We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

154
Countries delivered to

Our authors are among the

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

"From Saying to Doing" – Natural Language Interaction with Artificial Agents and Robots

Christel Kemke
University of Manitoba
Canada

1. Introduction

Verbal communication is one of the most natural and convenient forms of human interaction and information exchange. In order for artificial agents and robots to become adequate and competent partners of humans, they must be able to understand and respond to task-related natural language inputs. A typical scenario, which poses the background of our research, is the collaboration of a human and an artificial agent, in which the human instructs the artificial agent to perform a task or queries the agent about aspects of the environment, e.g. the state of an object. The artificial agent has to be able to understand natural language inputs, as far as they concern actions, which it can perform in the environment. The artificial agent can be a physical robot, or a virtual, simulated agent.

Several projects have explored the development of speech and language interfaces for cooperative dialogues with agent-like systems, in particular TRAINS and TRIPS for cooperative route planning (Traum et al., 1993; Allen et al., 1995; 1996); CommandTalk as spoken language interface for military planning (Stent et al., 1999); Situated Artificial Communicators for construction tasks (SFB-360, 2005; Rickheit & Wachsmuth, 2006) and the CoSy project on human-robot interaction (Kruijff et al., 2007; Kruijff, 2006). Other projects dealing with spoken language interfaces include Verbmobil (Wahlster, 1997) and BeRP (Jurafsky et al., 1994).

Inspired by this work, we developed a basic architecture for natural language interfaces to agent systems for the purpose of human-agent communication in task-oriented settings. Verbal inputs issued by the human user are analyzed using linguistic components, semantically interpreted through constructing a formal representation in terms of a knowledge base and subsequently mapped onto the agent's action repertoire. Since actions are the central elements in task-oriented human-agent communication, the core component of this architecture is a knowledge representation system specifically designed for the conceptual representation of actions. This form of action representation, with an aim to bridge the gap between linguistic input and robotic action, is the main focus of our research. The action representation system is based on taxonomic hierarchies with inheritance, and closely related to Description Logic (Baader et al., 2003), a family of logic-based knowledge representation languages, which is becoming the prevalent approach in knowledge representation.

In order to define the formal action representation system, we combine aspects of action descriptions found in computer science (formal semantics for program verification; functional programming); mathematics (logic, structures, functions); linguistics (verb-oriented semantics; case frames); and artificial intelligence, especially formal methods in knowledge representation (Description Logic), planning and reasoning (STRIPS, Situation Calculus), as well as semantic representation and ontology.

The result of this endeavor is a framework suitable to:

- represent action and object concepts in taxonomic hierarchies with inheritance;
- instantiate object concepts in order to describe the actual world and the environment of the agent system;
- instantiate action concepts to describe concrete actions for execution through the agent/robot;
- provide structural descriptions of concepts in the form of roles and features, to connect to the linguistic processing components through case frames;
- provide a formal semantics for action concepts through precondition and effect formulas, which enables the use of planning and reasoning algorithms;
- support search methods due to the hierarchical arrangement of concepts;
- provide a connection to the action repertoire of artificial agent systems and robots.

We describe the formal structure of the action representation and show how taxonomies of actions can be established based on generic action concepts, and how this is related to object concepts, which are stored in a similar way in the hierarchy. The inheritance of action descriptions and the specialization of action concepts are illustrated in examples, as well as the creation of concrete actions. For further details on the exact formalization, we refer the reader to related literature discussed in section 2. We explain the basics of the natural language interface and the semantic interpretation through case frames. We outline the connection of the knowledge representation, i.e. the action (and object) hierarchy to the natural language interface as well as to the artificial agent system, i.e. arbitrary robots. An important aspect is the grounding of action concepts in the object world, which is crucial in case of physical agents. Other specific advantages of the action taxonomy relevant in this context are the support of retrieval and inference processes, which we will address briefly. In the last sections, we give an illustrative example using the CeeBot language and present some completed projects, which have been developed using the described interface architecture and the action representation methodology. Finally, we provide an informal assessment of the strengths and weaknesses of the approach and an outlook on future work.

2. Knowledge Representation

One core issue of our research is to provide a sound, formal framework for the representation of actions, which can be used as a basis for the development of adaptable speech and language interfaces for agent systems, in particular physical agents, i.e. robots. A crucial point in the development of such interfaces is a representation of the domain actions and objects in a format, which is suitable to model the contents of natural language expressions and to provide a connection to the agent systems as well.

2.1 Background and Related Work

The representational framework we discuss here is based on structured concepts, which are arranged in a taxonomic hierarchy. A formal semantics defines the meaning of the concepts, clarifies their arrangement in the hierarchy, captured in the subclass/superclass relationship, and the inheritance of descriptions between concepts within the hierarchy. The representational approach we use was motivated by KL-ONE, a knowledge representation formalism first proposed by Brachman and colleagues (Brachman & Schmolze, 1985), and is closely related to Term Subsumption Languages (Patel-Schneider, 1990) and the more recently studied class of Description Logic (DL) languages (Baader et al., 2003), which are both derived from KL-ONE. These representation languages are based on the same principles of structured concept representation in taxonomic hierarchies, focusing on a clearly defined formal semantics for concept descriptions, as well as encompassing the classification of concepts and the inheritance of concept descriptions within the hierarchy.

Description Logic languages were conceived in the first instance to represent static concepts and their properties. Dynamic concepts, like actions and events, which are defined through change over time, were considered only later in selected works. The problem of representing action concepts in DL and similar taxonomic representations is to provide a formal description of the changes caused by an action, which can be integrated into the formalism of the base language. Secondly, a clear formal semantics has to be provided and classification and inheritance algorithms have to be defined.

