# Natural Language Programming of Industrial Robots

Maj Stenmark, Pierre Nugues
Department of Computer Science, Lund University, Lund, Sweden
maj.stenmark@cs.lth.se, pierre.nugues@cs.lth.se

***Abstract*—In this paper, we introduce a method to use written natural language instructions to program assembly tasks for industrial robots. In our application, we used a state-of-the-art semantic and syntactic parser together with semantically rich world and skill descriptions to create high-level symbolic task sequences. From these sequences, we generated executable code for both virtual and physical robot systems. Our focus lays on the applicability of these methods in an industrial setting with real-time constraints.**

***Index Terms*—High-level programming, industrial robots, natural language.**

## I. Introduction

Robot programming is time consuming, complex, error-prone, and requires expertise both of the task and the platform. Within industrial robotics, there are numerous vendor-specific programming languages and tools, which require certain proficiency. However, to increase the level of automation in industry, as well as to extend the use of robots in other domains, such as service robotics and disaster management, it has to be possible for non-experts to instruct the robots.

Since humans communicate with natural language (NL), it is appealing to use speech or text as instruction means for robots as well. This is complicated for two main reasons: First, NL can be ambiguous and its expressivity is richer than that of a typical programming language. Secondly, tasks can be expressed as goals as well as imperative statements, hence, even if the instructions are correctly parsed, the description itself is often not enough to create a successful execution. There has to be a substantial amount of knowledge in the system to translate the high-level language instructions to executable robot programs.

In this paper, we introduce a method for using natural language to program robotized assembly tasks and we describe a prototype of it. The core idea of the method is to use a generic semantic parser to produce a set of predicate-argument structures from the input sentences. Such predicate-argument structures reflect common semantic situations described through language and at the same time use a logical representation. Using the predicate-argument structures, we can extract the orders embedded in a user's sentences and map them more easily onto robot instructions.

## II. Related Work

Natural language programming for robots has been investigated for both service and navigational robots from the early 1970's. SHRLDU [1] is an oft-cited example of the first attempts to give robots conversational competences. To interpret and convert a user's sentences into instructions, robotic system often make use of an intermediate representation. Examples include [2][3][4], where the authors have developed their own domain specific sematic representation for navigational robots.

Tenorth et al. [5] parse pancake recipes in English from the World Wide Web and generate programs for their household robots. They use the WordNet lexical graph [6] with a constituent parser and they map WordNet's *synsets* to concepts in the Cyc [7] ontology. Finally, they add mappings to common household objects.

In order to bridge the sentence to the robot actions, all the examples mentioned above seem to use ad-hoc intermediate formalisms that are difficult to adapt to other domains, languages, or environments. Frame semantics [8] is an attempt to provide generic models of logical representations of sentences. Frame semantics starts from prototypical situations shared by a language community, English for instance, and abstracts them into frames. While frame semantics is only a theory, FrameNet [9][10] is a comprehensive dictionary that provides a list of lexical models of the conceptual structures. Commercial situations like selling are represented with the Commerce_sell predicate-argument structure, where the arguments include a buyer, a seller, and goods. Given a sentence and a verb belonging to this frame, like *vend*, *sell*, or *retail*, a semantic parser will identify the predicate and its arguments.

As of today, FrameNet has not a complete coverage of English verbs and nouns. Propbank [11] and Nombank [12] are subsequent projects related to FrameNet that both developed comprehensive databases of predicate-argument structures for respectively verbs and nouns and annotated large volumes of text with it. As training data is essential to the development of statistical semantic parsers, most of the current parsers use the Propbank nomenclature, as they are easier to train.

To the best of our knowledge, few robotics systems use existing predicate-argument nomenclatures. An exception is RoboFrameNet [13], a language-enabled robotic system that adopts frame semantics. However, the authors wrote their own frames inspired from FrameNet. Their model includes a decomposition of the frames into a sequence of primitives. They built a semantic parser that consists of a dependency parser and rules to map the grammatical functions to the arguments. Such techniques

have been used from the early Absity system [14] and are known to have a limited coverage.

