agent expects to find inside the box. It is a constraint on the relation between α = *open the box* and β = *hand me the yellow block*: it is under the condition that *the yellow block is in the box* that α enables β, i.e. brings about a condition necessary for β to be executable. Again, we can see pragmatic overloading at work: the syntactic construction of *purposive and* (Doran, 1993) — for now, we have set aside the enormous complexity deriving from the ambiguity of *and* — explicitly conveys a purpose relation, while implicitly giving rise to this additional expectation.

For Pollack (1991), overloading intentions is a strategy for focusing means-end analysis. To demonstrate where overloading "pays off", Pollack compares it to a strategy of complete deliberation. To demonstrate the same of *pragmatic overloading*, we can compare it to providing the full descriptions α' or $\mathcal{R}'$. We have already invoked the Gricean notion of *relevance*. In a similar vein, the two strategies, overloading versus full description, can be seen as corresponding respectively to Grice's Maxim of Manner (more specifically, the sub-maxim *Be brief*) and his Maxim of Quantity (roughly, *Provide exactly enough information as is required for the current purposes of the exchange*). Intuitively, *pragmatic overloading* will be employed when its expected benefits (reduced generation time on the part of the instructor and less text for the agent to process) outweigh its expected costs (for the instructor, the cost of deciding what features of α' or $\mathcal{R}'$ can be left out without risk that the agent will compute a wrong α' or $\mathcal{R}'$, and for the agent, the cost of computing α' or $\mathcal{R}'$). In cases such as Ex. (5) above, where the only description of movement direction possible when the instruction is issued is that it will bring the two marks into coincidence, a pragmatic overloading strategy wins over full description.

Pragmatic overloading bears a resemblance to other phenomena discussed in the discourse processing and plan recognition literature. In discourse processing, Moore and Pollack (1993) have shown that multiple relations may simultaneously hold between two discourse elements, some being at the *informational* level of analysis, others being at the *intentional* level. In their framework, pragmatic overloading is something that would occur solely within the informational level: given a particular intentional level analysis of discourse elements, pragmatic overloading accounts for multiple kinds of information relevant to the agent's behavior they convey.

Reasoning similar to pragmatic overloading also occurs in plan recognition, which Kautz (1990) defines as:

> One is given a fragmented, impoverished description of the actions performed by one or more agents, and expected to infer a rich, highly interrelated description. The new description fills out details of the setting, and relates the actions of the agents in the scenario to their goals and future actions.

While the refinements effected by pragmatic overloading (α' or $\mathcal{R}'$) can be considered a *more specific description [that] fills out [some of] the details of the setting*, plan recognition has generally been concerned with inferring, from given observations, additional events that have not been directly observed. As such, this process resembles Lewis' notion of *accommodation* (1979), which he defines as the process by which *the conversational score tends to evolve in such a way as is required in order to make whatever occurs count as correct play.* By conversational score, Lewis means the state of the conversation, given in terms of components such as sets of presuppositions. Concretely, accommodation results in new "objects" being added to the conversational score if this is required in order for the conversation to "make sense".

In linguistics, accommodation has primarily been invoked to account for presuppositions and novel definites such as *the door* in *"I went home. The door was ajar"*. Even if *the door* has never been talked about, a referent for it is readily added to the conversational score. However, Lewis has also discussed a hearer's accommodation of additional actions that enable a conversation to "make sense", much in the way that plan recognition infers additional actions and a plan that "makes sense" of the observed actions. Pragmatic overloading differs from accommodation in refining an entity's description rather than introducing new entities: for example, we would ascribe inferring actions not mentioned in the input and necessary to execute an instruction, e.g. *depress the lever* in (3), to accommodation (or plan recognition), and not to pragmatic overloading. Resolving pragmatic overloading and handling accommodation appeal to different types of constrained inference.

Much still remains to be explored as regards pragmatic overloading. What this paper addresses, as will be explained in detail in the following sections, is inferences that

1. arise while mapping the input surface form to the symbolic knowledge the agent already has about α and β;

2. thus arise at the level of interpretation, before the agent engages in executing the instructions.

While these inferences (like those of plan recognition) rely on the fact that the actions being talked about are ones the agent is familiar with, they do accomplish two things: (1) they enable behavior-related information to be conveyed implicitly, and (2) they allow for significant variations in the way actions are described (see Sec. 4).

The patterns of inference we associate with pragmatic overloading are schematically represented in Fig. 2, where β corresponds to the goal description, α to the action description constrained by β, and $\mathcal{R}$ to the relation holding between α and β. α' and $\mathcal{R}'$ correspond to the more refined descriptions that the hearer infers: examples where α' is inferred are (3) and (5), while $\mathcal{R}'$ is inferred in Ex. (6). Note that Fig. 2 suggests a third possible kind of refinement — where the goal description β is refined to a more specific description β'. Although we have not yet found any naturally occurring examples of such phenomenon, see (Di Eugenio, 1993, pp.87, 147) for some considerations in this respect.

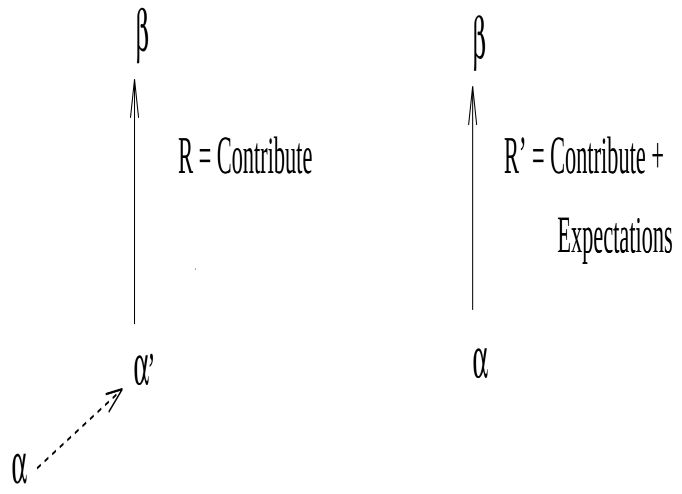

Figure 2: Two types of description refinement

To motivate the form of representation and reasoning presented in Sec. 4, we will show how in *purpose clauses* — *Do α to do β* — and *gerundive free adjuncts* — *Do α, doing β* — further specification of α and/or $\mathcal{R}$ follows directly from recognizing β as the intended reason for α. We also show how in *until* clauses — *Do α until κ* — further specification of α follows indirectly from linking the termination condition κ of α with its goal. The cases are complementary in that, if α in *Do α to do β* specifies a process, then β can be interpreted as constraining its termination condition. On the other hand κ in *Do α until κ* can often be construed as conveying the goal that α should achieve, as well as its termination condition.

The data was collected from four how-to-do books, (Hallowell, 1988b; Hallowell, 1988a; RD, 1991; McGowan and DuBern, 1991). As noted earlier, we will use α to refer to the action description in the main clause and β and κ to the action (β) or state (κ) description in the adjunct. In addition, we will use expressions such as *executing α/β* to mean *executing the action denoted by α/β.*

3.1 Direct expressions of purpose

3.1.1 Purpose Clauses

There are a variety of expressions that convey purpose, among them infinitival constructions introduced by *(in order) to, for nominalizations, for gerunds* (Vander Linden and Martin, 1995), subordinates introduced by *so that*, *such that*, purposive *and*, and *free adjuncts*. Here we will mainly discuss *Purpose Clauses* — PC's for short — infinitival *to* constructions that occur in patterns such as *Do α to do β*, or *To do β, do $\mathcal{A}$*, where $\mathcal{A}$ is a sequence of actions. Our corpus consists of 230 instances of PC's.[3]

A Purpose Clause is used to convey to the agent the goal β to whose achievement the execution of the action denoted by α contributes. As mentioned above with respect to (3), very often the construction is *pragmatically overloaded*, in that the goal β also *constrains* the interpretation of α. Another example in point is:

(7) *Turn screw* **to** *loosen.*

α = *Turn screw* is underspecified in that the direction in which to turn the screw is missing; the goal

β = *to loosen* provides such direction as *counterclockwise*, under the assumption the screw is right threaded.[4] Another example is:

(8) **To** *treat badly corroded brass that is showing signs of verdigris, immerse it in a dip.*

Here α = *immerse it in a dip* is underspecified, as clearly not any dip will achieve β = *treat badly corroded brass*.[5] The type of dip is then determined by β.

As noted in Sec. 1, β can also constrain the *contribute* relation $\mathcal{R}$ by creating expectations about object locations that have to be true for $\mathcal{R}$ to hold between α and β. Purpose clauses behave similar to the purposive *and* illustrated in Ex. (6) in this regard. Consider:

(9a) *[Stop by the library]*$_{\alpha_1}$ *[***to** *get me the Italian dictionary]*$_{\beta_1}$.

(9b) *[Stop by the library]*$_{\alpha_2}$ *[***to** *return the Italian dictionary]*$_{\beta_2}$.

(9c) *[Stop by the library]*$_{\alpha_3}$ *[***to** *buy me some vegetables]*$_{\beta_3}$.

In all three cases, α_i = *Stop by the library* is interpreted as contributing to achieving β_i. However, in (9a), but not in (9b), the basic *contribute* relation is further constrained by a condition inferred from β: the expectation that the referent of *the Italian dictionary* is in the library. These expectations arise when α changes the perceptual space the agent has access to from $\mathcal{S}$ to $\mathcal{S}$', and this change of perceptual space is interpreted as a necessary condition for β to be executed. Namely, if α is done with the purpose of doing β and results in the agent having access to $\mathcal{S}'$; and if β has among its requirements that a participant be at $\mathcal{S}'$ for β to be relevant, then a locational expectation develops as in (9a). These expectations don't just arise from world knowledge, as shown by the fact that in (9b), while the same natural association between libraries and books is evoked, contrary to (9a) no expectation arises. On the other hand, it does arise in (9c), when there isn't a natural association between libraries and vegetables or libraries and buying: that is why the example seems so strange. Notice also that these expectations are necessary to correctly *ground* referential NP's, namely, to resolve them against their correct referent in the world: if for example there is an Italian dictionary in sight at the time the instruction is issued, it won't do if the agent hands it to the speaker, on the grounds he has found a referent in the world satisfying the description of the referential NP *the Italian dictionary*.[6]

To solve this kind of cases, notions deriving from the planning literature, such as *qualifiers*, are necessary — see Sec. 4.

The Contribute relation. So far, we 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*. What we will show is that pragmatic overloading can occur in both cases, as long as the instructor intends the agent to recognize such relation.

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, 1993). Intuitively, if α generates β, executing α in appropriate circumstances is all that is required to achieve β. This is formalized (see e.g. (Pollack, 1986; Balkanski, 1993)) by requiring that

1. α and β be simultaneous, where simultaneity has to be strictly interpreted to exclude cases in which α is part of doing β;

2. when α occurs, a set of conditions $\mathcal{C}$ hold, such that the joint occurrence of α and $\mathcal{C}$ entail the occurrence of β. If $\mathcal{C}$ doesn't hold, α may still occur, but β won't.