A first approach towards an integration of action concepts into KL-ONE and taxonomic hierarchies in general was developed by the author in the context of a help system for the SINIX operating system; the approach was also motivated through the development of an action ontology for command-based software systems (Kemke, 1987; 2000).

Other relevant work on action concepts in KL-ONE and DL employed a simpler, more restricted formalization of actions similar to STRIPS-operators (Lifschitz, 1987)., describing the effects of an action through ADD- and DELETE-lists; these lists are comprised of sets of literals, which are basic predicate logic formulas (Weida & Litman, 1994; Devanbu & Litman, 1996), which has been embedded into the CLASSIC knowledge representation system. However, there is no direct connection between the actions and an environment model; the representation of actions is not well-integrated into the representational system and thus the semantics of action concepts is not well-grounded. An extension and application of this approach, with similar drawbacks, has been described by (Liebig & Roesner, 1997). Other work on action concepts in DL dealt with composite actions and specified required temporal relations between actions and sub-actions, forming an inclusion or decomposition hierarchy (Artale & Franconi, 1994; 1998). The crucial issues of action classification and inheritance, however, were not addressed in this work. Di Eugenio (1998), in contrast, provided a well-designed, structured representation of action concepts, including relations to object concepts, for describing instructions in the context of tutoring systems. This form of DL based action representation is similar to the one developed independently by Kemke (Kemke, 1987; 1988; 2003), who outlines a formal semantics for action concepts based on the notion of transformation of world models. Baader et al. (2005) suggest another approach to model actions in DL but they start with a description of concrete actions, in a STRIPS-like fashion. While they provide a thorough theoretical analysis regarding the computational complexity of their algorithms, their approach suffers from the same lack of grounding as the earlier work by Litman and colleagues mentioned above.

2.2. Knowledge Representation and Ontology

When working on a certain application of a natural language interfaces to an agent system, the first task is to elicit and model relevant aspects of the domain and describe them in terms of the knowledge representation system. The outcome is a domain ontology, which is used to implement a knowledge base for the application domain. We focus here on knowledge representation through concepts, and distinguish *object concepts*, representing physical or virtual objects in the domain, and *action concepts*, which are modeled conceptually at various levels of complexity or abstraction and provide a connection to the human user and her verbal instructions (complex, abstract actions), as well as to actions of the robotic system (concrete, low level actions).

The structure of the domain is internally represented based on a knowledge representation format closely related to Description Logics, as outlined above. Classes of objects of the domain are modeled through "object concepts" and are represented in a hierarchical fashion, with general concepts, like "physical object", at higher levels, and more specific concepts, like "tool" and "screwdriver" at intermediate levels, and even more special object classes, like "flathead screwdriver size 8" at lower levels.

object:	screwdriver
superclass:	tool
type:	{flathead, philips, pozidriv, torx, hex, Robertson, triwing, torqset, spannerhead}
size:	[1,100]
color:	colors

The hierarchical connection between concepts is a pure superclass/subclass or subsumption/specialization relation. Concepts in such hierarchies typically inherit all properties from their higher level concepts, e.g. "screwdriver" will inherit all descriptions pertaining to "physical object". Therefore, such hierarchies are also referred to as "IS-A" or "inheritance" hierarchies. We prefer to call them "taxonomies".

2.3 Object Concepts

Descriptions of object concepts (and also action concepts) in the representational system contain in addition connections to other concepts in the hierarchy, used to reflect relationships between concepts, called "roles" in DL, consisting of relations between objects from either concepts, and attributes, also called "features", which represent functions applied to an object from one concept, yielding an object from the other concept as value. Examples of roles are the "has-part" relation between a physical object and its components, like "cylinder" as part of a "car-engine", or spatial relations between physical objects, like "besides" or "above". Examples of features of physical objects are "color" or "weight".

Certain roles and features of objects can be marked as fixed and not changeable, while others are modifiable and thus form a basis for action specifications. For example, in an automobile repair system, an engine needs a specific number of cylinders and thus the relation between the engine and its cylinders cannot be changed, even though a cylinder might be replaced with a new one. On the other hand, there are items which are not essential to the operation of the vehicle, like the backseats. The seat could be removed without influencing the car's basic operations. This distinction between essential and non-essential characteristics parallels the notion of fluents in situation calculus, whose values can

change due to the execution of an action. Persistent features are useful as reference points or baseline for assurances about the domain.

2.4 Action Concepts

The formalization of action concepts combines two representational forms, satisfying different demands: a case frame representation, which allows an easy connection to the linguistic input processing module, and a logic-based representation to model preconditions and effects of an action through special features, which is helpful to establish the connection to the agent system, since it supports action selection and execution processes and allows the use of standard planning and reasoning algorithms.

The format we developed for representing action concepts thus provides a frame-like, structural description of actions, describing typical roles pertaining to the action, like the "source" location of a *move*-action, or the "object" to be moved. This kind of information is derived from linguistics and can be easily used to present some form of semantics (Fillmore, 1968; Baker et al., 1998). Current ontologies, like WordNet (CSL Princeton, 2007) or Ontolingua (KSL, 2007), typically use such roles in structural descriptions of verbs or actions.

