# Conversational Programming for Collaborative Robots

Maike Paetzel-Prüsmann
*Department of Linguistics*
*University of Potsdam*
Potsdam, Germany
paetzel-pruesmann@uni-potsdam.de

Julie Hunter
*LINAGORA Labs*
Toulouse, France
jhunter@linagora.com

Kranti Chalamalasetti
*Department of Linguistics*
*University of Potsdam*
Potsdam, Germany
kranti.chalamalasetti@uni-potsdam.de

Kate Thompson
*LINAGORA Labs*
Toulouse, France
cthompson@linagora.com

Alexandros Nicolaou
*Department of Linguistics*
*University of Potsdam*
Potsdam, Germany
alexandros.nicolaou@uni-potsdam.de

Ozan Güngör
*Synergeticon*
Hamburg, Germany
o.guengoer@synergeticon.de

David Schlangen
*Department of Linguistics*
*University of Potsdam*
Potsdam, Germany
david.schlangen@uni-potsdam.de

Nicholas Asher
*Institut de Recherche en Informatique de Toulouse*
*Centre National de Recherche Scientifique*
Toulouse, France
nicholas.asher@irit.fr

***Abstract*—In this position paper, we describe a novel approach of programming industrial robots via conversational dialogue. We believe that conversational programming, unlike other interfaces for humans to reprogram industrial robots, will enable novices to teach a robot complex new procedures without any knowledge of programming required. Using a sample conversation between a human User and an industrial robotic arm, we discuss how our approach differs from other (spoken) human-robot interfaces and why it has the potential to solve difficulties of such interfaces when it comes to learning to abstract from specific examples. We also describe the unique challenges conversational programming involves and how, once these are solved, it could be integrated into industrial settings of the future.**

***Index Terms*—Cognitive Human-Robot Interaction, Industrial Robots, Robust/Adaptive Control of Robotic Systems**

## I. Introduction

In the Industry 4.0 vision, collaborative robots (cobots) will assist workers in factories of the future, operating on assembly lines or assisting with maintenance, fetching tools for workers or helping them by sorting or preparing tools and assembly parts [1], [2].

Today's cobots are not yet flexible enough to implement the Industry 4.0 vision and rely on extensive manual programming for complex tasks, which is very time consuming and generally requires robotics experts [3]. These experts, however, have often limited knowledge of the domain in which the robot is situated, which prolongs the programming process and leaves it highly prone to errors. Especially in small lot production, robotic systems need to be reprogrammed on a regular basis as the type of work they are required to do frequently changes. We believe the difficult and time-consuming process of teaching the robot new skills is one of the major bottlenecks that limits the usage of robots in small lot production environments.

Recently, more industrial robotics systems have incorporated specific human-robot interaction interfaces, that in theory allow novices to demonstrate a task to the robot by guiding its joints (e.g., [4]–[6]; see [7] for a recent survey). The robot then learns from the human guidance and after seeing several examples it is able to abstract the knowledge to changing environments. However, as soon as the constraints become more complex, these systems, again, reach their limits [7].

To address these issues, we aim to develop a spoken dialogue interface for industrial human-robot collaboration that can be used by a typical human user without sophisticated programming skills or access to special training data [8]. This interface will allow a robot to take in instructions and descriptions expressed through conversation and convert them into code. In particular, given that our motivation is to teach robots new skills, our goal is to provide them with a means to exploit conversation to construct general, repeatable programs.

This objective introduces challenges on two levels. First, a program must be specified in such a way as to prompt

This work is supported by the German Bundesministerium für Bildung und Forschung (BMBF) and the French Ministère de l'Enseignement supérieur, de la Recherche et de l'Innovationn (MESRI).

the robot to execute the program in any situation that provides the requisite equipment and assembly parts within the robot's reach. That is, the robot must be able to exploit multimodal (visual and linguistic) representations at the type level to recognize particular and appropriate instances of those types at execution time. Second, effective programming through conversation will require much more than simply being able to translate natural language instructions directly into an executable program. Crucially, it will require the robot to be able to produce and exploit conversational moves as tools that allow it to collaborate with a human on the task of constructing the program itself [9]. As an illustration, consider the following dialogue.

HUMAN: Begin by picking up one flat head screw. Next, place the screw in a washer.
ROBOT: Any kind of washer?
HUMAN: No, actually you should only use square washers for this task.

While we might expect translations of the two first lines of instruction expressed in the sample dialogue to figure in the final code for the program that the human and robot are building, the following question-answer exchange is not meant to add new steps to the program under construction. It rather brings to light, and then settles, an ambiguity in the program up to that point – an objective that it achieves very efficiently. It is such a general capacity to engage in interactive conversation, as humans do, that we believe will allow a robot to quickly get the point of an instruction without being shown hundreds or thousands of examples.