The above definition relates *act-types*, rather than action *occurrences*. It is called *conditional generation* by Pollack and by Balkanski, who reserve the term *generation* for the relation holding in the world between two action *occurrences* when the corresponding act-types are related by *conditional generation*. The issues they address require both notions. Our discussion of action *descriptions* only requires one, which we will refer to as *generation* for brevity. What we will always mean when we say that a generation relation holds between α and β is that the act-type specified by the action description α (or a refinement of α) conditionally generates the act-type specified by the action description β.

Ex. (7) illustrates the *generation* relation, in that nothing else needs to be done after *turning the screw (counterclockwise)* to achieve *loosening the screw*. In other cases, 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. Ex. (8) may be interpreted as such: *immerse it in a dip* is part of a sequence of actions that generate *treating badly corroded brass*.

Generation is a pervasive relation between action descriptions in naturally occurring data — out of 230 PC's, about 75% express generation. Most of the other clauses express *enablement* — discussion of the few PC's that express neither generation nor enablement can be found in (Di Eugenio, 1993). Enablement is defined informally as α bringing about conditions $\mathcal{C}$ necessary for β to be executable: this may mean either that $\mathcal{C}$ are part of the *executability* conditions on β itself, or of the conditions under which a third action γ generates β — this definition and its formalization are due to (Balkanski, 1993).[7]

Ex. (3) above is an example of enablement, in which *holding the cup under the spigot* brings about one of the conditions necessary for a third action γ = *depress the lever* to generate *fill the cup with coffee*. Another example is:

(10) *Unscrew the protective plate* **to** *expose the box.*

Unscrew the protective plate brings about one of the executability conditions on *taking the plate off*, which in turn generates *exposing the box*.

Speakers can use pragmatic overloading when conveying either generation or enablement. Examples of generation involving overloading are Exs. (7), in which the further constraint on α is *counterclockwise*, and (8), in which the further constraint on α is *dip for cleaning badly tinted brass*. Examples of enablement involving overloading are Ex. (3), in which the further constraint on α is *concavity pointing upwards*; and Exs. (6), (9a) and (9c), where the further constraint does not refine α, but the relation $\mathcal{R}$ between α and β, and is the expectation concerning the location of one of the surface arguments of β.

As we mentioned earlier, pragmatic overloading does not necessarily occur: in (10), *exposing the box* places neither additional constraints on *unscrewing* nor expectations on the enablement relation. While it is true that new actions that are not explicitly mentioned in the input may be inferred in this and other cases involving enablement — γ = *depress the lever* in (3), and γ = *take the plate off* in (10) — we don't consider such inferences as part of pragmatic overloading, but rather, of the concurrent necessary inferences of plan recognition.

There are also examples of refinement involving the aspectual sense of action descriptions. In

(11) *If the tiles don't come up easily, warm them to soften the adhesive.*

the goal β = *to soften the adhesive* constrains the description (and hence execution) of α = *warm [the tiles]* by augmenting it with a (somewhat fuzzy) termination condition (i.e., when the adhesive becomes soft), which a process such as *warm* does not intrinsically have. Its more specific termination condition – when the adhesive becomes soft enough for the tiles to come up easily – requires additional reasoning that involves the conditional *if the tiles don't come up easily* as well. As we noted earlier, the implementation discussed in Sec. 4 does not cover this type of inference triggered by pragmatic overloading.

3.1.2 Free adjuncts

As mentioned above, there are a variety of expressions that directly convey purpose relations; among those expressions, we want to briefly discuss *free adjuncts*. A *free adjunct* is defined as *a nonfinite predicative phrase with the function of an adverbial subordinate clause* (Stump, 1985). It may be headed by a noun, adjective, prepositional phrase, or verb.[8] Here we focus on free adjuncts headed by progressive gerundives, as they are quite common in instructions – e.g., the clause in boldface in Ex. (12):

(12) *Pour mixture over cheese in casserole,* **spreading evenly**.

In gerundive adjuncts, the relation $\mathcal{R}$ between α (the action described in the main clause) and β (the one described in the adjunct) is more ambiguous than in the case of Purpose Clauses. As we reported in (Webber and Di Eugenio, 1990) with respect to a corpus of 97 free adjuncts, and verified recently on a second corpus of 49 free adjuncts, we found three kinds of relations possible between α and β: simple augmentation, generation, or temporal relations. As with purpose clauses, when the free adjunct conveys generation, pragmatic overloading is possible — for example,

(13) *Cut the square in half creating two triangles.*

The action to be performed is α = *cut 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 β = *create two triangles* restricts the choice to *cutting the square along one of the two diagonals*, which does generate β. Notice that in the case of free adjuncts the directionality of the relation is not restricted by the syntax, as it is in Purpose Clauses, where, when generation holds, it is *always* α that generates β: in free adjuncts, it may be α that generates β or β that generates α.

It appears that adjuncts are not used to express enablement. This may account for why we have not found examples of free adjuncts in which pragmatic overloading leads to additional expectations associated with $\mathcal{R}$: in fact, most expectations we have recognized arise in the context of α's that change the perceptual space of the agent and that are related to β by enablement.

3.2 Until clauses

Utterances of the form *Do* α *until* κ are common in maintenance and repair instructions. There are over 180 occurrences in (RD, 1991; McGowan and DuBern, 1991). Instructions of this form are a challenge to our instruction animation enterprise (Sec. 2) because they require directing animated agents to attend to the right things perceptually, as well as do the right things physically. For Natural Language Processing, such instructions provide additional evidence for the phenomenon we are calling *pragmatic overloading*.

The primary function of an *until clause* is to specify the condition κ under which an agent should terminate the *process* specified in the main clause α. For example, in

(14) *Squeeze riveter handles until rivet stem breaks off.*

the agent is meant to continue the process of squeezing the rivet until he observes that its stem has broken off.

Here *process* is used in Moens and Steedman's sense (1988) of a temporally-extended action with no intrinsic culmination point, what Vendler (1967) called an *activity*. Since *squeeze* has no intrinsic culmination point, an agent needs to determine when he or she can stop and go on to the next thing.

Pragmatic overloading only occurs in a sub-class of utterances containing *until clauses*: those in which the hearer interprets α as both *contributing to* and *controlling* the termination condition κ. In such cases, α will be interpreted as being done for the *purpose* of producing κ, and, as such, if α is underspecified or ambiguous, the hearer will infer a more specific sense of α that will bring about κ. To see this, consider the case where α is not interpreted as either contributing to or controlling κ – e.g.,

(15) *Do whatever you want until your mother gets back.*

or the case where α simply *contributes to* but does not control κ – e.g.,

(16a) *Hold new fixture in position with masking tape until the adhesive has set.*

(16b) *Let poultice stand until it dries.*

In neither of these cases is the hearer led to a more specific interpretation of α *because* it is being done until κ comes about. In particular, since it is exposure to the air that controls setting and drying, not holding or letting stand, the way these actions are done is not affected by their being done until the specified condition holds.

When α is interpreted as contributing to and controlling κ, pragmatic overloading can occur in at least two ways. One way is through description refinement, as in Ex. (5) above and in the following Ex. (17):

(17) *To make sure that all corners are square, measure diagonals AD and BC, and move stake D until the diagonals are equal.*

Because *move stake D* is interpreted as contributing to and controlling the diagonals becoming equal but is underspecified with respect to the direction of movement, the hearer infers that the direction must be that which will make the diagonals equal.

The second way that pragmatic overloading can occur is through an *until clause* directing the hearer to the appropriate aspectual sense of α, for example

(18) *Two 2x4s, worked in opposition, can serve as levers. After loosening boulder with pick and shovel, pry it with one 2x4, then with the other, until you can use one of the levers as a ramp to get stone out of hole.*

Here the aspectual interpretation of the main clause is not simply the two-step sequence of prying with one 2x4 and then prying with the other, but rather an *iteration* of this sequence until the boulder is loose enough to roll out of its hole on one of the 2x4s.

This property, that an *until* adjunct, like a temporal *for* adverbial (e.g. *for 10 minutes*), only makes sense in combination with a process specification has been observed before (Moens and Steedman, 1988; Jackendoff, 1990). But has also been observed (Moens and Steedman, 1988) that an event description can be coerced to a process interpretation in more than one way, e.g.:

(19a) *Play the Moonlight Sonata for a few minutes, just to get an idea what it sounds like.*

(19b) *Play the Moonlight Sonata for six hours, and then see how much you still like it.*

In the first case, the *for* adverbial leads to an interpretation of *play the Moonlight Sonata* in which only a small fragment of the sonata will be played (i.e., it won't be played to completion). In the second case, the *for* adverbial leads to a sense of iteration. It is the hearer's world knowledge of the relevant time periods involved that leads him or her to the appropriate aspectual sense. Similarly, it is the hearer's world knowledge of how actions contribute to and control conditions that leads the hearer to the appropriate aspectual sense of the action description in an *until clause*.

We explain how this happens using Moens & Steedman's event ontology (1988). This posits a structure called an *event nucleus* consisting of a *preparatory process*, a *culmination*, and a *consequent state*. Vendler's *accomplishment* corresponds to a complete nucleus, while his non-durative *achievement* corresponds to a nucleus minus a preparatory process and his atelic *activity* consists only of a preparatory process without culmination or consequent state.

Now hearers know that actions can contribute to and control conditions in different ways. For example, a condition κ can be the cumulative result of a repeated action, as in Ex. (18) above or the result of repeated attempts at an action, each of which has some probability of producing κ, as in

(20) *Try sample specks on the piece until you get a good match, wiping them away each time until you find the right colour.*

In such cases, given an instruction of the form *Do* α *until* κ, if the hearer takes κ as being the cumulative result of repeated instances of α or the result of repeated attempts at α, each of which has some probability of producing κ, then the hearer will interpret α as an iterative process, as in Ex. (18) above.

Alternatively, condition κ can be the result of some unspecified process that can only take place in a state that the specified action α is capable of bringing about; then, α will be interpreted in terms of the process of *maintaining* the consequent state it brings about. The intended interpretation of Ex. (21) is that the agent should turn off the iron and maintain it in that state while the ambient room temperature acts to cool it down:[9]

(21) *If solder gets runny or if iron smokes, turn off iron until it cools a bit.*

There is a third type of aspectual coercion occurring with *until* clauses that occurs only rarely and is illustrated by the following (constructed) example:

(22) *Play "Tenderly" until you get to the part you're having trouble with, then call me and I'll come help.*

Here, playing a specific composition denotes an entire event nucleus, consisting of preparatory process, culmination and consequent state. Yet the interpretation of α here is as an initial sub-sequence of the preparatory process. We speculate that this will happen when a preparatory process is viewed as leading cumulatively to condition κ before the culmination is reached.

The work described in this section is in a more preliminary stage than the work on *purpose clauses*, so we can only speculate on the knowledge representation and reasoning needed to support the particular inference patterns mentioned here. (Other knowledge will be needed as well for relating an agent's knowledge of possible ways of assessing κ – simple perception vs. "active" perception – to the agent's knowledge of ways of doing α, thereby producing an integrated action complex that accomplishes both α and perhaps repeated assessment of κ.) Clearly causal knowledge will be needed to relate actions with their eventual, cumulative, or non-deterministic results. In addition, knowledge will be needed of processes, the states that support them and their time-varying effects. Because such knowledge will serve other roles as well in both text understanding and planning, postulating their eventual availability is not so far-fetched.