The example below shows the frame-structure for a *grab*-action, using a grasper as instrument. Parameters to this action are "agent", "object", and "instrument". The concepts to which these parameters pertain are shown in the specification, e.g. the agent is a robot and the instrument is a grasper. The concept "liftable-object" denotes physical objects, which are not attached nor non-moveable but can be picked up.

grab <agent, object, instrument>
agent: robot
object: liftable-object
instrument: grasper

For formal reasoning and planning methods, however, this representation is insufficient. We thus provide in addition a formal semantics of action concepts using preconditions and effects. Actions are then described in a format similar to functional specifications, with a set of parameters, which refer to object concepts, their roles, features and possibly values for them. The functionality is specified through restricted first-order predicate logic formulas, where the precondition-formula constrains the set of worlds, in which the action is applicable, and the effects-formula describes the resulting modification of the earlier world state. These formulas involve the parameter objects as predicates, functions, and constants, and describe changes caused by the action as modification of role or attribute values of the affected object. We complete the *grab*-action from above with those formulas:

The precondition-formula above states that the grasper is not holding any objects, i.e. the relation named "holds" for grasper is empty, and that the grasper is close to the object, modeled through the fuzzy spatial relation "close". Relations are stored explicitly in the knowledge base, and this condition can be checked instantly, if an action is to be executed.

grab   <agent, object, instrument>
agent: robot
object: liftable-object
instrument: grasper
precond: $holds\,(grasper, \_\,) = \varnothing \wedge close\,(grasper, liftable\text{-}object)$
effect: $holds\,(grasper, liftable\text{-}object)$

The effect of the action is that the grasper holds the object. The terms "grasper" and "object" have the status of variables in these formulas, which will be instantiated with a specific grasper (of the respective robot) and the specific liftable-object to be picked up.

It should be noticed that the detailed modeling of actions, their preconditions and effects, depends on the domain and the specific robot. If, for example, a robot of type "roby" can pick up only items, which weigh less than 500 grams, we can specialize the action above accordingly for a "roby" type robot:

grab <agent, object, instrument>

agent: roby

object: liftable-object

instrument: grasper

precond: holds (grasper, _) = $\varnothing \wedge$ close (grasper, liftable-object)

                        $\wedge$ weight (liftable-object) < 0.5kg

effect: holds (grasper, object)

The occurrence of a specific *grab*-action, e.g. "grab <roby-1, sd-1, grasper-2>" involves the specification of the parameters, i.e. a specific robot named "roby-1" of type "roby", a specific liftable-object with identifier "sd-1", and a specific "grasper" named "grasper-2" of roby-1. The referenced object classes or concepts, like "grasper", specify the possible role fillers or function values; this is checked, when the action is instantiated. It will not be possible, for example, to grab an object, which is not classified as "liftable-object", or use a "wheel" instead of a "grasper".

For the action execution, the formula describing the precondition is checked, using the specific instances and values of the parameters. With these values, the execution of the action can be initiated, and the result is determined through the effect-formula. The effect-formula describes the necessary modifications of the knowledge base, needed to update the actual world state.

It is worth mentioning that, in the case above, we do not need to retract the formula "holds(grasper,_) = $\varnothing$" as is the case in standard planning or reasoning systems, like STRIPS, since these formulas are evaluated dynamically, on-demand, and the formula "holds(grasper,_)" will not yield the empty set anymore, after the object has been picked up. This is one of the significant advantages of the connection between actions and objects, which our approach provides.

The action representations in the ontology are described in such a way, that some level or levels of actions correspond to actions in the agent system, or can be mapped onto actions of the agent system. Since the action format defined and used in the ontological framework here includes precondition and effect representations, it allows planning and reasoning processes, and thus enables the connection to a variety of agent systems with compatible action repertoires.

2.5 Objects, Actions, and World States Concepts – Grounding and Restricting

As we have seen above, action concepts are arranged in a conceptual taxonomic hierarchy, with abstract action concepts at the higher levels and more concrete and detailed concepts at lower levels. Action concepts themselves are structured through features and roles, which connect them to object concepts. Object concepts model the physical environment and serve as parameters to actions, as in ADL (Pednault, 1989). They refer to those kinds of objects,

which are affected by the execution of an action. The restriction of parameters of actions through references to object concepts also grounds action specifications, i.e. connects them to the physical environment model, and helps to avoid absurd action specifications, which can never be fulfilled.

Instantiated objects are derived from respective object concepts, and they have specific values for features and roles, instead of references to other concepts. They represent concrete domain objects, and are also part of the knowledge base. This part is often referred to as the A-Box (Assertions), which stores information about facts and individuals in the world, in contrast to the T-Box (Terminology), which stores conceptual, terminological knowledge. The set of instantiated objects corresponds to the current world state. Effects of an action can be described through changing the feature or role values of these objects. We use special features for preconditions and effects, which have restricted predicate logic formulas as values. These formulas are constructed using only items from the knowledge representation system, i.e. all predicates, functions, variables, and constants refer to terms in the knowledge base, like concepts, features, roles, and their values.

This allows the description of dynamic effects of actions as changes to the current world state, on a conceptual level for generic action concepts as well as on a concrete level for instantiated actions to be executed.