In the project, we describe below, we used a multilingual high-performance statistical semantic parser [15][16] trained on the Penn Treebank and using the Propbank and Nombank lexicons. In contrast to RoboFrameNet, the parser we adopted can accept any kind of sentence.

## III. System Overview

### A. Architecture

The central part of the system architecture [17] is the *knowledge integration framework* (KIF). KIF consists of a client-server architecture where the server hosts ontologies, provides services, and object and skill libraries. The ontologies represent the world objects, such as robots, sensors, work-pieces and their properties, as well as robot skills. The skills are semantically annotated, platform-independent state machines, which are parameterized for reuse and executed using JGrafchart [18].

KIF interacts with the *engineering system* (ES), which is the high-level programming interface, and the robot controller. The ES is implemented as an extension to the programming and simulation environment ABB RobotStudio [19]. When creating the robot cell, the objects, such as sensors, work-pieces, and trays, can be generated or downloaded from KIF together with the ontology. Every physical object has an *object frame*, and a number of *feature frames* related to its object frame. These frames are used to express geometrical constraints; see Fig. 1.

A program consists of a sequence of steps, which in turn consists of actions, motions, skills, or nested steps. The sequence is created using the graphical interface of the ES. The steps for picking a printed circuit board (PCB) and placing it on a fixture are shown in Fig. 2. To execute the sequence, platform specific code (robot code or the XML file used by the state machine executor) is generated for the motions, actions and skills, and deployed on the target platform.

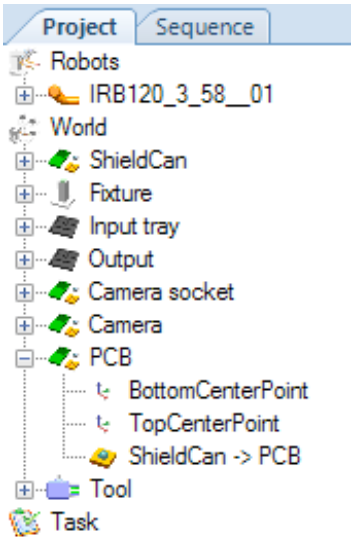


Fig. 1. In the object browser, the robots are listed under *robots*; all physical objects are listed under *world* and each object lists its own frames and relations.

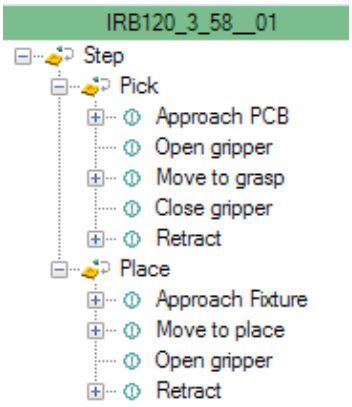


Fig. 2. The visual rendering of a program for picking and placing a PCB.

To help the user quickly setup a skeleton sequence of a task, we provide a natural-language parsing service on