4 Reasoning and representation

Here, we will discuss the inferences arising in the context of pragmatic overloading from a computational point of view. Approaching pragmatic overloading from a computational point of view does not only involve designing an algorithm, but also providing a knowledge representation formalism that can support such inferences.

The algorithm we have implemented can only handle purpose expressions relating actions: it cannot relate an action to a condition it is being done to bring about, as in *so that* and *until* adjuncts; and it cannot recognize *purpose* when it is indirectly conveyed in expressions such as *until* clauses. Thus, the algorithm accepts instructions of the form *Do* α $<$ *for the purpose of* $>$ *doing* β:[10] <*for the purpose of*> can either be conveyed explicitly by *to, so as to, in order to*, or implicitly by adjacency. The algorithm will then try to find the connection between α and β, by exploiting the fact that β describes the goal to be achieved.

A second limitation of the algorithm is that it is not able to recognize relations between actions that are not somehow derivable from the underlying prior knowledge. Our current algorithm can compute refinements of action descriptions and expectations if it can recognize that the instructor means to convey a specific generation or enablement relation already known to the agent: it does not yet learn new generation or enablement relations, or recognize temporal relations. For example, if Ex. (12) is interpreted as α = *pour mixture over cheese in casserole* generating β = *spreading [mixture] evenly* and we don't have any previous knowledge about pouring a mixture in such a manner as to spread it evenly, we won't be able to recognize such generation relation and thus we won't be able to compute any contingent refinements. While this is clearly a shortcoming of our current approach, we believe we have laid down some solid foundations that will support the implementation of more complex inference processes in the future.

4.1 The formalism

There are two issues that have been implicitly raised in our discussion so far and that motivate our approach to represent action descriptions.

- The same action can be described in a variety of ways. In the same text, even on the same page, the instructor may use different descriptions to refer to the same action to be performed in the world. Examples from (McGowan and DuBern, 1991) are:

  (23a) *(p. 62) Use masking tape to hold them [each piece] in position.*

  (23b) *(p. 63) Secure each piece in place with masking tape.*

  (24a) *(p. 179) Turn on the tap to drain away the water in the pipe.*

  (24b) *(p. 179) Open the tap to drain the pipe.*

  Our perspective is that we should try to capture as many variations with as few mechanisms as possible; in this paper, we present one of them.

- Talking about the "same" action described in different ways presupposes that for every action there is a canonical description that we can use as the *anchor*. This canonical description can be considered as the representative of the equivalence class of all equivalent descriptions for a certain action α — the representative can be chosen on the practical grounds of what is best suited for the processing that the system performs. Intuitively, there is such a distinction between "external" action descriptions and the "internal" knowledge about actions the agent has. Whatever form this knowledge has — and certainly we don't intend to make any cognitive claims in this respect — the problem arises of reconciling the "external" form of the instruction with the agent's "internal" knowledge. Clearly, an important part of the problem is *not* to infer equivalence when it does not exist. The problem is made more difficult by the fact that the "external" description may be totally new to the agent.[11]

Shieber (1993) makes a similar point as regards generation systems. He points out that the *strategic* reasoner — the module of the system that decides *what* to say — may employ a logical-form language in which there may exist several representations of any given meaning of a certain string; at the same time, the grammar in the *tactical* generator — the module that decides *how* to express a certain meaning — will presumably pair only one such representation, called *canonical* logical form, with that string. Shieber points out that thus the problem of computing the equivalence of any two logical forms arises, and convincingly argues that

> either the strategic component will have to perform linguistic reasoning, or the interface representation language together with the tactical component will constitute a solution to the AI problem.

Clearly, we do not claim to have solved the problem; rather, we provide a step towards the solution, by supplying a knowledge representation formalism flexible enough to support comparing action descriptions one to the other in a perspicuous way.

These two characteristics, the variability of "external" descriptions, and the need to reconcile "external" and "internal" descriptions, led us to two specific choices in our representation formalism: adopting a lexical semantics approach to the representation of verbs, and embedding our representation in a description logic based system (once called *hybrid* systems or systems based on *terminological logics*).

A decompositional approach to the lexical semantics of verbs (Jackendoff, 1990; Levin and Rappaport Hovav, 1992; Levin and Rappaport Hovav, 1995) focuses attention on components of meaning that affect both surface behavior and interpretation, and allows us to capture some descriptional variations: more importantly, it helps us infer whether two descriptions describe the "same" action, in particular by blocking potentially wrong inferences. For example, (Levin and Rappaport Hovav, 1995) postulate a fundamental dichotomy between *manner/means* verbs, such as *wipe* and *stab*, and *result* verbs, such as *remove* and *kill*. The *location-as-object* variant is possible only with (some) *manner/means* verbs, and not with *result* verbs, as exemplified in (25) and (26) (from (Levin and Rappaport Hovav, 1992)).

(25a) *Wipe the fingerprints from the counter.*

(25b) *Wipe the counter.*

(26a) *Remove the groceries from the bag.*

(26b) *Remove the bag.*

While (25a) and (25b) may describe the same action, (26b), while a grammatical English sentence, can't have the same meaning as (26a). Notice however that nothing in the surface structure of the descriptions allows the inference from (25a) to (25b) and prevents the identical one from (26a) to (26b). Only by differentiating between these two classes of verbs via their lexical semantic representations can we ensure that inferences are allowed and prevented in accordance to the facts. Moreover, the importance of the *manner/means* components of meaning is also shown by the abundance of modifiers expressing them, such as *evenly* in (12), in NL instructions.

Other descriptional variations are captured by the description logic based system (DLBS) we use, CLASSIC (Brachman et al., 1991). The choice of a DLBS neither happened by chance nor is simply motivated by implementation concerns:[12] a DLBS provides appropriate Knowledge Representation tools to support an algorithm that must compare different action descriptions, and deal with those that are not known to the system. The basic representation in a DLBS is a virtual lattice of concepts, action descriptions in our case, based on the partial order induced by subsumption. Subsumption captures hierarchical relation between action descriptions, and concept classification, the algorithm that computes subsumption, provides the appropriate inferential basis to deal with different action descriptions.

The reasoning required to deal with pragmatic overloading also requires representing other relations between actions, including *generation*, *enablement*, *substep*, *qualifier* and *effect*. For this, our formalism consists of two components, the *Action Taxonomy* and the *Plan Library*, both implemented in CLASSIC. The Action Taxonomy stores lexical semantic knowledge about actions, using primitives derived from Jackendoff's decompositional approach to the lexical semantics of verbs (1990; 1991). The action terms defined in the Action Taxonomy are the components of the *recipes*, i.e. *common sense* plans about actions (Pollack, 1990),[13] stored in the Plan Library. The recipes in the Plan Library are the knowledge the system has about actually planning.

We are not claiming that we are able to recognize every descriptional variant of an action. For example, the two descriptions *skillet on the stove* and *skillet in the kitchen* may refer to the same object, but are not simply descriptional variants. Understanding their relationship requires spatial reasoning, which the current implementation carries out in an *ad hoc* manner. However, we do believe that the two conceptual components we have chosen, lexical semantics and DLBS, provide the core of an appropriate representation system, to which other representation and reasoning systems are joined — see Sec. 4.2.

4.1.1 The Action Taxonomy

Some elements that we use to represent the lexical semantics of verbs[14] were drawn from Jackendoff's Conceptual Semantics — CS for short (1990; 1991). We found CS useful for the following reasons. First of all, CS provides a link between surface form and semantics as discussed above with respect to Exs. (25) and (26). A related point is that the CS primitives capture generalizations about action descriptions and reveal more of their relationships to one another, such as that *carry* is *move object* augmented with a specific physical means of moving the object. This makes them suitable for a hierarchical representation. Second, CS representations are particularly amenable to expressing the logical form of an instruction, as in general they reveal where information may be missing from an utterance and must be provided by inference; and in particular, they are well suited as the semantic representation associated to the Combinatory Categorial Grammar used by *AnimNL*'s parser (White, 1992).

A CS entity may be of ontological type *Thing, Place, Path, Event, State, Manner* or *Property*.[15] The CS for a LIBRARY is shown in (27):

(27) $[_{\text{Thing}}\ \text{LIBRARY}]_k$

Square brackets indicate an entity of type *Thing* the enclosed featural description. Indexes such as k, which we will often omit for the sake of readability, are used to distinguish instances of a type.

CS's may also contain complex features generated by conceptual functions over other CS's. The conceptual function IN: Thing → Place is used to represent the location *in the library* as shown in (28a) below. Likewise, the function TO: Place → Path describes a path that ends in the specified place, see (28b). Finally, by adding GO: Thing × Path → Event, we obtain the representation for *Go into the library*, as shown in (28c):

(28a) $[_{\text{Place}}\ \text{IN}([_{\text{Thing}}\ \text{LIBRARY}]_k)]_l$

(28b) $[_{\text{Path}}\ \text{TO}([_{\text{Place}}\ \text{IN}([_{\text{Thing}}\ \text{LIBRARY}]_k)]_l)]_m$

(28c) $[_{\text{Event}}\ \text{GO}_{\text{Sp}}([\text{YOU}]_i, [_{\text{Path}}\ \text{TO}([\text{IN}([\text{LIBRARY}]_k)]_l)]_m)$

An important notion in CS is that of *semantic field*, i.e., the semantic domain to which a certain primitive applies. For example, a CS primitive like GO embodies the concept of change of state, and the semantic field specifies in which domain the change takes place: it can be a change of location as in (28c) (Sp stands for Spatial), or a change of ownership as in (29) (Poss for Possessional), which represents part of the meaning of a sentence such as *Bill bought a watch*.

(29) $[\text{GO}_{\text{Poss}}([\text{WATCH}], [\text{TO}([\text{AT}([\text{BILL}])])])]$

In *AnimNL* we introduced a new semantic field, called Control (Ctrl), intended to represent the notion of *having control over* some object, relevant to any action involving direct physical manipulation. For example in sports, the meanings of *having the ball*, *getting the ball*, and *losing the ball* embody this notion. Ex. (30) illustrates both the similarity and the difference between *Jack has the money* (Possessional) and *Jack has the ball* (Control):

(30a) $[\text{BE}_{\text{Poss}}([\text{MONEY}], [\text{AT}([\text{JACK}])])]$

(30b) $[\text{BE}_{\text{Ctrl}}([\text{BALL}], [\text{AT}([\text{JACK}])])]$

CS's are readily integrated into CLASSIC, as shown in Figs. 3, 4 and 5, that present part of the Action Taxonomy in our system. As customary in graphical representations of KB's based on DLBS's, ellipses represent *concepts*; small circles encircling squares represent *roles*, i.e. relations between concepts: the numbers in parentheses, such as `(1,1)` represent the arity of the relation; and diamonds containing the '=' sign represent constraints that impose equality of role fillers — this is the restricted type of *structural map*, namely, of relation between fillers of different roles, that CLASSIC allows.

