Natural Language For Human Robot Interaction

Huda Khayrallah
UC Berkeley Computer Science Division
University of California, Berkeley
Berkeley, CA 94704
+1 (510) 642-6000
huda@icsi.berkeley.edu

Sean Trott
International Computer Science Institute
1947 Center Street #600
Berkeley, CA 94704
+1 (510) 666-2900
seantrott@icsi.berkeley.edu

Jerome Feldman
International Computer Science Institute
1947 Center Street #600
Berkeley, CA 94704
+1 (510) 666-2900
feldman@icsi.berkeley.edu

ABSTRACT

Natural Language Understanding (NLU) was one of the main original goals of artificial intelligence and cognitive science. This has proven to be extremely challenging and was nearly abandoned for decades. We describe an implemented system that supports full NLU for tasks of moderate complexity. The natural language interface is based on Embodied Construction Grammar and simulation semantics. The system described here supports human dialog with an agent controlling a simulated robot, but is flexible with respect to both input language and output task.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Natural language.*

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems – *Natural language interfaces.*

I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *discourse, language parsing and understanding*.

General Terms

Experimentation, Human Factors, Languages.

Keywords

Natural language understanding (NLU), robotics simulation, referent resolution, clarification dialog.

1. NATURAL LANGUAGE INTERFACES

Natural language interfaces have long been a topic of HRI research. Winograd's 1971 SHRDLU was a landmark program that allowed a user to command a simulated arm and to ask about the state of the block world (Winograd, 1971). There is currently intense interest in both the promise and potential dangers of much more capable robots.

Table 1. NLU beyond the 1980's

1)	Much more computation
2)	NLP technology
3)	Construction Grammar: form-meaning pairs
4)	Cognitive Linguistics: Conceptual primitives, ECG
5)	Constrained Best Fit: Analysis, Simulation, Learning
6)	Under-specification: Meaning involves context, goals. ..
7)	Simulation Semantics; Meaning as action/simulation
8)	CPRM= Coordinated Probabilistic Relational Models; Petri Nets ++
9)	Domain Semantics; Need rich semantics of Action
10)	General NLU front end: Modest effort to link to a new Action side

As shown in Table 1, we believe that there have been sufficient scientific and technical advances to now make NLU of moderate scale an achievable goal. The first two points are obvious and general. All of the others except for point 8 are discussed in this paper. The CPRM mechanisms were not needed in the current system, but are essential for more complex actions and simulation (Barrett 2010). In this paper, simulation is action in a robot model; more general simulations using our framework are discussed in Feldman(2014) and Narayanan(1999).

2. EMBODIED CONSTRUCTION GRAMMAR

This work is based on the Embodied Construction Grammar (ECG), and builds on decades of work on the Neural Theory of Language (NTL) project. The meaning side of an ECG construction is a schema based on embodied cognitive linguistics. (Feldman, Dodge, and Bryant 2009).

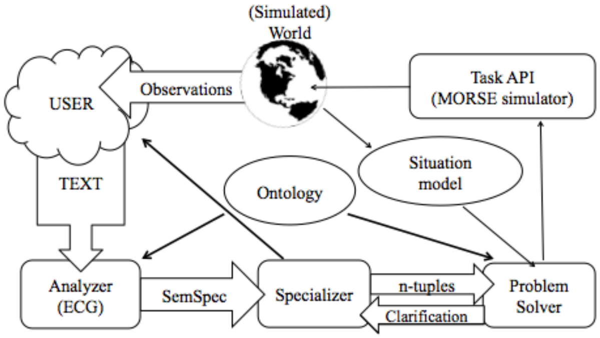
ECG is designed to support the following functions:
1) A formalism for capturing the shared grammar and beliefs of a language community.
2) A precise notation for technical linguistic work
3) An implemented specification for grammar testing
4) A front end for applications involving deep semantics
5) A high level description for neural and behavioral experiments.
6) A basis for theories and models of language learning.

In this work, we focus on point 4; we are using ECG for the natural language interface to a robot simulator. We suggest that NLU can now be the foundation for HRI with the current generation of robots of limited complexity. Any foreseeable robot will have limited capabilities and will not be able to make use of language that lies outside its competence. While full human-human level NLU is not feasible, we show that current NLU technology supports HRI that is adequate for practical purposes.