For further details on the formal semantics of action taxonomies, we refer to (Kemke, 2003), and for a related semantic based classification of action concepts and discussion of the ontology to (Kemke, 2001); implementation examples can be found in (Kemke, 2000; 2004; 2006).
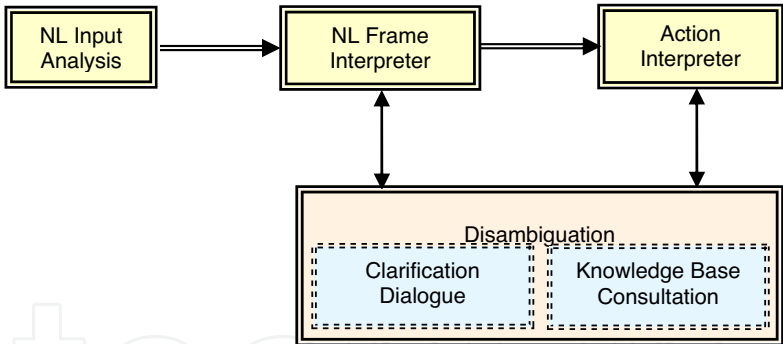
3. Global System Architecture

Figure 1. Global System Architecture

The processing within the general system architecture, which connects the natural language interface to the agent system, is illustrated in figure 1. It shows the main components of the system, including the *Natural Language Input Analysis*, the *Natural Language Frame Interpreter*, and the *Action Interpreter*. In addition, two modules for resolving ambiguities in the verbal input or in the stated request are shown: the *Clarification Dialogue* module, and the *Knowledge Base Consultation*.

3.1 Overview of the Processing

The natural language inputs considered here, which manifest the verbal communication with artificial agents, comprise typically commands, questions, and possibly declarative statements, as well as questions and confirmations as part of clarification dialogues between the system and the user.

The natural language input is firstly processed with a standard natural language parser, based on the Earley algorithm. We developed a base grammar, which specifically models the above mentioned sentence types and related sentence structures. The chosen sentence types reflect the essential speech acts needed for the intended human-agent communication, like *command*, *question*, *statement*. They are similar to and can be considered a subset of the *Agent Communication Language (ACL)* suggested by FIPA (2002).

The structural representation of the verbal input is generated by the parser using typical grammatical constructs, which elicit noun phrases, verb phrases, prepositional phrases etc. This structural representation is then processed in a shallow syntactic-semantic analysis, which selects and determines the contents of a case-frame representation of the linguistic input, based on the sentence type and the main syntactic constituents of the sentence, i.e. the subject noun phrase, object noun phrase(s), prepositional phrases for locations etc. plus an indicator for the queried part of the frame in case of questions.

3.2 NL Input Analysis

We discuss the natural language input processing using the following example command sentence:

"*Bring the red screwdriver from the living-room*."   (example 1)

A command sentence is typically characterized through a leading verb, followed by complements to this verb, like an object noun phrase; prepositional phrases further describing the object or action, like the object's location or the source or destination of an action, e.g. from where or to where to bring something.

For the example sentence above, the NL Input Analysis yields the following structure:

sentence-type: command
verb: bring
direct-object: NP1 (det *the*) (adj *red*) (noun *screwdriver*)
source: PP1 (prep *from*) (det *the*) (noun *living-room*)

The *parser* accesses during processing a *lexicon*, which stores words relevant to the domain and their synonyms, in order to map different words with similar meanings to a standard form. The major part of the lexicon is constituted by verbs, nouns, adjectives, prepositions, and adverbs. The lexicon contains also information about necessary and optional complements to verbs, reflecting the so-called "verb subcategorization". Verb-categories together with required, optional or no complements are modeled through grammar rules, and this information is taken into account during parsing. For example, verbs like "*bring*" require a direct object or "theme" (what to bring) and an explicitly or implicitly specified destination or "recipient" of the bring-action (where-to or to whom to bring the object); furthermore, "*bring*" can have in addition an (optional) source specification (from where to bring the object).

3.3 Frame Interpreter

This structure is now being used by the *Frame Interpreter* to construct a case-frame representation, which brings the input closer to a semantic representation and a mapping onto the knowledge base. In the example above, which is related to some natural language

action: bring
theme: screwdriver(sd-1) ∧ color(sd-1)=red
location: living-room
destination: location(speaker)

interfaces we developed for simulated household robots (see section 5), the direct-object or *theme* is specified through the phrase *"the red screwdriver"*, the location of the object and source of the *bring*-action is the living room, and the destination is implicitly assumed to be the speaker's location.

The term "sd-1" acts like a logic constant and refers to a specific knowledge base object, representing the screwdriver addressed in the command. Descriptions of objects like "screwdriver" with features like "color" are provided through object concepts and their descriptions through roles and features in the knowledge base; they serve in the action-description as predicates (for concepts and roles) and functions (for features), respectively.

3.4 Disambiguation

The case frame representation might be ambiguous and allow several interpretations, which have to be resolved by the NL Frame Interpreter or the Action Interpreter, in order to derive one unique interpretation of the NL input in terms of the knowledge base. We encounter ambiguities or under-specification in several forms. On the linguistic level, we find lexical and structural ambiguity. Lexical ambiguity refers to the phenomenon that a single word can have different meanings; for example, "bank" can refer to a bench to sit on, or to an institution, which deals with money (like "Royal Bank of Canada"), or to a specific building, which houses a branch of such institution (e.g. the Royal Bank building on Pembina Highway).

In example 1 above, the word "screwdriver" could refer to a tool or to a cocktail drink. Which meaning is the preferred one can depend on the task domain. If our system is supposed to support trades people in the construction phase of a house, the interpretation of "screwdriver" as tool would be obvious. This restriction can be modeled on the lexical level, by using a preferred word set to match the input words, and the lexical mapping mentioned above would take care of this ambiguity by excluding unlikely word meanings. If this is, however, not the case, we can use other context information to resolve the ambiguity. If we assume, for example, that we deal with a homeowner doing renovation work, the interpretation of "screwdriver" as a tool might be more suitable - although this is not necessarily the case, since the homeowner might be stressed out from her renovation work and is in need of an alcoholic beverage.