Some comments on the elements in the figures are in order.

Entity. The taxonomy rooted in *entity* is similar to others used in other KB's.

Place. This subhierarchy encodes conceptual functions of the form F: Thing → Place, such as AT, IN, ON. There are different kinds of places and paths corresponding to different semantic fields.[16] In Fig. 3, only the concept *spatial-place*, and its subordinate *at-sp-place*, corresponding to the AT conceptual function, are shown. *at-sp-place* has a single role `at-role` with exactly one filler, of type *Entity*.

Path. Concepts belonging to this subhierarchy represent functions yielding *Paths*. Consider *from-to-path-sp(atial)*, defined by means of multiple inheritance from *to-path-sp* and *from-path-sp*. *from-to-path-sp* has

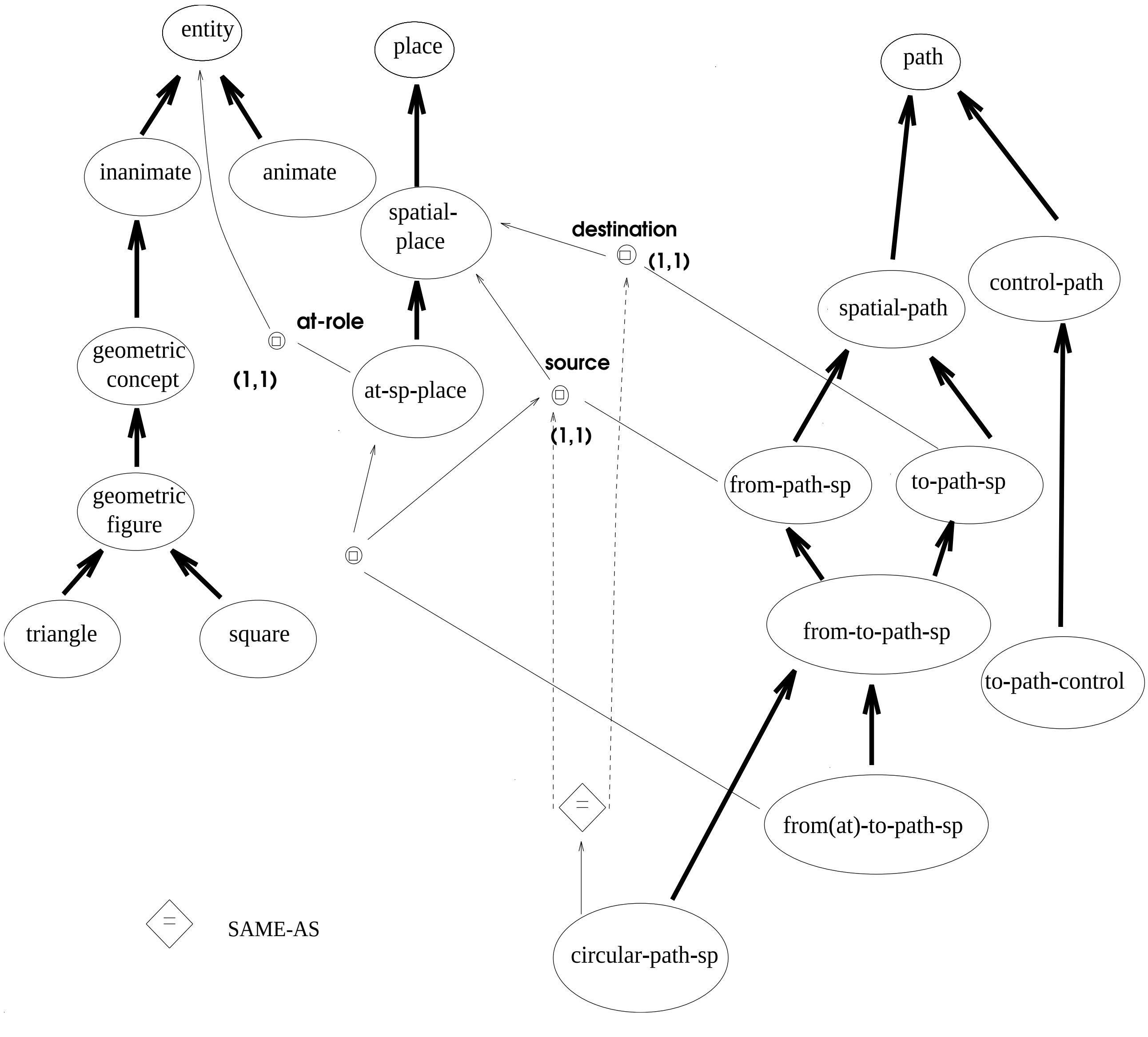

Figure 3: Parts of the *Entity, Place* and *Path* Hierarchies

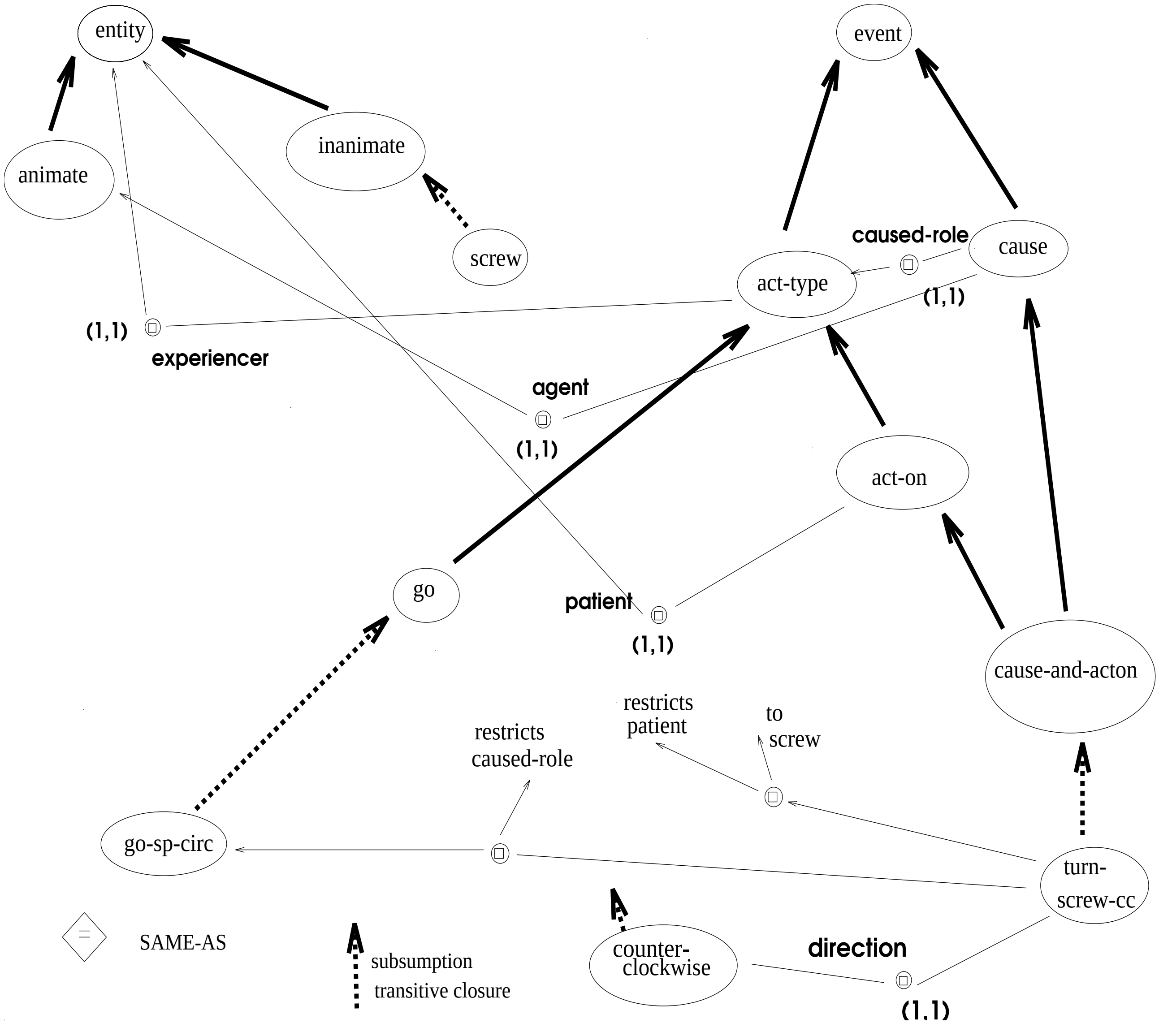

Figure 4: Part of the *Action* Hierarchy

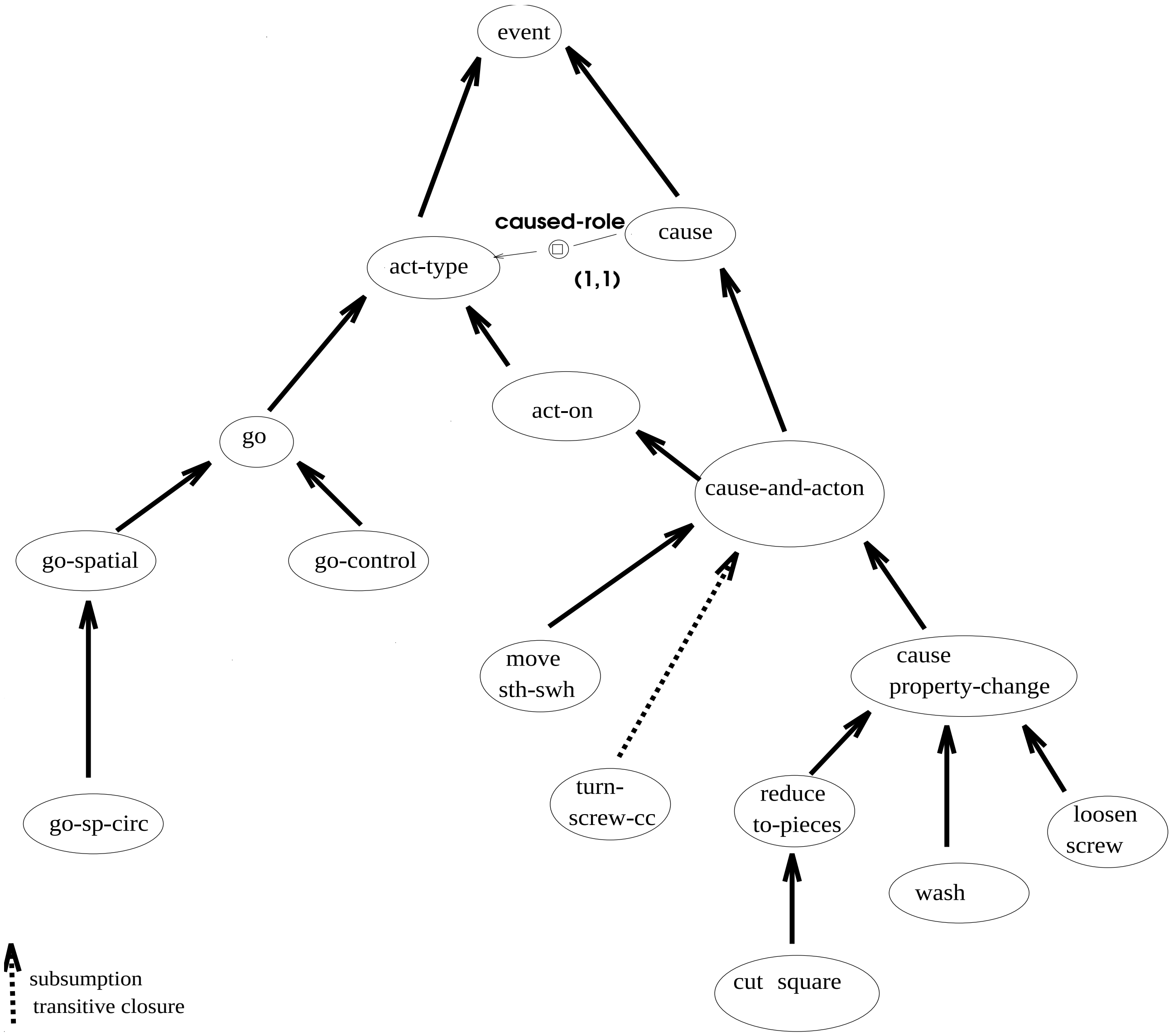

Figure 5: Some actions in the *Action* Hierarchy

two roles, `source` and `destination`, each of which has a filler *spatial-place*. The concept *from(at)-to-path-sp* restricts the role `source` inherited from *from-to-path-sp* to be filled by *at-sp-place*.