In this position paper, we introduce our efforts to bring together research on discourse structure, conversational grounding [9], multimodal grounding [10], [11] and robotics [12] to develop a conversational programming toolkit that would allow a robot to learn complex programs from simpler ones through multimodal conversation. While we use both statistical and symbolic data representations of natural language found in many robotics applications, our approach extends the current body of work by using discourse information to construct an intermediate representation of human instructions which then informs robot code creation. We elaborate on this aspect further in Section III after we describe our robotics use case in Section II. The paper concludes in Section IV by discussing the potential of our work for industrial settings of the future.

## II. The Domain: Pre-Assembly of a Pegboard

We designed a sample domain that is representative of real scenarios from the industry while allowing for adapting to different levels of difficulty depending on the number and diversity of assembly parts as well as the complexity of the combined procedures to be learned.

The aim for the robot, in our case a Universal Robotics[1] robot arm with six degrees of freedom, is to assemble a pegboard using individual parts that are randomly distributed on a source board. The parts consist of different types of 3D-printable screws, washers, nuts and plates that differ both in their type and color to add more variety and complexity for the program to be learned.

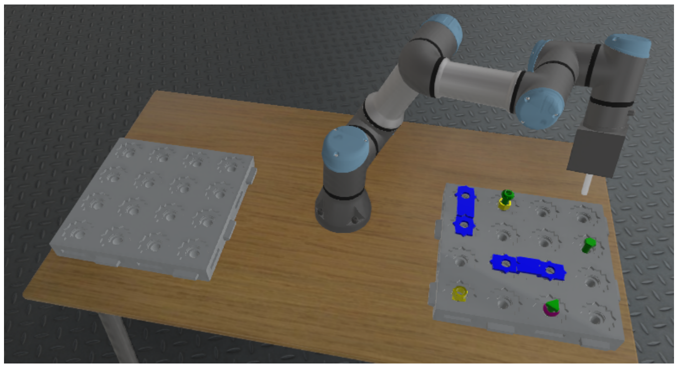
Figure 1. The sample industrial use-case shown in the Webots simulated environment. On the right side of the table is the source board with the randomly placed pieces. To the left is the currently empty target board the robot needs to fill.

The generic program of the robot allows it to find and pick up the individual parts from the source board, and place them on the target board at a desired pose. What it needs to learn is how exactly the target board needs to be assembled, which requires it to learn how to combine parts into complex structures and repeat this procedure until certain conditions are met. A setup of our use-case in the Webots[2] simulator is shown in Figure 1. We aim for our use case to not only work in simulation, but to translate to the real world, requiring as little adaptation as possible. Consequently, the generic program the robot generates needs to be agnostic to the type of environment it was learned in. A secondary program responsible for the grounding to the real environment then ensures that the learned processes can be executed in various environments.

## III. Our Vision for Conversational Programming

Let's say that a human wants the robot to fill the target pegboard with *bridges*, which are made of nuts, screws and a plate. A dialogue to get the robot to assemble the board could look as follows:

USER: Can you build a *bridge*?
ROBOT: I don't know how to build a *bridge*.
USER: Okay, to build a bridge, we need **two washers** in two holes that are next to each other.
ROBOT: Can I use any washer?
USER: Yes. Once you place the washers, you need to add **a plate** on top of the two washers so it connects the two. Can you show me?
ROBOT: [*executes the first steps*]

[1]https://www.universal-robots.com/

[2]https://cyberbotics.com/

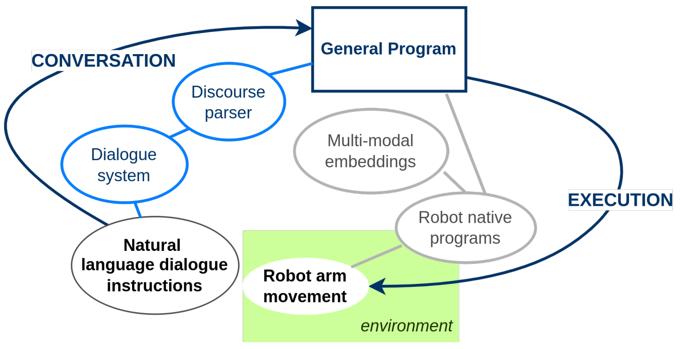

Figure 2. A schematic overview of the proposed system. The conversational dialogue leads to the construction of the general program, which can then be executed by a secondary program in varying environments.

USER: Actually, you can only use **red plates** for this task. Please fix what you've done.

ROBOT: OK.

ROBOT: [*Exchanges blue with red plate*]

USER: Excellent. Now we need to add a cross slot screw in each of the holes of the plate. Can you finish the assembly?

ROBOT: Does the color of the screw matter?

USER: Yes! Please only use blue screws.