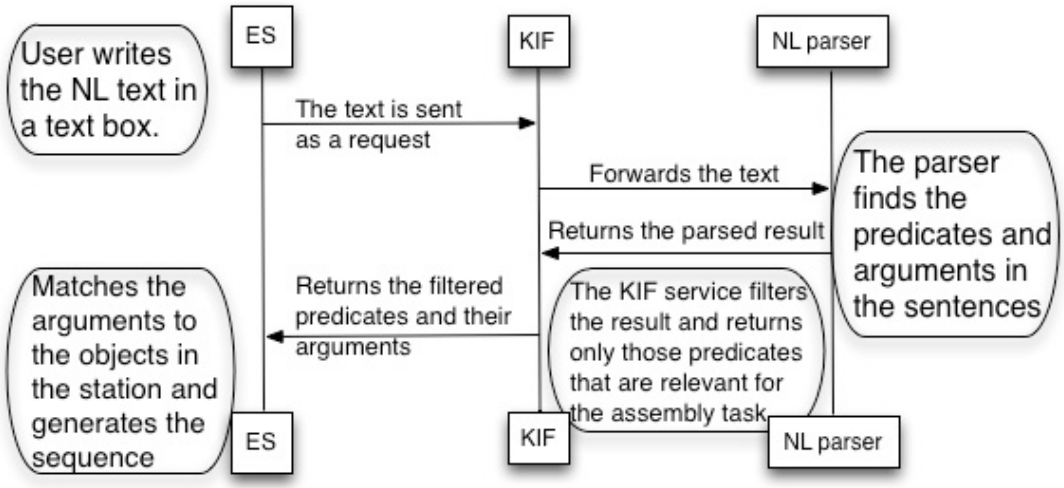


Fig. 3. The data flow between the user, the KIF service and the semantic parser.

KIF; see Fig. 3. The service reads the text input, parses the text in search of predicate-arguments structures, and returns those containing predicates that match the task vocabulary.

On the client side, the predicates are mapped to programs; the arguments representing station objects and the other parameters are filled with default values or geometrical relations taken from the station. The programmer can then check the sequence, possibly alter it, and finally execute it.

### B. Predicate-Argument Structures

An assembly task can be defined as e.g.: *Pick the PCB from the input tray and place it on the fixture. Then take a shield can and insert it on the PCB*. These sentences are parsed to extract the predicates-argument structures *pick(PCB, input tray)* and *place(it, fixture)*, while the agent parameter, *robot,* is implicit.

The parser is trained on the Penn Treebank that uses the Propbank lexicon [20]. Propbank labels each English verb with a sense and defines a set of arguments that is specific to each verb. In the sentence: *Pick the PCB from the input tray and place it on the fixture*, both *pick* and *place* have sense 1 (pick.01 and place.01):

- Pick.01 has three possible arguments; *arg0*: agent, entity acquiring something, *arg1*: thing acquired and *arg2*: seller.
- Place.01 has *arg0*: putter, *arg1*: thing put, and *arg2*: where put.

The parsing output is shown in Fig. 4. As shown in this figure, the *arg1* and *arg2* arguments to pick.01 are matched to *the PCB* and *the input tray* respectively, while the robot (*arg0*) is implicit.

| | Pick | the | PCB | from | the | input | tray | and | place | it | on | the | fixture |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| pick.01 | | A1 | | A2 | | | | | | | | | |
| place.01 | | | | | | | | | | A1 | A2 | | |

Parsing sentence required 20ms.

Fig. 4. Parsing result from the first sentence. The parser identified two predicates, *pick* and *place*, and two arguments for each predicate.

Before mapping the identified arguments to the station objects, the arguments corresponding to the same entity have to be gathered into *coreference chains*; see Fig. 5. The last step links the coreference chains to the entities in the station using the object name or type.

### *Task Vocabulary*

The vocabulary is currently rather limited. We only considered predicates matching programs that the robot could generate. Each program has arbitrary language tags such as *take*, *insert*, *put*, *calibrate*, either predefined or edited by the user. Possible arguments to the programs are the objects in the station, which is a well-defined, finite world.

## IV. High-Level Programming Prototype

On the highest level, the task is represented by an assembly graph [21], which is a partially ordered tree of

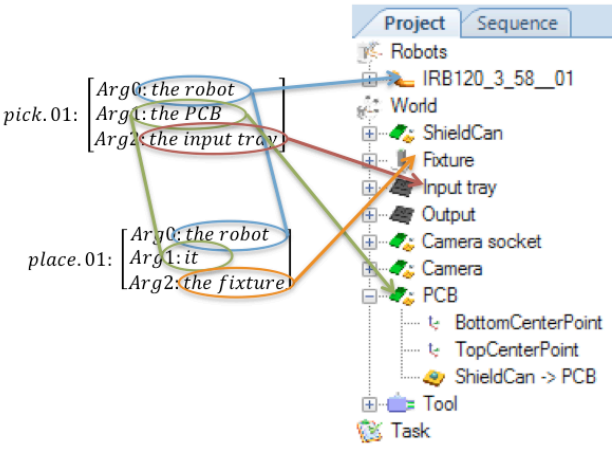


Fig. 5. Coreference solving of entities in the first sentence. Mentions corresponding to the same entity are gathered into coreference chains.