3. SYSTEM ARCHITECTURE

As shown in the system diagram, Figure 1, the system is designed to be modular; a crucial part of the design is that the ECG grammar is designed to work for a wide range of applications that have rich internal semantics. ECG has previously been demonstrated as a computational module for applied language tasks' for understanding solitaire card game instructions (Oliva el al. 2012).

Figure 1: The system diagram.



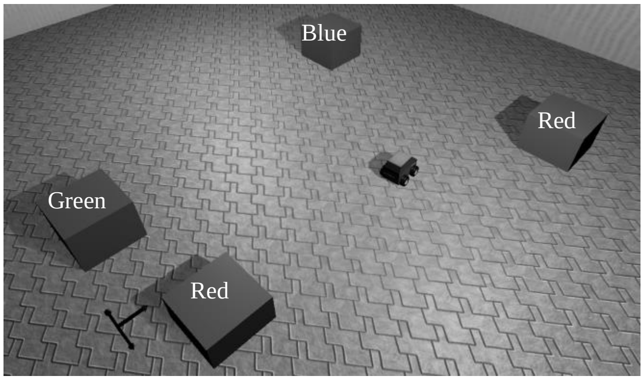
The main modules are the analyzer, the specializer, the problem solver, and the robot simulator. The analyzer semantically parses the user input with an ECG grammar plus ontology and outputs a data structure called the SemSpec.

The specializer crawls the SemSpec to capture the task relevant information, which it sends to the problem solver as a data structure called an n-tuple. The problem solver then uses the information from the n-tuple, along with the problem solver's internal model of the world, to make decisions about the world and carry out actions. Additionally, the problem solver updates its model of the world after each action, so it can continue to make informed decisions and actions.

While this paper focuses on English, the system also works in Spanish. The same analyzer, N-tuples, problem solver, and simulator can be used without alteration. Spanish and English have major grammatical differences; therefore, they use different constructions, so a modified specializer is needed. The specializer extracts the relevant information and creates the same n-tuple. This allows the problem solver and robot simulator to remain unchanged.

In addition to the application to robotics, a similar architecture is also used for metaphor analysis. For this domain, more constructions must be added to the grammar, but the same analyzer can be used. Instead of carrying out commands in a simulated world, metaphors and other information from the SemSpec are stored in a data structure, which can be queried for frequency statistics, metaphor entailments, and inferences about a speaker's political beliefs.

Figure 2: The Simulated World.



3.1 Supported Input

Table 2 highlights a representative sample of working input, corresponding to the scene in Figure 2. There is an obvious focus on motion, due to the functionality of the robot used. The location to which the robot is instructed to move can include specific locations "location 1 2," and specific items "Box1." The system can also handle more complicated descriptions, using color and size. Additionally, when the user references an indefinite object, such as, "a red box," and there are multiple objects that fit the description, one of the objects that satisfies the condition is chosen randomly. For definite descriptions, such as "*the* red box", the system requests clarification, asking: "which red box?"

Table 2: Sample supported input (English)

1)	Robot1, move to location 1 2!
2)	Robot1, move to the north side of the blue box!
3)	Robot1, push the blue box East!
4)	Robot1, move to the green box then push the blue box South!
5)	Robot1, if the small box is red, push it North!
6)	where is the green box?
7)	is the small red box near the blue box?
8)	Robot1, move behind the big red box!
9)	which boxes are near the green box?

Table 3: Sample supported input (Spanish)

1)	Robot1, muévete a posición 1 2!
2)	Robot1, muévete al parte norte de la caja azul!
3)	Robot1, empuje la caja azul al este!
4)	Robot1, muévete a la caja verde y empuje la caja azul al sur!
5)	Robot1, si la caja pequeña es roja, la empuje al norte!
6)	dónde está la caja verde?
7)	está la caja roja y pequeña cerca de la caja azul?
8)	Robot1, muévete detrás de la caja roja y grande!
9)	cuáles cajas están cerca de la caja verde?

In addition to commands involving moving and pushing, the system can also handle yes or no questions—as demonstrated in Example 7 in Table 2. Example 5 demonstrates a conditional imperative; the robot will only perform the instruction if the condition is satisfied. The system can also handle basic referent resolution, as demonstrated in Example 5. This is done by choosing the most recent antecedent that is both syntactically and semantically compatible. This method is described in (Oliva et al, 2012) and is based on the way humans select antecedents.