A second way to resolve this ambiguity is to check the "world", i.e. we can consult the knowledge base and see whether it can find a matching item (drink or tool) in the living-room, or the robot could start a search process to figure out, whether it can find one or the other in the living-room. The additional information from the input sentence that the screwdriver is red can be used to identify the respective object in the current world model, which is the part of the knowledge base consisting of all object instances.

An easier way, which can also be applied if there is more than one "red screwdriver" in the living-room, and the system cannot determine, which one is meant, is a clarification dialogue with the user, in which the system can ask for further information, which it can use to identify the screwdriver.

The phrase "from the living-room" could be considered as ambiguous, since it can be either attached to the object-NP (i.e. the screwdriver from the living-room) or considered as part of the verb-NP, serving as modifier of the bring-action (bring _ from the living-room). Such structural requires a clarification dialogue with the user, where the system asks the user for the correct interpretation, or alternatively the system can pick either of the options.

3.5 Action Interpreter

The system thus constructs a complete instruction in case frame format from the NL input, which can then be translated into an action to be executed by the robotic agent. The processing includes several reasoning steps, and the result is an instantiated frame representation of the action, with references to necessary parameter-objects involved in the performance of the action. This representation can be passed over to the agent-system for execution. The agent system might still have to do further planning or reasoning processes but the action-frame is provided in such a format, that any agent system based on standard action-representations with pre-conditions and effects specified in a logic format should be able to deal with it. In the case of physical agents, like robots, they can be connected using an additional controller or interpreter, which transforms this representation and generates actions in the specific format required by the agent or robot.

Obviously, the action description must be fitted to the robot's action repertoire; further detail may have to be added or the action may have to be decomposed into smaller sub-actions, in order to provide a precise instruction (or set of instructions), which are in the robot's action repertoire and can thus be performed by the robot. We developed an algorithmic approach to work with integrated action abstraction and action/plan decomposition hierarchies (Walker, 2004; Kemke & Walker, 2006), in particular a smart search algorithm, which accesses simultaneously both hierarchies, in order to construct a suitable plan by combining pre-defined plan schemata from the decomposition hierarchy and generic actions from the abstraction hierarchy. The idea is essentially to check actions in the abstraction hierarchy, whether they fulfill a given task or goal description, and to switch over to the plan hierarchy, if no primitive action can be found; vice versa, if a rough plan can be located in the decomposition hierarchy, it can be instantiated with more specific actions taken from the abstraction hierarchy. This approach combines the advantages of using pre-defined plan schemata to model complex actions in an easy and efficient way, and the well-structured organization of abstraction hierarchies, which require significantly less time for search processes than standard lists.

4. CeeBot Example

An example of mapping actions to the instructions of a virtual robot using the CeeBot language is shown below. CeeBot is an educational system, developed by Epsitec SA for teaching programming languages to children and adults of all ages, using virtual robots as application (Epsitec SA, 2007). The CeeBot system is suitable as a test application for

knowledge representation and natural language interfaces to robots, since it is easy to use but still sufficiently complex. It is also very inexpensive compared to real physical robots.

The CeeBot language provides, for example, an action "turn left", which is realized through an instruction to the motors of the left and right wheels. A left turn can be expressed in CeeBot as follows:

action: turn left *instruction:* motor(-1,1)	*left motor reverse, right motor forward*

In our action representation formalism, we can be represent "turn" in various forms, as shown in the small action hierarchy below.

The first action-concept implements a generic *turn*-action with two possible values for the direction of the turn, which can be left or right. The turn changes the orientation of the robot, which is given in degrees in a base coordinate system, with a still unspecified value. In the knowledge base, the orientation would be a part of the description of a robot using a feature (function).

action:	turn <direction>	*generic low level turn-action;*
direction:	{left, right}	*"direction"-parameter with*
precond:	orient (agent) = x	*values 'left' or 'right'*
effect:	orient (agent) = y	

The two action-concepts below are specializations of the turn-action above. The values of the direction have been further restricted, for one action to "left", for the other action to "right". This describes the action concepts "turn left" and "turn right". The effect can now be determined precisely by adding or subtracting 90 degrees from the original value of the robot's orientation.

action:	turn <left>		*action:*	turn <right>	
direction:	left		*direction*:	right	
precond:	orient (agent) = x		*precond*:	orient (agent) = x	
effect:	orient (agent) = x+y ∧ y=-90		*effect:*	orient (agent) = x+y ∧ y=90	

A more general form of representing turns, which can also be used to derive the above specifications, is to allow values between -180 and +180 degrees for a turn.

action:	turn <direction>
direction:	[-180, +180]
precond:	orient (agent) = x
effect:	orient (agent) = x+y

This specification can also be used to describe "fuzzy expressions", for example, a "*turn slightly left*", using a fuzzy mapping yielding a value between 0 and -90 degrees for the direction of the turn.