Event. Fig. 4 shows part of the event subhierarchy. The intuitive notion of *action* corresponds to the concept *cause-and-acton*.[17] To avoid cluttering the figure, some restrictions on *cause-and-acton* are not shown: namely, *cause-and-acton* restricts the two roles `agent` inherited from *cause* and `experiencer` inherited from *act-on* to have the same filler; analogously, `patient` inherited from *cause* and `experiencer` inherited from *cause* via `caused-role` are restricted to have the same filler, as shown in the following CLASSIC definition:

```
(cl-define-concept 'cause-and-acton
                   '(and cause act-on
                         (same-as agent experiencer)
                         (same-as patient (caused-role experiencer))))
```

turn-screw-cc — for *turn screw counterclockwise* — is defined as a subconcept of *cause-and-acton* by imposing the restriction that the filler of the `caused-role` be *go-sp-circ*, namely, an *act-type* of type *go-spatial*, with a role *path-role* restricted to be *circular-path-sp*. The `experiencer` role on *go-sp-circ* is the turned object, i.e. the concept *screw* restricting the *patient* role on *turn-screw-cc*. The definition of *turn-screw-cc* exactly corresponds to its CS representation, shown as the body in Fig. 10. Finally, Fig. 5 is a schematic version of a portion of the *event* subhierarchy, that includes 44 concepts — *move-sth-swh* stands for *move something somewhere*. *cause-property-change* is the root of the subhierarchy which includes actions that change a property of an object. Consistent with Jackendoff's terminology, the semantic field *Ident*ificational refers to properties: for example, CLEAN and DIRTY in Fig. 7, and LOOSE and TIGHT in Fig. 10 are values of the *Ident* field. CS's such as AT([DIRTY]) in Fig. 7 represents "places" within the *Ident* semantic field.

While the choices of CS's and of DLBS's were independently motivated, they reciprocally benefit from their integration: the usage in the KB of linguistically sound primitives is a first step towards providing a real lexicon; on the other hand, a representation based on description logic makes it possible to use CS's in a computational framework, by endowing it with a hierarchical organization and with the possibility of extending the lexicon.

Before closing, we note that in our use of a DLBS, we are not distinguishing between the T-Box, the repository of terminological knowledge, and the A-Box, used to represent the individual instances of the concepts. There are two reasons for this. First, CLASSIC does not really distinguish between T-Box and A-Box, as the same language is used to define concepts and individuals. Second, while both the Action Taxonomy and the Plan Library make use of CLASSIC terminological components to the fullest, we use the "A-Box" in a limited way: we simply create the individual actions that correspond to the action descriptions in input, and exploit CLASSIC's subsumption mechanism to understand of which concepts they are instances.

4.1.2 The Plan Library

As mentioned above, the second component of our formalism, the Plan Library, contains *planning* knowledge in the form of simple recipes, and is implemented in CLASSIC as well; thus, classification is used to maintain an organized KB of action recipes.

The syntax of the recipes is described in Fig. 6, and two examples of recipes, which respectively illustrate a method for *washing an object* and for *loosening a screw*, are shown in Figs. 7 and 10; these two figures are expressed in CS terms rather than as a CLASSIC concept for readability. The terminals that are not relations in Fig. 6 — *basic-act-type, act-type, state* — are concepts belonging to the Action Taxonomy, and thus indicate type restrictions on components of the recipes. *ACHIEVE* in Fig. 7 maps a state into an action that achieves that state.

RECIPE	→	BASIC-RECIPE | NON-BASIC-RECIPE
BASIC-RECIPE	→	BASIC-HEADER QUALIFIER* EFFECT+
NON-BASIC-RECIPE	→	HEADER BODY
		QUALIFIER* EFFECT+
BASIC-HEADER	→	basic-act-type
HEADER	→	act-type
BODY	→	act-type+ ANNOTATION*
ANNOTATION	→	act-type$_1$ enables act-type$_2$ |
		act-type$_1$ TEMP-REL act-type$_2$
QUALIFIER	→	state
EFFECT	→	state
TEMP-REL	→	precedes | before | meets ...

Figure 6: The Recipe BNF

Header
$[\mathrm{CAUSE}([\textsc{agent}]_i, [\mathrm{GO}_{\mathrm{ident}}(j,k)])]$ $\left[\ \mathrm{TO}([\mathrm{AT}([\textsc{clean}])])\ \right]_k$
Body
- $[\mathrm{ACHIEVE}(i,\ \mathrm{BE}_{\mathrm{Sp}}(i,\ [\mathrm{AT}([\textsc{washing-site}])]))]_{\gamma_1}$
- $[\mathrm{ACHIEVE}(i,\ \mathrm{BE}_{\mathrm{Sp}}(j,\ [\mathrm{AT}([\textsc{washing-site}])]))]_{\gamma_2}$
- $[\mathrm{PHYSICAL\text{-}WASH}(i,\ j,\ [\mathrm{AT}([\textsc{washing-site}])])]_{\gamma_3}$
- Annotations -
- γ_1 *enables* γ_3
- γ_2 *enables* γ_3
Qualifiers
- $[\mathrm{BE}_{\mathrm{Ident}}(j,\ \mathrm{AT}([\textsc{dirty}]))]$
Effects
- $[\mathrm{BE}_{\mathrm{ident}}(j,\ \mathrm{AT}([\textsc{clean}]))]$

Figure 7: A *Wash* Recipe

Recipes have a *header*, *body*, *qualifiers* and *effects*. The terminology, especially *header* and *body*, is reminiscent of STRIPS, but the relations between these components are expressed in terms of *enablement* and *generation*, e.g. the body *generates* its header.

The distinction between *basic-recipe* and *non-basic-recipe* is due to the need of providing the base case

Given *Do* α *for the purpose of doing* β,

(31a) using β as an index into the plan library, find a collection of recipes $\mathcal{M}_l$ that achieve β;

(31b) *match* α to an action $\gamma_{l,j}$ that appears in the body of $\mathcal{M}_l$;

(31c) choose the best matching $\mathcal{M}_l$.

Figure 8: High level description of the algorithm

of the representation, namely, of defining *basic act-types*. It is a notoriously difficult problem to define what a *basic act-type* is — cf. (Pollack, 1986, p.59). For the purpose of this paper, we will assume that CS Event functions that don't have another event as argument are basic: this implies that all the act-types which are descendants of GO are basic. Moreover, those action types that, as discussed above, cannot be distinguished solely on the basis of their CS representations, such as PHYSICAL-WASH in Fig. 7, are left as basic: it is *AnimNL* that provides the corresponding decomposition into lower level actions.

The representation does not employ preconditions, and thus, action recipes express a part of what is traditionally expressed by means of preconditions by means of actions, which are substeps in the body that generates a certain header. Other functions that preconditions have been used for, such as ordering substeps in a plan, can be performed by means of the *annotations* on the body, that specify the relations between the subactions, e.g. enablement and temporal relations. One of the reasons behind the choice of not having preconditions is the fact that NL instructions generally describe an action α, rather than α's effects: thus a representation focused on substeps rather than on preconditions keeps the mapping process between surface form and stored knowledge more direct. Another reason for not having preconditions is the difficulty of distinguishing between preconditions and substeps in the body of an action. See (Di Eugenio, 1993; Webber et al., 1992) for further details.

Qualifiers are those conditions on actions that must hold for an action to be relevant, and are not meant to be achieved. Procedurally, qualifiers don't give rise to subgoaling. Finally, a recipe has *effects*, what must be true after an action has been executed.[18]

Finally, there may be many recipes with the same header, e.g. the recipe in Fig. 7 is just one of those possible for washing an object: for example, another one (that we haven't included in our system yet) could describe washing an object by having another agent, possibly a professional, wash it.

This definition of recipes can be translated quite directly into CLASSIC — see (Di Eugenio, 1993) for further details.

4.2 The algorithm

At a high level, our algorithm can be described as in Fig. 8. The inferences due to pragmatic overloading occur during step (31b): the refinement of α to α' is performed by means of a flexible match between α and $\gamma_{l,j}$, which is one of the actions that appear in the recipe that achieves β. Notice that instead step (31a) can be considered as belonging to plan recognition inferences, in that one of its side effects is that new actions, not mentioned in the input — namely, the ones belonging to $\mathcal{M}_l$ — may be included in the agent's plan. Discussion of the third step of the algorithm (31c) can be found in (Di Eugenio, 1993).

The inferences performed in steps (31a) and (31b) all exploit CLASSIC subsumption through its classifier. Step (31a) is performed by retrieving recipes of whose header β is an instance. Step (31b) can be concisely described as checking the characteristics of the concept (and α^{conc} $\gamma_{l,j}$), where α^{conc} is the most specific, possibly virtual concept of which α is an instance.

Although the algorithm in Fig. 8 may appear quite abstract, it is embedded in *AnimNL* in the modules labeled *plan graph initialization* and *plan inference*, as shown in Fig. 9. Instructions are given to *AnimNL* in *steps* consisting of one or more utterances. A step specifies a continuous behavior that the agent must attend to. Steps are processed by a parser (White, 1992) based on Combinatory Categorial Grammar, and that produces a logical form LF. The LF for the current step is incrementally developed into the *plan graph*, which is composed of nodes that contain descriptions of individual actions, and edges that denote relations between these actions. The algorithm in Fig. 8 has as input the LF produced by the parser, which is expressed in terms of CLASSIC individuals, one for each for the main clause and for each adjunct clause, plus the necessary connectives. As regards purpose expressions, the algorithm we have so far (which obviously can interpret also simple main clauses, even if the description in Fig. 8 is tailored towards purpose expressions) assumes that the goal β has been identified: this is straightforward for Purpose Clauses, where β is explicit; the goal can be easily recognized in certain cases of *until clauses*, in which the condition κ is stated in the passive form — e.g. *until the screw is loosened* — or with a modal, either active or passive — e.g. *until you can remove the tile* or *until the tile can be removed*. However, this assumption is too strong to hold in general for *until* clauses, and doesn't work for *free adjuncts*, which, as we saw, don't necessarily convey a purposive relation. Heuristics can help: for example when a free adjunct is headed by a verb such as *create* in (13), *make, form*, it is plausible that the free adjunct expresses a goal. Clearly, heuristics won't solve the whole problem, and more sophisticated search strategies are necessary.