The total range of supported input is considerably greater than the sentences included in the tables; this sample is intended to give a sense of the general type or structure of supported input in both English and Spanish.

If the analyzer cannot analyze the input, the user is notified and prompted to try typing the input again. If the user attempts to describe an object that does not exist in the simulation, such as "the small green box", the system informs the user, "Sorry, I don't know what 'small green box' is."

If there is more than one object that matches an object's description (e.g. "red box"), and a definite article is used (e.g. "the red box"), the system asks for clarification, such as: "which red box?" The user can then offer a more specific description, such as: "the small one."

4. Extended Example: Robot Simulation

In order to demonstrate the integration and functionality of the system, we will trace an extended example from text to action. We will consider the command, "Robot1, if the box near the green box is red, push it South!" First, notice that this sentence involves a number of complex linguistic constructions including conditionals, definite descriptions, pronoun resolution, etc. Any simple analysis would incorrectly match "it" with the green box. This example is discussed in the context of the example situation in Figure 2, the system diagram of Figure 1, and the supplementary video: https://docs.google.com/file/d/0B6rDSJGnf4t6SlhyeG9NcEpZdTQ/edit?pli=1

4.1 Analyzer

The input text is first parsed by the analyzer program using the ECG grammar. The analyzer uses syntactic and semantic properties to develop a best-fit model and produce a SemSpec. This SemSpec is a grammatical analysis of the sentence, consisting of conceptual schemas and their bindings (Bryant 2008). A constructional outline of the SemSpec for this example can be found in Appendix A.

4.2 Specializer

The specializer extracts the relevant information for the problem solver from the SemSpec. This output is in the form of an n-tuple, a data structure implemented using Python dictionaries. The n-tuple for this example can be found in Appendix B. Our Python-based n-tuple templates are a form of Agent Communication Language; although the content of the n-tuples changes across different tasks and domains (such as robotics and metaphor analysis), the structure and form can remain the same. When new applications are added, new n-tuple templates are defined to facilitate communication with the problem solver. The n-tuples are not limited to a Python implementation.

In this case, the command is in the form of a conditional, so the specializer must fill in the corresponding template by extracting the bindings from the SemSpec.

Additionally, the direct object of the "push" command is a pronoun; the analyzer cannot match pronouns with their antecedents, so the specializer uses a combination of syntactic and semantic information to perform reference resolution (Oliva et al, 2012). In this case, the antecedent of "it" is "the box near the green box", so the specializer passes this information to the problem solver in the n-tuple.

4.3 Problem Solver

The problem solver parses the n-tuple to determine the actions needed, and then performs those actions in context. Language almost always under-specifies meaning, so context plays a large role in the Problem Solver. The core app, such as our MORSE robot simulator, cannot be expected to maintain and exploit context. The solver begins by determining the type of command, which is here a conditional. Before it performs the command, it must evaluate the truth of the condition.

In this example, the problem solver must determine which box is "near the green box" and then determine whether that box has the property *red*. Using the information provided by the specializer, the solver searches through its data structure, which contains the current and updated state of the simulated world. Once the solver identifies the box that is located near the green box, it can evaluate whether that box is red using its vision system or world knowledge.

If the condition is satisfied, the robot performs the specified action: in this case, "push it [the box near the green box] South!" This action is considerably more complex than simply moving to a box, and involves a nontrivial amount of trajectory planning. First, the solver disambiguates the location of the box by searching through its data structures. Then, it determines that to push the box South, it must move to the North side of the box (avoiding obstacles along the way), rotate to face the box, and move South. This results in pushing the box South. The planning functionality is encapsulated and could be replaced by one of the elaborate trajectory planners in the literature.

Finally, the call to execute a move action is made through the wrapper class of the robot or simulator API, here MORSE. This additional level of abstraction allows the system to work with an arbitrary robot or simulator, assuming it supports the same primitives.