ROBOT: [*Adds two blue screws*]

USER: Great, this is how you build a bridge. Now can you please do that until there are no red plates available anymore?

Moving from the conversation above to an ability to execute the task of building a bridge raises challenges related both to moving from conversation to a general program and to executing that program, as mentioned in the introduction and visualized in Figure 2. A program is an abstract object that can be executed or instantiated on multiple occasions and in multiple environments, and program execution requires a particular set of skills. When executing the program the robot must ensure that the requisite objects are actually available in the scene it is operating in; the execution fails if this is not the case. To do this, the robot will need to exploit multimodal (vision and linguistic) representations at the type level to recognize tokens of the corresponding concepts that are appropriate for the task. We believe this will require building a flexible coupling between visual and linguistic representations both at the *type* level and *token* level.

The execution of the program need not, of course, always require a grounding of a definite description to a particular object. In the dialogue above, the robot may potentially pick up any object of the type that is required and position it on the pegboard according to the instructions. To resolve the *pick_and_place* calls, a separate program with access to the visual scene queries the position of the next potential object with the required properties and sends this as well as the position of the next free space on the board to the robot arm. Separating these two parts of the program allows the programming to work independently of the visual scene as much as possible. More generally, the distinction between programs and executions, and between type grounding and token grounding, reveals an important distinction between instructing a robot with *natural language commands* on a particular occasion of use to some sequence of actions (as frequently used in the related work on human-robot collaborative task-solving and natural language programming), and *programming by conversation*, in which the robot learns a program that it can then execute in multiple environments.

Language is a helpful tool for drawing attention to important properties and specifying conditions: when the User says "Now we need to add a cross-slot screw", they make it clear in one shot that the type of screw matters. If they were to only use images or, say, demonstrations without language, it would likely require considerably more examples before the robot (justifiably) felt confident that the type of screw matters. Nevertheless, language use is often vague and underspecified and human teachers can forget to include important details. In this case conversation becomes key—a tool that allows two (or more) agents to collaboratively spell out and even correct a program under construction.

Let us see how this plays out in our sample dialogue, which illustrates multiple examples of underspecified instructions involving the type of washers needed as well as the color of the plates and screws. In some cases, the robot explicitly asks for clarification (e.g., "Can I use any washer?") and the User answers by either specifying the missing property or making it explicit that no further specification is needed. The robot's questions in these exchanges are motivated by knowledge that multiple types of washers may be available and by reasoning about these possibilities. Another strategy adopted by the robot is to simply try something and see if it works, as it does with the plate. In this case, it is upon seeing the result that the human realizes they have left out an important detail. This exchange requires an interaction between visual and linguistic information as well as a strategy how to balance potential errors in the task execution and unnecessarily extending the teaching dialogue.

A second type of ambiguity illustrated by the dialogue concerns interpreting an underspecified command like *fix what you have done*. By itself this command doesn't make clear what the robot needs to do. To resolve that ambiguity, the robot needs to understand that it has made a particular error (using a plate of the wrong color), which requires it to understand the corrective move made by the User and its relation to what the robot has done so far.

These examples make clear that using conversation to collaboratively construct a program requires being able to infer how conversational moves are linked to each other, or even to nonlinguistic actions such as placing a plate on top of two washers. When the robot asks “Can I use any washer?” the human must understand that it is asking specifically about whether it can use any washer when it carries out the command of placing a washer in each of two consecutive holes. When the User replies affirmatively, the robot needs to relate the answer to the preceding question. But what’s more, the robot needs to know what it should *do* as a response to a particular type of conversational move. It needs to know that an answer, for example, means that it should disregard certain alternatives that it previously thought possible when it asked its question; an acknowledgment like “Excellent” means that the robot has done well and can move on in its program construction; a corrective move tells it to delete certain content from its program and replace it with other content, and so on.

To build a program from the conversation, the robot requires an understanding of the *structure* of the conversation as well as its content. Conversational moves are anaphorically related to other conversational moves in semantically significant ways and these different relations affect how a conversational move will influence the content of the final program including whether or not it shows up at all [13]. A correction move, for instance, will both remove and add content to some part of the final program, but crucially, the program will not have a representation of the corrective move itself.

Turning back to the sample dialogue, we see that many of the elaborations and sequential moves *should* be reflected in the final program. In fact, if we focus only on these moves, we see that they form a coherent substructure on their own and in particular, we start to get something that looks much more like the final desired abstract program as detailed in Algorithm 1.