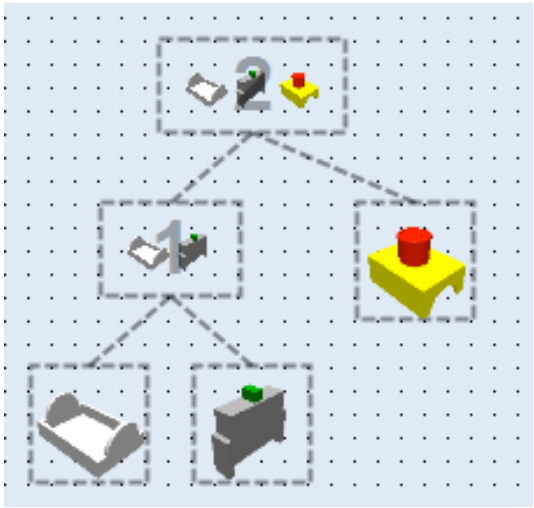

Fig. 6. The assembly graph is created by dragging and dropping icons of the objects. Here, the first assembly operation involves the base of the emergency button (left) and the switch (right). In the second operation the lid is added to the subassembly.

assembly operations; see Fig. 6. The graph describes the assembly of an emergency stop button box.

Each operation specifies the desired geometrical relations of the involved objects and the skill type for the assembly. Examples of skill types in the ontology are *screw*, *glue* and *peg-in-hole,* where each type can have several different implementations. The assembly operations are subgoals, and the root node represents the final goal of the task. The motivation for the assembly graph is to have a platform independent task description, so that different implementations can be compared and reasoned about.

The assembly graph is realized by sequences of actions and motions for each robot. The sequence can be: 1) created manually by adding actions and motions one by one and editing their properties, 2) generated from the assembly graph or 3) created by using a natural language interface. An example of the latter is shown in Fig. 7: two assembly steps of a stop button box assembly are described by natural language.

Fig. 8 shows the parsed result from Fig. 7. Each predicate is mapped to a type of skill. For example, a *pick* or *take* consist of a sequence of primitive actions: approaching the object to be picked, opening the gripper, moving slowly to a grasp position, closing the gripper, and then retracting. The mapping of the objects are rudimentary: by name (ignoring space and case) or, if this is unsuccessful, by the ontology type (e.g. fixture, tray or pin). When generating the motions for picking and placing the objects, the application uses the existing grasp positions and relations between the work-pieces as default values. If no relations exist, a new one is created with zero offset. The actions for opening and closing the gripper are taken from the selected tool, since each tool describes its own procedures. The resulting sequence is shown in Fig. 9.

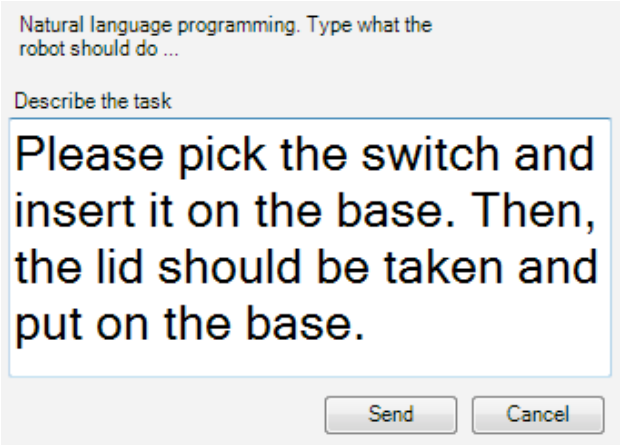


Fig. 7. The commands are written into a simple text field, the narrative is then sent to the KIF service that facilitates semantic parsing.