4.4 Simulator

The demo system is built on top of MORSE (Echeverria et al, 2011), which in turn relies on Blender for 3d visualization. MORSE is an open-source simulator designed for academic robotics. While our system is designed to work on an arbitrary simulator, it has been influenced by the specifications of this one. MORSE provides some useful functionality, including realistic physics and a variety of interfaces (including the Python bindings we use). It also has some key limitations, such as restricted functionality and the lack of path planning. The use of Blender allows for realistic physics simulations, and easy modeling.

4.5 Alternate Platforms

While we demo our system with the MORSE simulator, we are also considering physical robot platforms. A leading option is the QRIO robot, which was developed by Sony. We are working on incorporating QRIO in conjunction with a research partner and are exploring other possibilities.

4.6 Additional System Features

The system also incorporates several other key features that aid both its semantic understanding and its functionality. First, as mentioned above, the specializer performs basic referent resolution between pronouns and their antecedents. This is done by maintaining a LIFO stack of the syntactic heads of past Nominal Phrases; when a pronoun is encountered, the specializer matches its syntactic and semantic context with the most recent compatible object reference on the stack. This procedure is somewhat novel because it incorporates semantic features, as well as syntactic ones, in determining the compatibility of a pronoun and its antecedent. For example, if the robot is instructed to "push" something, the specializer checks that the antecedent is "movable", using the ontology lattice.

Second, the specializer resolves cases of the "anaphoric one" using a related but distinct method. The usage of the anaphoric one is problematic because it often refers simply to an antecedent's *category*. For example, in the sentence "John has a red cup and I have a green *one*", *one* refers to the category of "cup". Research suggests that discourse and semantic context are necessary for proper resolution (Salmon-Alt et al, 2001). Our system has semantic and world knowledge, and it uses these contextual features in the resolution process; qualifiers in the antecedent, such as "red", are compared with the anaphor's qualifiers ("green"), and are added iteratively until the system is able to locate a referent in the simulated world.

Finally, the system handles under-specified input via appropriate clarification dialogs. The problem solver has information about the simulated world, so it can determine when to query the user for more information. If the user instructs the robot to move to "the red box", and there are two red boxes, the system asks: "which red box?" The user might reply: "the small one". This clarification process allows the system to interact with the user, and continues until the input is properly specified. It then uses one-anaphora resolution to determine the correct referent.

.

5. RELATION TO PRIOR WORK

In addition to the early work on SHRDLU, there has also been some recent work on using natural language to control robots. In contrast to Winograd's work— as well as our own— these approaches focus on learning from examples (Howard et al. 2014).

Both our work and Winograd's focus on a specific domain (Winograd, 1971). SHRDLU knew the properties of blocks, and understood how to interact with them specifically. While our Analyzer is general, our Problem Solver (Figure 1) is specific to each application.

SHRDLU analyzed the sentence in terms of the definitions of the individual words. It was not designed to be adapted to different tasks. In contrast, our modular system allows for portions to be used for different tasks (such as metaphor analysis). For SHRDLU, language understanding was highly coupled with the simulated world, and the world was re-simulated based on the language. In order to model a more realistic interaction with robots, our problem solver issues commands to a robot simulator API, which could be replaced with a robot API.

Recent work has also approached the problem of providing a natural language interface for robots. Matuszek et al. learned a parser based on pairs of English commands and control language expressions In contrast, our work builds upon the Embodied Construction Grammar(ECG), which we believe allows us to better understand the intentions of the human, because embodied semantics directly models intentional aspects of language. ECG constructions can also capture indirect speech acts like: "I want the red boxed moved North".

Other work has focused on robots asking humans for help when stuck (Tellex et al.). We implement a basic request for clarification, since scope of our work is to perform commands issued by the user. However, in an environment where the robot has more autonomy, the need to ask with assistance on a task— and not just ask for clarification about a command—can be crucial. All of this requires a much richer NLU system, like ECG.

6. CONCLUSION

This paper demonstrates a fully integrated yet modular system that provides a natural language interface, based on embodied semantics, to a robotic simulator. In combination with (Oliva et al, 2012) this demonstrates that the ECG and Analyzer architecture of Figure 2 can be used for diverse applications: solitaire, robotic control, and metaphor analysis. The Spanish version further illustrates the flexibility of the system. The use of the ECG allows for a deep semantic understanding, which supports full treatment of different input languages, as well as providing solid framework for analyzing embodied concepts, such as motion and spatial relations. The main goal of this work is not just in the domain of robotics, but rather a general NLU front end for autonomous systems.