The representation can also include a decomposition of complex actions, i.e. a representation of complex actions and their decomposition into a sequence of simpler sub-actions as addressed in section 3.4. For example, if the human "master" instructs the robot to bring a certain object, the action generated to describe this task on a higher level could be: bring <robot, object, master>; this can be decomposed into a more detailed, finer grained action sequence, e.g. goto <robot, object>; grab <robot, object>; goto <robot, master>.

action:	bring <robot, object, master>
agent:	robot
theme:	object
recipient:	master
precond:	none
effect:	close (robot, master) ∧ holds (robot, object)

The decomposition includes the following sub-actions:

action#1:	goto <robot, object>
agent:	robot
destination:	location (object)
precond:	none
effect:	close (robot, object)

action#2:	grab <robot, object>
agent:	robot
theme:	object
precond:	close (robot, object)
effect:	holds (robot, object)

action#3:	goto <robot, master>
agent:	robot
destination:	location (master)
precond:	none
effect:	close (robot, master)

The final outcome is that the robot is close to the master "close(robot, master)" and holds the object "holds(robot, object)". The formula "close(robot, object)", which is the effect of the first action, will be overwritten due to the effect of the last action "close(robot, master)".

5. Complete Prototype Systems

Several prototype systems have been developed with different degrees of complexity regarding their natural language capabilities, and varying levels of sophistication regarding their knowledge representation and reasoning components. All these systems have in addition been equipped with a speech input and output facility, except for the toy car - which does not yet speak.

5.1 Interior Design System

This system is a prototype of a combined natural language and visual scene creation system, inspired by the WordsEye system, which generates images based on verbal descriptions (Coyne & Sproat, 2001). The current implementation simulates an interior design system, which models actions and questions related to the positioning of furniture in a room, and includes the addition of furniture to the scene as well as the removal of furniture from the scene. The system works with speech input processed through the DragonTM speech recognition system. It performs a syntactic and semantic analysis of the recognized verbal input and produces a *request-frame* representing the user's query or command. The query or command is processed through accessing a Description Logic style knowledge base.

The system is implemented in two versions, one with a knowledge base implemented in PowerLoom[1], and the other one with a knowledge representation and reasoning module in CLIPS[2]. Both represent domain objects and actions in a conceptual framework based on inheritance hierarchies. In addition, the knowledge representation features spatial relations and respective spatial rules and constraints, like transitivity or asymmetry of spatial

---

[1] www.isi.edu/isd/LOOM/LOOM-HOME. html

[2] www.ghg.net/clips/CLIPS.html

relations and constraints, e.g. that large objects cannot be put on top of small objects. The system also integrates commonsense rules, like a preference to specify relations in the natural language output with respect to static inventory like windows, doors or a fireplace.

5.2 Virtual Household Agent

One of the earliest prototype projects is a simulated household agent, which interprets and performs tasks issued by a user in written language. The household agent moves in a virtual world, which includes, among other things, a fridge, a dishwasher, a TV, a sofa (for the "Master"), a mop and a bucket (for cleaning the floor), and random dirt. The agent operates in auto-mode, as long as there is no user input, and will do regular chores like cleaning the floor. The agent has some kind of "self-awareness" and returns to its charging station whenever it notices that its batteries are low in power. The user (Master) can issue command statements in natural language to the agent, for example, to switch on the TV or get a drink or food item, like "Bring me a glass of milk".

In case of a verbal command input, the household agent analyzes the natural language instruction and builds a case-frame-like representation. Based on the essential components of this frame-representation, in this example the action "bring", with destination "Master", and the object "glass of milk", the agent finds respective actions in the conceptual action hierarchy and develops suitable plans, if necessary. Using the action and the object hierarchies, the agent determines a higher level action "bring", and fills respective features of this action with information from the case frame. In this example, the object of the bring-action is instantiated as a glass containing milk, e.g. a yet unspecified instance "glass#1" of type "glass", which is a sub-class of the concept "container" with feature "contents = milk". This information can be inferred from the object hierarchy. The bring-action involves two sub-actions: one is to get the respective object (i.e. a glass of milk) and the other one is to deliver it (i.e. to the Master). In order to fulfill the preconditions of the generic bring-action, the agent must have the object: "have(agent, object)" or, with unified variables: "have(agent, glass#1)", and the agent must be at the destination location, i.e. the Master's sofa, to deliver it. In order to fulfill the first condition, the agent performs a get-action, and thus starts a planning process to get a glass of milk, which involves going to the dishwasher, picking a glass from the dishwasher, going to the fridge and filling the glass with milk. The precondition of the deliver-action states that the agent has to be at the Master's location. Thus, the agent plans now a route to the location of the Master (which is the sofa) and moves there. The actions referenced at this level, e.g. "get a glass from the dishwasher", are translated into executable actions for the household agent. The execution of the move itself is based on a planned route involving obstacle avoidance and is similarly assumed to be executable by the robotic agent.

5.3 Speech Controlled Toy Car

A speech interface for a remote controlled toy car has been developed using the DragonTM speech recognition software. The speech interface for the remote controlled toy car allows the user to issue spoken commands like 'forward', 'right', or 'north', 'east' etc. instead of using the manual remote control. The verbal speech commands are hooked to menu items in a window which is again connected to a parallel output port. Signals arriving at this port are transmitted through a custom-made electronic switching circuit to the remote control. This system is supposed to be extended with a more complex natural language interface and

command interpreter, which allows the processing of complex command sentences like "Make a slight turn right and then stop."

6. Conclusions

In this paper we presented a framework for action descriptions and its connection to natural language interfaces for artificial agents. The core point of this approach is the use of a generic action/object hierarchy, which allows interpretation of natural language command sentences, issued by a human user, as well as planning and reasoning processes on the conceptual level, and connects to the level of agent executable actions. The linguistic analysis is guided by a case frame representation, which provides a connection to actions (and objects) represented on the conceptual level. Planning processes can be implemented using typical precondition and effect descriptions of actions in the conceptual hierarchy. The level of primitive actions (leaf nodes of this conceptual hierarchy) connects to the agents' executable actions. The level of primitive actions can thus be adapted to different types of physical agents with varying action sets.