Using reasoning services available from KIF, the generated sequence can then be checked for inconsistencies and additional skills are suggested to solve missing constraints (e.g. an object has to be placed in a fixture before an assembly or a tool needs to be exchanged between drilling and picking).

The code generated from the sequence is executable on both virtual and physical robots; see Fig. 10. To expand the vocabulary, the user can add natural language tags to existing steps and upload them to KIF.

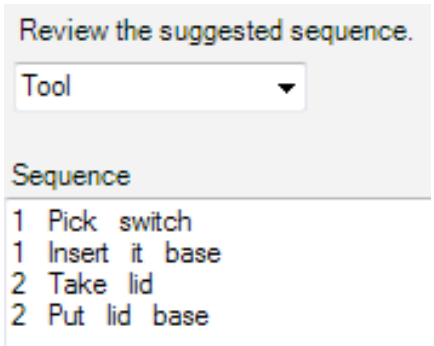


Fig. 8. The result the parsed predicates along with their arguments.

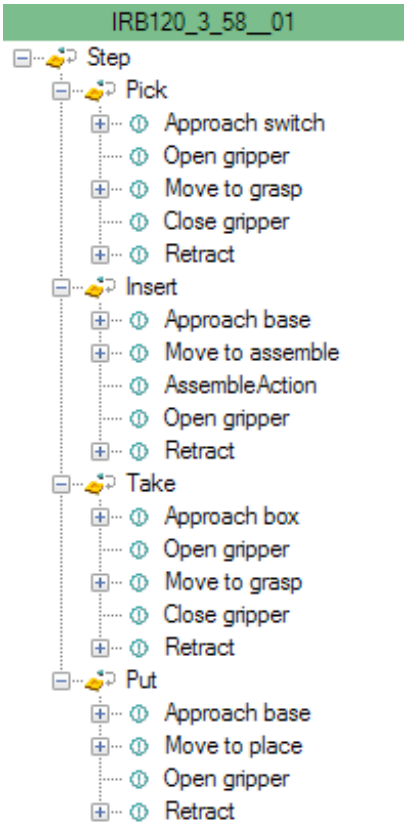


Fig. 9. The generated sequence for inserting a switch on the base of a stop bottom and putting the top of the box on the base.

Fig. 10. The sequence from Fig. 8 executed on a physical robot.

## V. Conclusions

In this paper, we have presented a system to describe robot assembly tasks in the RobotStudio environment using natural language. From an input sentence, the processing pipeline applies a sequence of operations that parses the sentence and produces a set of predicate-argument structures. The semantic module uses statistical techniques to extract automatically these structures from the grammatical functions.

The NLP pipeline is designed so that it reaches high accuracies and has short response times required for user interaction. Parsing a sentence takes from 10 to 100 milliseconds. Drawing from the frame semantics theory, the semantic parser uses a standardized inventory of structures and can be applied to unrestricted text. This makes the pipeline more easily adaptable to new tasks and new environments.

As second step, the system maps the predicate and the arguments extracted from the sentence to robot actions and objects of the simulated world. These objects and actions are stored in a unified architecture, the *knowledge integration framework* that represents and manages the entities, services, and skill libraries accessible to the robot.

Making the application part of a tool already used by industry is a conscious choice: high-level natural language programming is convenient to get an application up and running quickly. However, when tuning the parameters of a task, the programmer can still use the traditional tools, e.g. to edit the generated code directly. Also, because of the industrial focus, we have real-time performance on the underlying sensor and control systems, which is necessary for many manipulation tasks in assembly operations.

Unlike previously reported results, our approach supports both a command-like interface and parsing of longer texts, yielding multistep programs.