6.1 Limitations

The primary limitation of the current system is scale. We have implemented and tested English grammars much richer than shown here, but well short of complete coverage (Feldman, Dodge, and Bryant 2009). This is a focus of current research. In order to facilitate more natural interactions, we have also begun the integration of a spoken language recognizer. But our restricted domains should support much more robust recognition.

6.2 Ongoing Work

This project is still in active development. In the domain of robotics, we have begun to study more complex (possibly humanoid) robots operating in complex real-world situations. We are also exploring totally different NLU tasks including the interpretation of metaphorical language.

On the system level, scaling remains the core issue. The constructional structure of a language (e.g., English) is complex but bounded. We believe that enough is now understood to support realization of the fixed compositional subset of a language.

We have also implemented a morphological pre-processing system that reduces the number of necessary constructions and exploits the existing schema lattices in the grammar. Additionally, we have reduced the need for lexical constructions by developing a new method of expanding the lexicon, which involves inserting "tokens" of various syntactic and semantic categories into a token list (e.g., "red" is a token of the "color" type). The tokens do not need to be read in when the grammar is compiled. This allows us to significantly increase the size of the lexicon, while maintaining the complex semantics of the grammar.

For full coverage, the lexicon and idiomatic usage of each domain will need to be captured, almost certainly through incremental machine learning. Syntactic and semantic usage frequencies can be exploited as well. We could also benefit from the lessons learned in dialog systems (McTear 2002).

For coupling the general NLU front end to varying application domains, some additional system work should be done. As with any system coupling, the ontology referenced by Analyzer needs to be shared with that of the Problem Solver in order to give both modules the relevant information about the terms used. We are hopeful that RDF/OWL will be helpful here, but have not tried this yet.

Obviously enough, we should test our ideas in more realistic (or real) environments, including those with multiple agents. We are exploring several possible applications and welcome ideas and collaboration on this.

7. ACKNOWLEDGMENTS

We thank Luca Gilardi of the International Computer Science Institute for his help in designing the system framework and also the reviewers for several insightful comments. This research was supported by the Office of Naval Research (Grant Code: N000141110416).

8. REFERENCES

[1] Barrett, L. R. 2010. An Architecture for Structured, Concurrent, Real-Time Action. Ph.D. diss., Department of Computer Science, University of California at Berkeley

[2] Bryant, J. E. 2008. Best-Fit Constructional Analysis. Ph.D. diss., Department of Computer Science, University of California at Berkeley

[3] Chang N. 2008. Constructing grammar: A computational model of the emergence of early constructions. Ph.D. diss., Computer Science Division, University of California at Berkeley

[4] Echeverria, G.; Lassabe, N.; Degroote, A. and Lemaignan, S. 2011. Modular open robots simulation engine: Morse. In the proceedings of the 2011 IEEE International Conference Robotics and Automation, 46-51 IEEE.

[5] Feldman J.; Dodge E.; and Bryant J. A Neural Theory of Language and Embodied Construction Grammar. In The Oxford Handbook of Linguistic Analysis, Heine B. and Narrog H. 111-138, Oxford University Press, 2009

[6] Affordances, Actionability, and Simulation. J. Feldman , S. Narayanan. First Workshop on Affordances: Affordances in Vision for Cognitive Robotics, held in conjunction with Robotics Science and Systems 2014 (RSS 2014), Berkeley, California

[7] Howard, T. M., Tellex, S., and Roy, N. 2014. A Natural Language Planner Interface for Mobile Manipulators. In Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA).

[8] Matuszek, M; Herbst, E, Zettlemoyer, L; and Fox, D. 2012. Learning to Parse Natural Language Commands to a Robot Control System. In Proceedings of the International Symposium on Experimental Robotics (ISER)

[9] McTear, M. 2002 Spoken dialogue technology: enabling the conversational user interface. ACM Computing Surveys. [DOI: 10.1145/505282.505285]

[10] Mok E., 2008. Ph.D. diss., Department of Computer Science, University of California, Berkeley, CA

[11] Narayanan, S. (1999), Reasoning About Actions in Narrative, IJCAI '99, pp. 350-358.