The algorithm in Fig. 8 models the commitments the agent adopts simply based on the input instruction and the stored knowledge and produces a first pass of the plan graph, that is further developed by processes of *reference resolution*; *plan expansion* — e.g. if the plan calls for moving from one room to another, this step will insert a step *open door* if the door between the two rooms is closed; *referent finding* — the referent in the world for a certain referring expression may not be immediately available, e.g. in *Get me a soda* the soda will presumably be in the fridge; *object specific reasoning* — verbs are often underspecified with respect to the geometric characteristics of the object: e.g. *open* is used both for *open a can of paint* and *open the door*, but the movements that realize these two actions are quite diverse. When a commitment to act for a particular purpose, e.g. *goto(door1, open(door1))* "go to door1 for the purpose of opening it", becomes sufficiently specified for the agent to be ready to commit to it and temporal dependencies permit such commitment, other, low-level planning processes are triggered. The output of these processes is a collection of *behaviors* to be executed (simulated) in parallel, or an indication that the agent's "body" is unable to carry out the behavior.

A final remark on our KB and algorithm: both the Action Taxonomy and the Plan Library are small-scale knowledge bases. Scaling them up, while requiring more sophisticated indexing and search, will not, we believe, affect the basic algorithm.

We will now illustrate how the inferences about refining action descriptions are computed by step (31b). Space constraints prevent us from illustrating how the *Contribute* relation is refined by means of expectations — the interested reader is referred to (Di Eugenio, 1993).

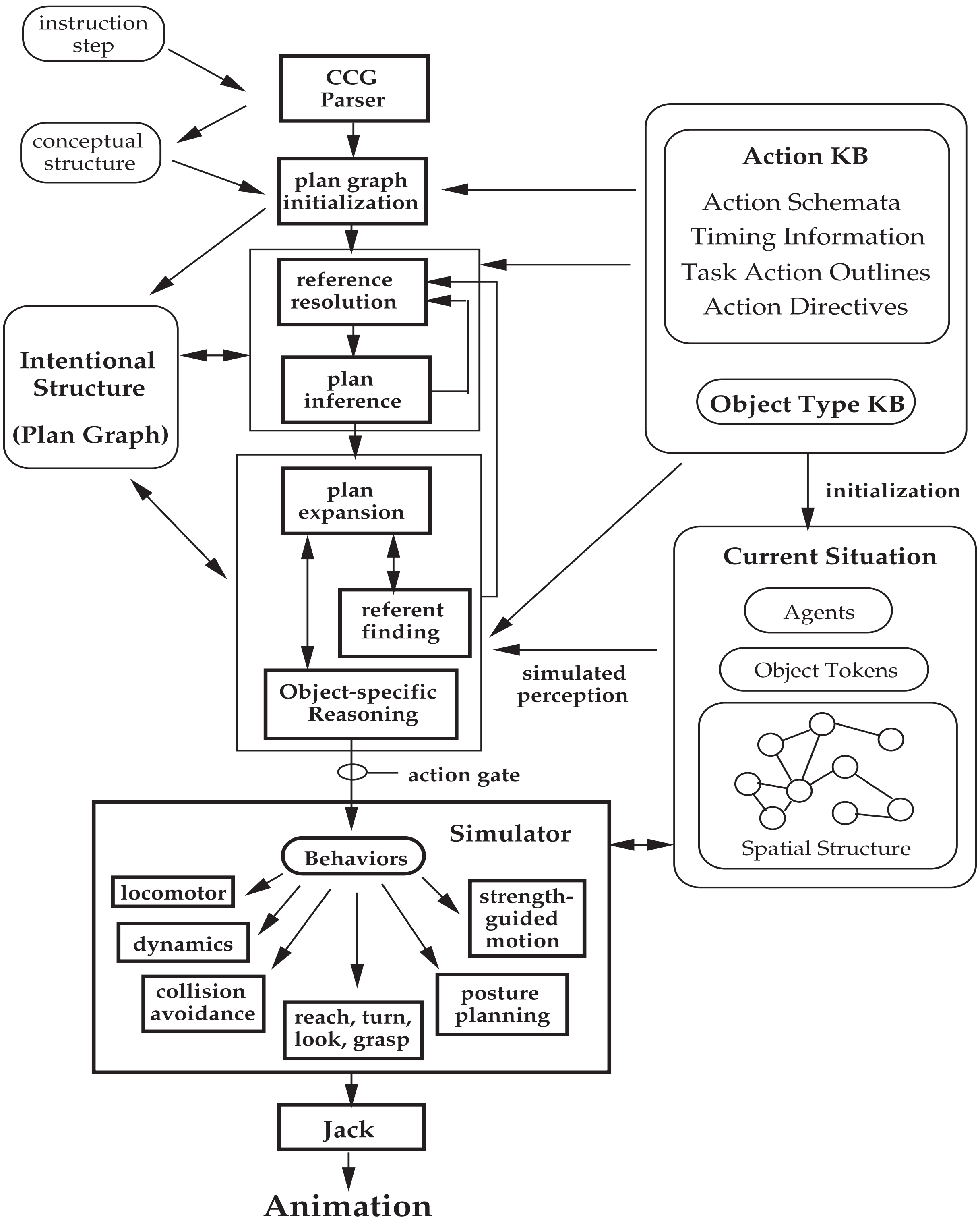

Figure 9: AnimNL System Architecture

4.2.1 Refining action descriptions

Let's consider Ex. (7), *Turn screw to loosen*. As we discussed above, α = *Turn screw* is underspecified in that the direction in which to turn the screw is missing; the goal β = *to loosen* provides such direction as *counterclockwise* (under the assumption the screw is right threaded). Notice that there are a very large number of ways of expressing such an instruction; four of them are listed in (32), where (7) is repeated as (32b):[19]

(32a) *[To loosen the screw]*$_{\beta_1}$, *[turn it counterclockwise with the big screwdriver]*$_{\alpha_1}$.

(32b) *[Turn the screw]*$_{\alpha_2}$ *[to loosen]*$_{\beta_2}$.

(32c) *[To loosen the screw]*$_{\beta_3}$, *[turn it with the big screwdriver]*$_{\alpha_3}$.

(32d) *[Turn the screw clockwise]*$_{\alpha_4}$ *[to loosen]*$_{\beta_4}$.

Now, in all cases apart from (32a), α_i undergoes some refinement, brought about by β_i through the match with $\gamma_{l,j}$, γ for short — in this case, the only step in the body of the recipe in Fig. 10.[20] Such recipe is retrieved by step (31a) for each of the cases in (32). Step (31b) is implemented by asking the following queries in succession, stopping as soon as a positive answer is found — α^{conc} is the most specific, possibly virtual concept of which α is an instance.

(33a) `(and` α^{conc} γ`)` $\stackrel{?}{=}$ α^{conc}, i.e., does γ subsume α^{conc}?

(33b) `(and` α^{conc} γ`)` $\stackrel{?}{=}$ γ, i.e., does α^{conc} subsume γ?

(33c) `(coherent (and` α^{conc} γ`))`, i.e, do α^{conc} and γ have a common subsumee?

Header

$$[\mathrm{CAUSE}(i, \mathrm{GO}_{\mathrm{ident}}(\textsc{screw}_j, k))] \begin{bmatrix} \mathrm{FROM}(\mathrm{AT}(\textsc{tight})) \\ \mathrm{TO}(\mathrm{AT}(\textsc{loose})) \end{bmatrix}_k$$

Body

$$\begin{bmatrix} \mathrm{CAUSE}(i, \mathrm{GO}_{\mathrm{sp}}(j, n)) \\ [\mathrm{DIRECTION}(\textsc{counterclockwise})] \end{bmatrix}_\gamma \begin{bmatrix} \mathrm{FROM}(m) \\ \mathrm{TO}(m) \end{bmatrix}_n$$

Effects

$$[\mathrm{BE}_{\mathrm{ident}}(j,\ \mathrm{AT}([\textsc{loose}))]$$

Figure 10: The recipe for *loosen screw*

1. In (32a), α_1 is more specific than γ in the recipe, and thus undergoes no refinements: $\alpha_1' = \alpha_1$. However, this is established only *after* the added modifier *with the big screwdriver* is checked for consistency with γ: this is verified by (33a) that returns a positive answer.

2. In (32b), α_2 is more abstract than γ. However, only γ is specified enough to achieve β_2, thus α_2 is refined to $\alpha'_2 = \gamma$. This is illustrated in Fig. 11. α_2^{conc}, the virtual concept of which α_2 is an instance, is an ancestor of γ: thus α'_2 is inferred by (33b) to be an instance of γ, with the roles common to γ and to α_2^{conc} filled with the fillers from α_2.