## VI. Future Work

The obvious drawback of this implementation is the lack of speech as an input modality. However, since many smartphones have sufficient speech recognition for our purposes, this was not our main scientific concern. Rather, we wanted to extend the skill library with relevant and generic assembly skills. We plan to extend our application with tools that make it simple to extract the natural language predicate-argument structures given a skill, its parameters (objects, velocities, forces), and a textual description of the skill. Another extension is to automatically search after suitable implementations that are tagged with synonyms to the used words.

## Acknowledgment

The research leading to these results has received funding from the European Union's seventh framework program (FP7/2007-2013) under grant agreements N° 230902 (ROSETTA) and N° 285380 (PRACE) and from the Swedish Research Council grant N° 2010-4800 (SEMANTICA).

## References

[1] T. Winograd. *Procedures as a representation for data in a computer program for understanding natural language*. Technical report, MIT, 1971.

[2] S. Tellex, T. Kollar, S. Dickerson et al. "Understanding Natural Language Commands for Robotics Navigation and Mobile Manipulation". *In Proceedings of AAAI 2011*.

[3] M. MacMahon, B. Stankiewicz and B. Kuipers. "Walk the Talk: Connecting Language, Knowledge, Action in Route Instructions". *In Proceedings of AAAI 2006*.

[4] N. Shimizu and A. Haas. "Learning to Follow Navigational Route Instructions". *In IJCAI 2009*.

[5] M Tenorth, D. Nyga and M. Beetz. "Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web." *In ICRA 2010*.

[6] WordNet http://wordnet.princeton.edu/.

[7] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira, "An introduction to the syntax and content of Cyc," *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pp. 44–49, 2006.

[8] C. Fillmore, Frame semantics and the nature of language. Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech, 280:20–32, 1976.

[9] FrameNet https://framenet.icsi.berkeley.edu/fndrupal/.

[10] J. Ruppenhofer, M. Ellsworth, M. R. L. Petruck, C. R. Johnson, and J. Scheffczyk. *FrameNet II: Extended Theory and Practice.* Technical report, 2010.

[11] M. Palmer, D. Gildea and P. Kingsbury. "The Proposition Bank: an annotated corpus of semantic roles." *In Computational Linguistics*, 31(1): 71-105.

[12] Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., and Grishman, R. (2004). The NomBank project: An interim report. In Meyers, A., editor, HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation, pages 24– 31, Boston.

[13] B. J. Thomas and O. C. Jenkins. "RoboFrameNet: Verb-centric Semantics for Actions in Robot Middleware." *In ICRA 2012*.

[14] G. Hirst, *Semantic interpretation and the resolution of ambiguity*. Cambridge University Press. 1987.

[15] A. Björkelund, L. Hafdell, and P. Nugues. "Multilingual semantic role labeling." *In Proceedings of CoNLL-2009*, pp 43-48, Boulder.

[16] A. Björkelund, B. Bohnet, L. Hafdell, and P. Nugues. "A high-performance syntactic and semantic dependency parser." *In COLING 2010: Demonstration Volume*, pp 33-36, Beijing.

[17] A. Björkelund, L. Edström, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. G. Robertz, D. Storkle, A. Blomdell, R. Johansson, and et al. "On the integration of skilled robot motions for productivity in manufacturing," *In Proc. IEEE ISAM 2011*, pp 1–9, 2011.

[18] Grafchart. http://www.control.lth.se/Research/tools/grafchart.html, 2012.

[19] ABB RobotStudio. http://www.abb.com/product/seitp327/78fb236cae7e605dc1256f1e002a892c.aspx, 2013.

[20] R. Johansson and P. Nugues. "Dependency-based syntactic-semantic analysis with ProbBank and NomBank." *In Proceedings of CoNLL-2008*, pp 183-187.

[21] J. Malec, K. Nilsson, and H. Bruyninckx. "Describing assembly tasks in a declarative way." *In ICRA 2013 WS Semantics, Identification and Control of Robot-Human-Environment Interaction*, 2013.