Further work includes the construction of a suitable, general action ontology, based on standard ontologies like FrameNet (ICSI, 2007), Ontolingua (KSL, 2007), Mikrokosmos (CRL, 1996), Cyc (Cycorp, 2007), or SUMO (Pease, 2007; IEEE SUO WG, 2003), which will be enriched with precondition and effect formulas. Other topics to be pursued relate to communication of mobile physical agents (humans) in a "speech-controlled" environment. The scenario is related to the "smart house" but instead of being adaptive and intelligent, the house (or environment) is supposed to respond to verbal instructions and questions by the human user. A special issue, we want to address, is the development of a sophisticated context model, and the use of contextual information to resolve ambiguities in the verbal input and to detect impossible or unreasonable actions.

7. Acknowledgements

8. References

Allen, J. F.; Miller, B. W.; Ringger, E. K. & Sikorski, T. (1996). A Robust System for Natural Spoken Dialogue, *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96),* pp. 62-70, Santa Cruz, California, 1996

Allen, J. F. ; et al. (1995). The TRAINS project: A Case Study in Defining a Conversational Planning Agent, *J. of Experimental and Theoretical AI*, 7, 1995, 7-48.

Artale, A. & Franconi, E. (1998). A Temporal Description Logic for Reasoning about Actions and Plans, *Journal of Artificial Intelligence Research*, 9, pp. 463-506

Artale, A. & Franconi, E. (1994). A Computational Account for a Description Logic of Time and Action, *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, pp. 3-14, Bonn, Germany, 1994

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D. & Patel-Schneider, P. (Eds.) (2003). *The Description Logic Handbook*, Cambridge University Press

Baader, F., Milicic, M.; Lutz, C.; Sattler, U. & Wolter, F. (2005). Integrating Description Logics and Action Formalisms: First Results. *2005 International Workshop on Description Logics (DL2005)*, Edinburgh, Scotland, UK, July, 2005, CEUR-Workshop Proceedings Volume 147

Baker, C. F.; Fillmore, C. J. & Lowe, J. B. (1998). The Berkeley FrameNet Project, *Proceedings COLING-ACL*, Montreal, Canada, 1998

Brachman, R. J. & Schmolze, J. G. (1985). An Overview of the KL-ONE Knowledge Representation System, *Cognitive Science*, 9(2), pp. 171-216

Coyne, B. & Sproat, R. (2001). WordsEye: An Automatic Text-to-Scene Conversion System, *Siggraph*, 2001. See also: Semantic Light, www.semanticlight. com

CRL (1996). *Mikrokosmos*, http://crl.nmsu.edu/Research/Projects/mikro/

CSL Princeton (2007). *WordNet*, http://wordnet.princeton.edu/

Cycorp (2007). *Cyc*, http://www.cyc.com/

Devanbu, P. T. & Litman, D. J. (1996). Taxonomic Plan Reasoning, *Artificial Intelligence*, 84, 1996, pp. 1-35

Di Eugenio, B. (1998). An Action Representation Formalism to Interpret Natural Language Instructions, *Computational Intelligence*, 14, 1998, pp. 89-133

Epsitec SA (2007). CeeBot, www.ceebot.com/ceebot/index-e.php, Epsitec SA, Belmont, Switzerland, 2007

IEEE SUO WG (2003). *SUMO Ontology,* IEEE P1600.1 Standard Upper Ontology Working Group (SUO WG), http://ontology.teknowledge.com/

Fillmore, C. J. 1968. The case for case, In: *Universals in Linguistic Theory*, E. Bach & R. Harms, (Eds.), pp. 1-90, Holt, Rhinehart and Winston, New York

FIPA (2002). FIPA ACL Message Structure Specification. Foundation for Intelligent Physical Agents (FIPA), 6 December 2002, www.fipa.org/specs/fipa00061

ICSI (2007). *FrameNet*, http://framenet.icsi.berkeley.edu/

Gomez, F. (1998). Linking WordNet Verb Classes to Semantic Interpretation. *Proceedings COLING-ACL, Workshop on the Usage of WordNet on NLP Systems*. Université de Montréal, Montréal, Canada, 1998

Jurafsky, D.; Wooters, C.; Tajchman, G.; Segal, J.; Stolcke, A.; Fosler, E. & Morgan, N. (1994). The Berkeley Restaurant Project, *Proceedings ICSLP*, pp. 2139-2142, 1994

Kemke, C. (2006). Towards an Intelligent Interior Design System, *Workshop on Intelligent Virtual Design Environments (IVDEs)* at the *Design Computation and Cognition Conference*, Eindhoven, the Netherlands, 9th of July 2006

Kemke, C. (2004). Speech and Language Interfaces for Agent Systems, *Proc. IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pp.565-566, Beijing, China, September 2004

Kemke, C. (2003). A Formal Approach to Describing Action Concepts in Taxonomical Knowledge Bases, In: *Foundations of Intelligent Systems*, Lecture Notes in Artificial Intelligence, Vol. 2871, N. Zhong, Z.W. Ras, S. Tsumoto, E. Suzuku (Eds.), pp. 657-662, Springer, 2003

Kemke, C. (2001). *About the Ontology of Actions*, Technical Report MCCS -01-328, Computing Research Laboratory, New Mexico State University