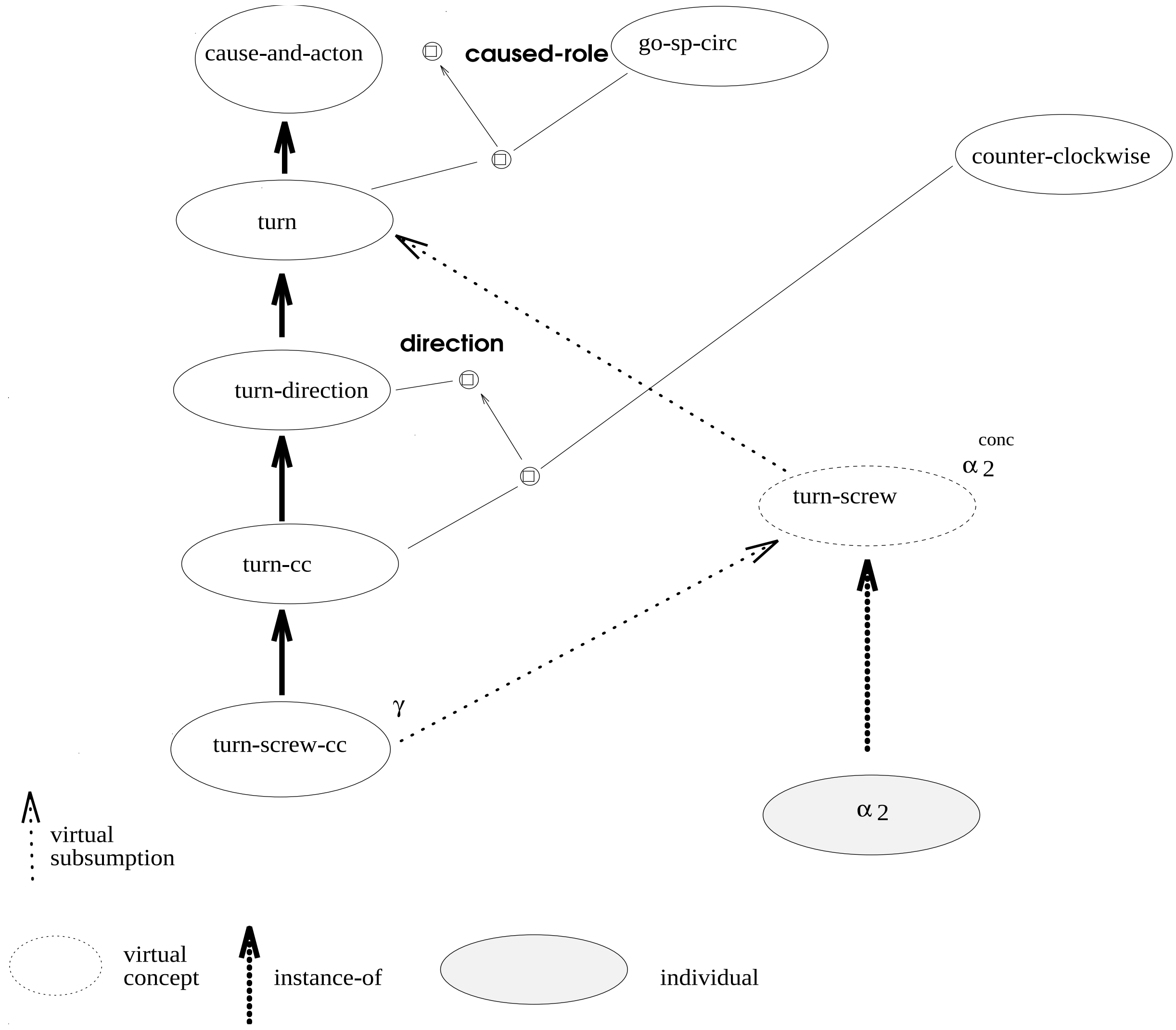

Figure 11: A less specific *turn*

3. In (32c), with regard to γ, α_3 lacks the direction of rotation, but adds the instrument modifier *with the big screwdriver*, compatible with everything else we know about γ and β. In this case, (33c) verifies whether `(and` α_3^{conc} γ`)` is coherent: as it is, α'_3 is taken to be an instance of this latter virtual concept, with the roles inherited from α_3^{conc} filled with the fillers on α_3.

4. In (32d), α_4 is inconsistent with γ, as found out by (33c), and no α'_4 is admissible — see Fig. 12. The value restrictions of the role `direction` on the two concepts are respectively `counterclockwise` and `clockwise`, two concepts defined as disjoint and thus incoherent. Clearly, the incoherence of (32d) depends on the available knowledge. Suppose that the system also has a recipe $\mathcal{M}_{left}$ about left

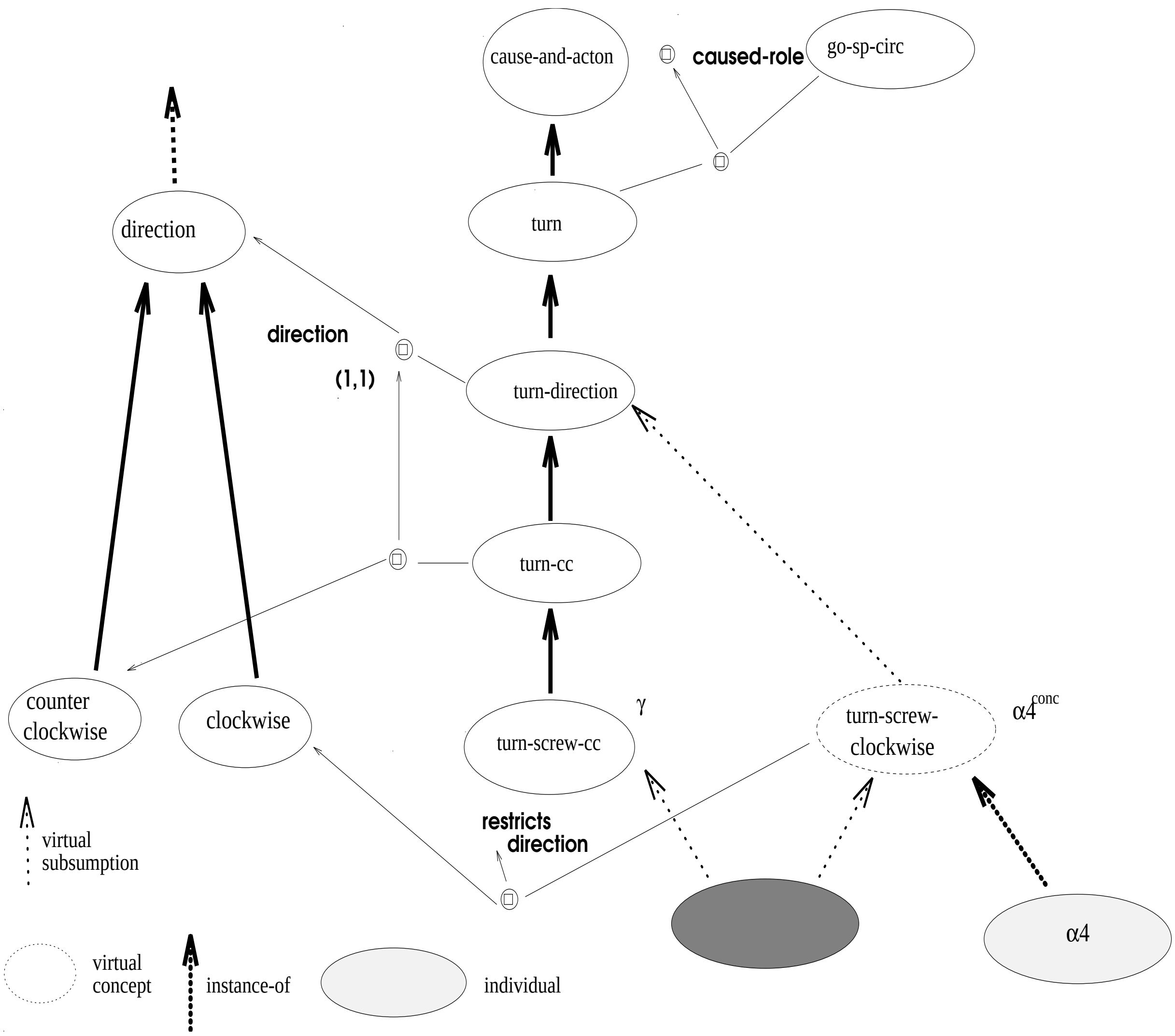

Figure 12: An incoherent *turn*

handed screws, that loosen in the opposite way. β_4 — and so all the other β_i in (32) — would then select both $\mathcal{M}_{left}$ and $\mathcal{M}_{right}$ (the recipe in Fig. 10). As a consequence, α_4, rather than being deemed incoherent, would be found compatible with the γ in the body of $\mathcal{M}_{left}$: the matching step could in fact be used to select the right recipe. This is a plan recognition type inference, that our algorithm is able to perform, but that we don't discuss any further in this paper because it does not belong to pragmatic overloading.[21]

5 Conclusion

There are two things we would like to do in this final section: re-iterate our main point and indicate how our research is currently proceeding. We start with the latter.

As we already noted, we do not currently carry out any reasoning from conditions to the actions that control their coming about. This capability will be needed for at least two purposes: to interpret *so that* and *until* adjuncts in general and to derive refinements due to pragmatic overloading of these adjuncts in particular. Animating instructions containing *until* adjuncts also requires knowledge of the actions and resources used in assessing these conditions, since termination condition assessment actions must be appropriately integrated with the main action the agent is instructed to carry out. This is discussed in somewhat more detail in (Webber, 1995).

Reasoning about the nature of the process that an agent must carry out until some condition holds also requires a finer-grained representation of action than one usually finds in action representations for AI planning and certainly one finer-grained than that which we are currently using. One that suggests itself is Steedman's version of the situation calculus (1995), which can capture both process termination and action culmination. We are currently looking into a mapping between the parallel transition networks (*PaT-Nets*) that serve as an executable high-level specification for animation in *Jack* (Becket and Badler, 1993; Badler et al., 1995) and Steedman's representation. Both are process-oriented, and the mapping would support the ability to reason in Steedman's representation and execute a corresponding set of PaT-Nets (possibly with an optimizing stage mediating the two, to eliminate unnecessary actions and ensure smoother movement).

Work on the matching step (31b) in Fig. 8, to integrate spatial reasoning with CLASSIC-style subsumption, is also part of our future plans when resources become available.

It would also be beneficial to consider *pragmatic overloading* from the point of view of language generation and try to relate it to previous work on generating referential noun phrases (NP), where content becomes included in an NP for purposes other than picking out the speaker's intended referent (Appelt, 1985).

In closing, we want to re-iterate our take-home point: the current view of "language as action" leads us to consider analogous mechanisms in the two. In the case of action, Pollack has called attention to cases where intentions are *overloaded* to achieve multiple goals. This suggests similar mechanisms in language. We have tried to show how one such mechanism, which we have called *pragmatic overloading*, might work, illustrating it with examples of *purpose clauses*, *free adjuncts* and *until clauses*. We believe our work highlights the value of instructions as a source of data in studies of Natural Language pragmatics: in short, they provide a well-defined basis (i.e., what you need to know in order to act appropriately) for Gricean judgements of *relevance* and reasoning triggered by relevance (Wilson and Sperber, 1986).

Acknowledgements

Thanks to Joseph Rosenzweig for collecting the (RD, 1991) and (McGowan and DuBern, 1991) materials. We acknowledge him, Mark Steedman, Matthew Stone, Keith Vander Linden and three anonymous reviewers for their comments on earlier drafts of the paper. This research is partially supported by ARO DAAH04-94-G-0426, DMSO DAAH04-94-G-0402, and ARPA N66001-94-C-6043.

Notes

[1]We will use the terms *speaker* or *instructor* for the agent issuing the instructions, and *hearer* or *agent* for the agent carrying them out. For ease of exposition, we will use feminine pronouns to refer to the speaker, and masculine ones to refer to the hearer.

[2](Suppes and Crangle, 1988) have also talked about implicit constraints on procedure execution associated with instructions:

> Expressed intentions carry with them a bundle of *ceteris paribus* conditions that impose a variety of constraints on the specific procedures actually executed. [p. 319]

They have not, however, taken the same direction as we have, to derive some of these *ceteris paribus* conditions in constrained ways from a combination of lexical semantics and knowledge about acting in the world.

[3]We are 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 we have concentrated on are adjoined to a matrix clause, and are termed *rationale clauses* in syntax; in fact all the data we will discuss in this paper belong to a particular subclass of such clauses, subject-gap rationale clauses.