**Algorithm 1** Assemble Bridge

```
0: $w_1, w_2 \leftarrow object(type : washer)$
0: $s_1, s_2 \leftarrow object(type : screw, color : blue)$
0: $p \leftarrow object(type : plate, size : 2, color : red)$
0: $required\_objects \leftarrow [w_1, w_2, s_1, s_2, p]$
0: **if** $scene$ has $required\_objects$ **then**
0:     $(x, y) \leftarrow find\_space\_on\_board(2)$
0:     $board.pick\_and\_place((x, y), w_1)$
0:     $board.pick\_and\_place((x, y+1), w_2)$
0:     $board.pick\_and\_place((x, y), (x, y+1), p)$
0:     $board.pick\_and\_place((x, y), s_1)$
0:     $board.pick\_and\_place((x, y+1), s_2)$
0: **else**
0:     $program\_state \leftarrow fail$
```

This algorithm consequently needs to be interpreted by a different program that *grounds* the abstract program given the available types of assembly parts as well as the specific robot and its distinct abilities to execute certain moves.The algorithm responsible for grounding the program also needs to be able to handle underspecified abstract commands like *pick_and_place* of $w_1$ and $w_2$, which can be any type of washer on the board.

It is important to note that the attempt to ground a program in a given setting could lead to errors. For example, it could be possible that the robot finds itself in an environment certain parts are unavailable. It is important that the robot is then able to communicate the problem and differentiate between responses that require a change in its abstract program and the current environment. The robot could, for example, say: “I don’t find any blue screws on the board, but I was told to use blue screws. What do you want me to do?” The human could then either correct a misunderstanding in the abstract program (i.e., the screws can be of any color), or make the missing type of screws available to the robot.

## IV. Discussion and Future Work

In this position paper, we described our vision to design a novel way of programming industrial robots using spoken conversations. We presented the use-case of pre-assembling a pegboard and how a human User could teach a robot a new routine in the given scenario. The aim is to scale the scenario to become increasingly complex and to show that we can adapt the grounding of the program to different environments. With our German industrial partner Synergeticon[3] we plan to develop a proof-of-concept in a realistic industrial setting and consequently highlight the potential of our project for the Industry 4.0 vision. We will evaluate our conversational programming technique against standard kinesthetic and visual teaching interfaces common in the industry. We expect our system to not only decrease the time it takes to program the robot to learn a new task and lead to less errors in the execution of new programs, we believe that this will also decrease the cognitive load of the human User as well as increase their perception of the robot and comfort in the task.

We foresee that our project will also bring to light future challenges for roboticists and speech processing researchers to tackle. For example, learning how to pick up objects of unknown size, shape and material is still challenging for robots. In our work, we hence rely on assembly parts that are known to the robot. However, we are confident that our work can be extended to unknown objects once grasping algorithms become more reliable. Similarly, automatic Speech Recognition (ASR) in an industrial setting may produce very unreliable results given the noise level common in such environment. While more robust ASR systems are an active area of research, we believe that using simulated environments could present an intermediate solution to the problem: Instead of performing the entire teaching dialogue in the factory, the robot could be preprogrammed in simulation and the program then merely executed in the physical environment. With our work, we aim to enable such sim2real transfer of conversational programming.

[3]https://synergeticon.de/en/

## References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[2] T. L. Olsen and B. Tomlin, "Industry 4.0: Opportunities and challenges for operations management," *Manufacturing & Service Operations Management*, vol. 22, no. 1, pp. 113–122, 2020.

[3] E. Matheson, R. Minto, E. G. Zampieri, M. Faccio, and G. Rosati, "Human–robot collaboration in manufacturing applications: a review," *Robotics*, vol. 8, no. 4, p. 100, 2019.

[4] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll, "Intuitive instruction of industrial robots: Semantic process descriptions for small lot production," in *2016 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, 2016, pp. 2293–2300.

[5] C. Schou, J. S. Damgaard, S. Bøgh, and O. Madsen, "Human-robot interface for instructing industrial tasks using kinesthetic teaching," in *IEEE ISR 2013*. IEEE, 2013, pp. 1–6.

[6] M. Tykal, A. Montebelli, and V. Kyrki, "Incrementally assisted kinesthetic teaching for programming by demonstration," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2016, pp. 205–212.

[7] J. Berg and S. Lu, "Review of interfaces for industrial human-robot interaction," *Current Robotics Reports*, vol. 1, no. 2, pp. 27–34, 2020.

[8] N. Mavridis, "A review of verbal and non-verbal human–robot interactive communication," *Robotics and Autonomous Systems*, vol. 63, pp. 22–35, 2015.

[9] M. Appelgren and A. Lascarides, "Interactive task learning via embodied corrective feedback," *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 2, pp. 1–45, 2020.

[10] C. Silberer and M. Lapata, "Learning grounded meaning representations with autoencoders," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 721–732.

[11] H. Tan and M. Bansal, "Vokenization: Improving language understanding with contextualized, visual-grounded supervision," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 2066–2080.

[12] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek, "Robots that use language," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 25–55, 2020.

[13] N. Asher and A. Lascarides, *Logics of Conversation*. Cambridge University Press, 2003.