Kemke, C. (2000). What Do You Know about Mail? Knowledge Representation in the SINIX Consultant, *Artificial Intelligence Review*, 14, 2000, pp. 253-275

Kemke, C. (1988). Die Darstellung von Aktionen in Vererbungshierarchien (Representation of Actions in Inheritance Hierarchies). In: *GWAI-88, Proceedings of the German Workshop on Artificial Intelligence*, pp. 57-63, Springer, 1988

Kemke, C. (1987). Representation of Domain Knowledge in an Intelligent Help System, Proc. of the Second IFP Conference on Human-Computer Interaction INTER-ACT'87, pp. 215-200, Stuttgart, FRG, 1987

Kemke, C. & Walker, E. (2006). Planning through Integrating an Action Abstraction and a Plan Decomposition Hierarchy, *Proceedings IEEE/WIC/ACM International Agent Technology Conference IAT-2006,* 6 pages, December 2006, Hong Kong

Kruijff, G.-J. M (2006). Talking on the moon. Presentation at the AAAI 2006 Spring Symposium "To Boldly Go Where No Human-Robot Team Has Gone Before", Stanford, California, 2006

Kruijff, G.-J. M.; Zender, H.; Jensfelt, P. & Christensen, H. I. (2007). Situated Dialogue and Spatial Organization: What, Where… and Why? *International Journal of Advanced Robotic Systems*, 4(1), 2007, pp. 125-138

KSL (2007). *Ontolingua*, http://www.ksl.stanford.edu/software/ontolingua/

Liebig, T. & Roesner, D. (1997). Action Hierarchies in Description Logics, *1997 International Workshop on Description Logics (DL'97),* Gif sur Yvette (Paris), France, September, 1997, http://www.lri.fr/~mcr/ps/dl97.html

Lifschitz, V. (1987). On the Semantics of STRIPS, In: *The 1986 Workshop on Reasoning about Actions and Plans*, pp.1-10, Morgan Kaufmann

Patel-Schneider, P. F.; Owsnicki-Klewe, B.; Kobsa, A.; Guarino, N.; MacGregor, R.; Mark, W. S.; McGuiness, D. L.; Nebel, B.; Schmiedel, A. & Yen, J. (1990). Term Subsumption Languages in Knowledge Representation, *AI Magazine*, 11(2), 1990, pp. 16-23

Pease, A. (2007). *Suggested Upper Merged Ontology (SUMO),* http://www.ontologyportal.org

Pednault, E. (1989). ADL: Exploring the middle ground between STRIPS and the situation calculus. *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pp. 324-332, 1989

Rickheit, G. & Wachsmuth, I. (Eds.) (2006). *Situated Communication*, Mouton de Gruyter

SFB-360 (2005). *Situated Artificial Communicators*, www.sfb360.uni-bielefeld.de/sfbengl.html

Stent, A.; Dowding, J.; Gawron, M.; Owen Bratt, E. & Moore, R. (1999). The CommandTalk Spoken Dialogue System, *Proceedings of the 37th Annual Meeting of the ACL*, pp. 183-190, University of Maryland, College Park, MD, 1999

Torrance, M. C. (1994). *Natural Communication with Robots*, S.M. Thesis submitted to MIT Department of Electrical Engineering and Computer Science, January 28, 1994

Traum, D.; Schubert, L. K.; Poesio, M.; Martin, N.; Light, M.; Hwang, C.H.; Heeman, P.; Ferguson, G. & Allen, J. F. (1996). Knowledge Representation in the TRAINS-93 Conversation System. *Internat'l Journal of Expert Systems, Special Issue on Knowledge Representation and Inference for Natural Language Processing,* 9(1), 1996, pp. 173-223

Wahlster, W. (1997). VERBMOBIL: *Erkennung, Analyse, Transfer, Generierung und Synthese von Spontansprache*. Report, DFKI GmbH, 1997

Walker, E. (2004). *An Integrated Planning Algorithm for Abstraction and Decomposition Hierarchies of Actions*, Honours Project, Dept. of Computer Science, University of Manitoba, May 2004

Weida, R. & Litman, D. (1994). Subsumption and Recognition of Heterogeneous Constraint Networks, *Proceedings of CAIA-94*, pp. 381-388

Human Robot Interaction

Edited by Nilanjan Sarkar

Human-robot interaction research is diverse and covers a wide range of topics. All aspects of human factors and robotics are within the purview of HRI research so far as they provide insight into how to improve our understanding in developing effective tools, protocols, and systems to enhance HRI. For example, a significant research effort is being devoted to designing human-robot interface that makes it easier for the people to interact with robots. HRI is an extremely active research field where new and important work is being published at a fast pace. It is neither possible nor is it our intention to cover every important work in this important research field in one volume. However, we believe that HRI as a research field has matured enough to merit a compilation of the outstanding work in the field in the form of a book. This book, which presents outstanding work from the leading HRI researchers covering a wide spectrum of topics, is an effort to capture and present some of the important contributions in HRI in one volume. We hope that this book will benefit both experts and novice and provide a thorough understanding of the exciting field of HRI.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Christel Kemke (2007). "From Saying to Doing" - Natural Language Interaction with Artificial Agents and Robots, Human Robot Interaction, Nilanjan Sarkar (Ed.), ISBN: 978-3-902613-13-4, InTech, Available from: http://www.intechopen.com/books/human_robot_interaction/_from_saying_to_doing__-_natural_language_interaction_with_artificial_agents_and_robots

INTECH
open science | open minds