[4]In general, action descriptions are underspecified in many respects; as we discussed in Sec. 1, the additional components of meaning that we illustrate in this paper crucially depend on the goal β that α contributes to. There may be other additional components of meaning which are relevant to the execution of α, e.g. the amount of force to apply to the screw in (7). Those relevant component of meanings that are not computed by inferences derived from pragmatic overloading are derived by other modules of the *AnimNL* system, see Sec. 4.2.

[5] β = *Treat badly corroded brass* is underspecified in itself, as *treat* could mean many different things: however, this instruction belongs to a section of (RD, 1991) that deals with cleaning metals.

[6]It is true that, because of the uniqueness presupposition associated with definite referential NP's, (9a) implies that there is only one Italian dictionary in the library; however, the same expectation would arise if the NP were indefinite.

[7]A point similar to the distinction between *conditional generation* versus *generation* needs to be made for *conditional enablement* versus *enablement*.

[8]Constructions headed by subordinating conjunctions and containing a nonfinite verb, such as *while fighting in France, he was taken prisoner* are not considered to be free adjuncts by Stump (1985), who calls them *augmented adjuncts*.

[9]The above analysis differs from one given in (Moens and Steedman, 1988), where it is suggested that the combination of a *for* temporal adverbial with an achievement, as in

(34) *John left the room for a half hour.*

expresses *intention* rather than duration, observing that Ex. (34) would be true even if John is only out of the room for an instant, returning immediately to get his umbrella. A similar analysis does not seem appropriate for the *until* clauses given above.

[10]We are slightly abusing our own notation here. Up to now we have been using α to denote the action description in the main clause, and β to denote the action description in the adjunct. Under this interpretation, and by saying that the algorithm interprets instructions of the form *Do* α *"for the purpose of" doing* β, it follows that the action in the adjunct, β, is the goal: this is not necessarily true in free adjuncts. For the sake of brevity, we will keep using *Do* α *<for the purpose of> doing* β, but the reader should keep the in mind that in the case of free adjuncts the input to the algorithm may be *Do* β *<for the purpose of> doing* α.

[11]Basically, the terms "external" and "internal" refer to the input surface form, that for us includes the logical form, and to the (semantic / world) knowledge an agent has about actions: these two terms are used here to highlight the fact that there is often a mismatch between the two kinds of knowledge.

[12]Our choice of DLBS, CLASSIC as opposed to e.g. LOOM (MacGregor and Burstein, 1991), was determined by practical considerations such as ready availability, ease of installation, efficiency, etc. Thus, we are not advocating CLASSIC in particular, but rather, DLBS's in general.

[13]We are using the term *recipe* in the sense of Pollack's distinction between *recipes* and *plans* (1990): *recipes* are what an agent knows about how to perform a certain action or achieve a certain goal, while *plans* are what an agent adopts in order to act. An agent may know the recipe about how to rob a bank, without ever adopting it as one of his plans.

[14]So far we have concentrated on action representation, and we don't deal with issues related to the representation of object descriptions.

[15]This taxonomy is based on (Jackendoff, 1990). (Jackendoff, 1991) adds *Time* and *Amount* to the ontological types.

[16]In CLASSIC the semantic field is represented by defining a role `semfield-role` — not shown in the figures — whose value restriction is the concept *semfield* defined by enumeration.

[17]Such definition is used to maintain the distinction between the *thematic* and *action* tiers that Jackendoff argues for in (1990).

[18]We don't expect recipes to be complete, as in general neither the qualifier nor the effect list is exhaustive: they both merely list some necessary conditions. We refer the reader to e.g. (Genesereth and Nilsson, 1987) for discussion of the related issues of the *qualification* and *frame* problems in AI.

[19]The really occurring example is (32b); the PC is preposed or not according to the "heaviness" of the main clause — see (Vander Linden and Martin, 1995).

[20]Such recipe also illustrates the lack of aspectual knowledge in our representation: the facts that *turn counterclockwise* may need to be repeated more than once, and that *loosen screw* is a process that may have different culminations, as a screw may be more or less loosened, are missing.

[21]Clearly, if the system's knowledge includes both $\mathcal{M}_{left}$ and $\mathcal{M}_{right}$, all of the other examples in (32) will also yield different results: (32a) will still select $\mathcal{M}_{right}$, while (32b) and (32c) do not provide enough information by themselves to select either recipe.

References

Allen, James. 1984. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23:123–154.

Appelt, Douglas. 1985. *Planning English Sentences.* Studies in Natural Language Processing. Cambridge University Press.

Badler, Norman, Welton Becket, Barbara Di Eugenio, Christopher Geib, Libby Levison, Michael Moore, Bonnie Lynn Webber, Michael White, and Xinmin Zhao. 1993. Intentions and expectations in animating instructions: the AnimNL project. In *Intentions in Animation and Action*. Institute for Research in Cognitive Science, University of Pennsylvania, March 11-12.

Badler, Norman, Bonnie Lynn Webber, Welton Becket, Christopher Geib, Michael Moore, Catherine Pelachaud, Barry Reich, and Matthew Stone. 1995. Planning for animation. In D. Thalmann and N. Maginet-Thalmann, editors, *Computer Animation*. Prentice Hall Inc.

Badler, Norman I., Cary B. Phillips, and Bonnie Lynn Webber. 1993. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press.

Balkanski, Cecile. 1993. *Actions, Beliefs and Intentions in Multi-Action Utterances*. Ph.D. thesis, Harvard University. Technical Report TR-16-93.

Becket, Welton and Norman Badler. 1993. Proc. workshop on computer generated forces and behavior representation. In *Integrated behavioral Agent Architecture*.

Brachman, Ronald, Deborah McGuinness, Peter Patel-Schneider, Lori Alperin Resnick, and Alexander Borgida. 1991. Living with CLASSIC: When and How to Use a KL-ONE-like Language. In John F. Sowa, editor, *Principles of Semantic Networks — Explorations in the Representation of Knowledge*. Morgan Kaufmann, pages 401–456.

Chopra, Sonu. 1994. Strategies for Simulating Direction of Gaze and Attention. University of Pennsylvania, May.

Di Eugenio, Barbara. 1993. *Understanding Natural Language Instructions: a Computational Approach to Purpose Clauses*. Ph.D. thesis, University of Pennsylvania, December. Technical Report MS-CIS-93-91 (Also Institute for Research in Cognitive Science report IRCS-93-52).

Doran, Christine. 1993. Purposive AND Clauses in Spoken Discourse. Unpublished manuscript, University of Pennsylvania.

Geib, Christopher. 1995. *The Intentional Planning System (ItPlans)*. Ph.D. thesis, University of Pennsylvania, May.

Genesereth, Michael R. and Nils J. Nilsson. 1987. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers.

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

Grice, H.P. 1975. Logic and Conversation. In P. Cole and J.L. Morgan, editors, *Syntax and Semantics 3. Speech Acts*. Academic Press.

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

Hallowell, Alice Rich, editor. 1988a. *Tile, Remodeling Handbook*. Sunset Books. Lane Publishing Company.

Hallowell, Alice Rich, editor. 1988b. *Wall Coverings*. Sunset Books. Lane Publishing Company.

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

Jackendoff, Ray. 1991. Parts and boundaries. *Cognition*, 41:9–45.

Kautz, Henry. 1990. A Circumscriptive Theory of Plan Recognition. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*. MIT Press.

Levin, Beth and Malka Rappaport Hovav. 1992. Wiping the slate clean: a lexical semantic exploration. In Beth Levin and Steven Pinker, editors, *Lexical and Conceptual Semantics, Special Issue of Cognition: International Journal of Cognitive Science*. Blackwell Publishers.

Levin, Beth C. and Malka Rappaport Hovav. 1995. *Unaccusativity : at the syntax-lexical semantics interface.* MIT Press.

Lewis, David. 1979. Scorekeeping in a Language Game. *Journal of Philosophical Language*, 8:339–359.

MacGregor, Robert and Mark H. Burstein. 1991. Using a Description Classifier to Enhance Knowledge Representation. *IEEE Expert*, 6(3):41–46.

McGowan, J. and R. DuBern, editors. 1991. *Home Repair.* London: Dorlin Kingersley Ltd.

Moens, Mark and Mark Steedman. 1988. Temporal Ontology and Temporal Reference. *Computational Linguistics*, 14(2):15–28.

Moore, Johanna and Martha Pollack. 1993. A Problem for RST: the Need for Multi-Level Discourse Analysis. *Computational Linguistics*, 18(4):573–544.

Pollack, Martha. 1986. *Inferring Domain Plans in Question-Answering.* Ph.D. thesis, University of Pennsylvania.

Pollack, Martha. 1990. Plans as Complex Mental Attitudes. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication.* MIT Press.

Pollack, Martha. 1991. Overloading Intentions for Efficient Practical Reasoning. *Noûs*, 25:513–536.

1991. Reader's Digest New Complete Do-It-Yourself Manual.

Shieber, Stuart M. 1993. The Problem of Logical Form Equivalence. *Computational Linguistics*, 19(1):179–190.

Steedman, Mark. 1995. Dynamic Semantics for Tense and Aspect. In *IJCAI95, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, August.

Stump, Gregory. 1985. *The semantic variability of absolute constructions.* D. Reidel Publishing Company.

Suppes, P. and C. Crangle. 1988. Context-Fixing Semantics for the Language of Action. In J. Dancy, J. Moravcsik, and C. Taylor, editors, *Human Agency: Language, Duty, and Value.* Stanford University Press, pages 47–76.

Vander Linden, Keith and James Martin. 1995. Expressing Local Rhetorical Relations in Instructional Text. *Computational Linguistics*, 21(1):29–57.

Vendler, Zeno. 1967. *Linguistics in Philosophy.* Ithaca NY: Cornell University Press. (Chapter 4. Verbs and Times).

Webber, Bonnie Lynn. 1995. Instructing animated agents: viewing language in behavioral terms. In *Proceedings of the International Conference on Cooperative Multi-modal Communication*, Eindhoven, Netherlands, May.

Webber, Bonnie Lynn, Norman Badler, F. Breckenridge Baldwin, Welton Becket, Barbara Di Eugenio, Christopher Geib, Moon Jung, Libby Levison, Michael Moore, and Michael White. 1992. Doing What You're Told: Following Task Instructions In Changing, but Hospitable Environments. Technical Report MS-CIS-92-74, University of Pennsylvania.

Webber, Bonnie Lynn, Norman Badler, Barbara Di Eugenio, Christopher Geib, Libby Levison, and Michael Moore. 1995. Instructions, Intentions and Expectations. *Artificial Intelligence, Special Issue on Computational Theories of Interaction and Agency*, 73, February.

Webber, Bonnie Lynn and Barbara Di Eugenio. 1990. Free Adjuncts in Natural Language Instructions. In *COLING90, Proceedings of the Thirteenth International Conference on Computational Linguistics*, pages 395–400.

White, Michael. 1992. Conceptual Structures and CCG: Linking Theory and Incorporated Argument Adjuncts. In *COLING92, Proceedings of the Fourteenth International Conference on Computational Linguistics*, pages 246–252.

Wilson, Deirdre and Dan Sperber. 1986. Inference and Implicature. In C. Travis, editor, *Meaning and Interpretation*. Basic Blackwell, pages 45–75.