[12] Oliva J.; Feldman J.; Giraldi L. and Dodge E. Ontology Driven Contextual Reference Resolution in Embodied Construction Grammar. 2012. In the proceedings of the 7th Annual Constraint Solving and Language Processing Workshop. Orléans, France

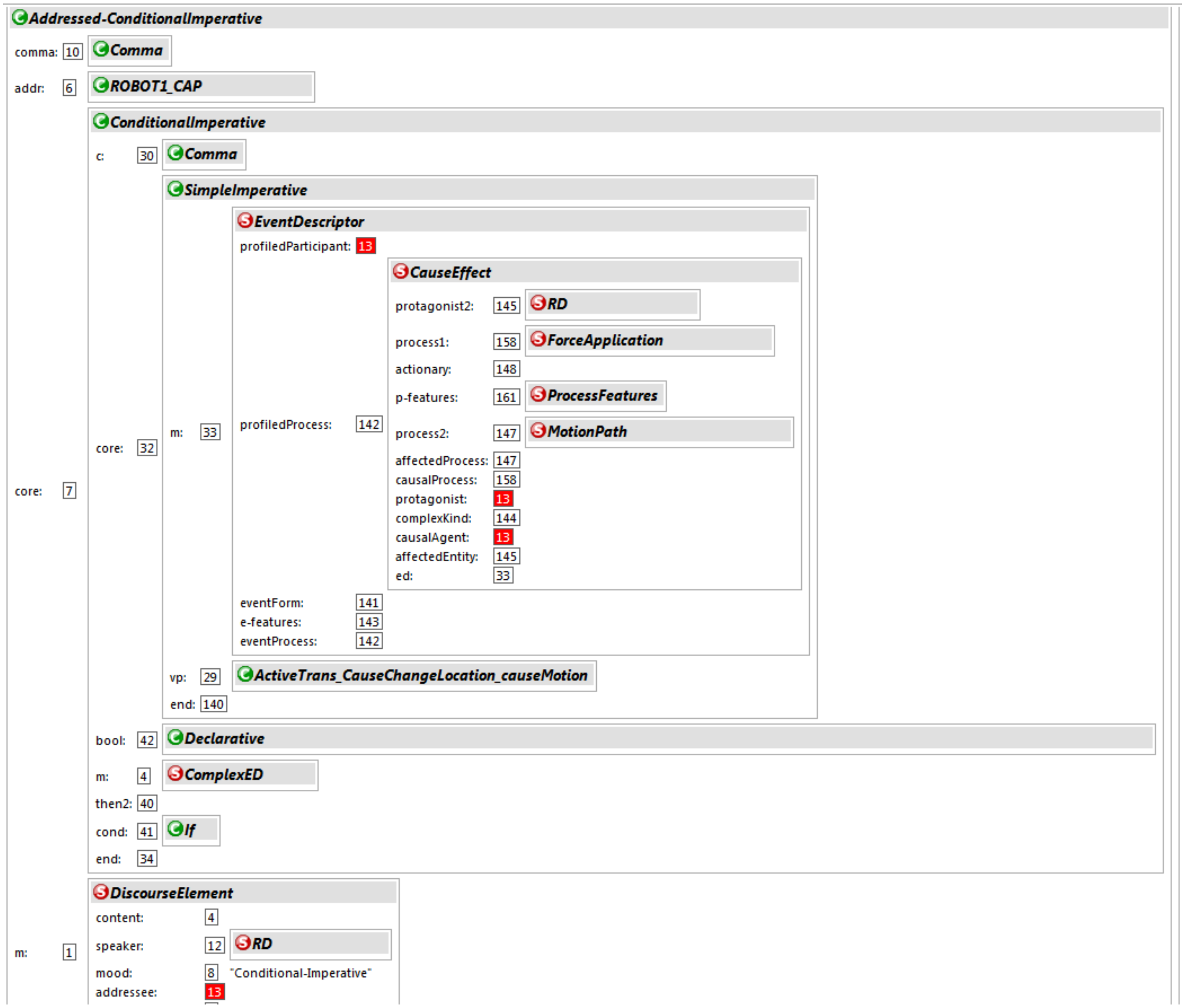
[13] Salmon-Alt, S; Romary, L. Reference Resolution Within the Framework of Cognitive Grammar, 2001. International Colloquium on Cognitive Science, San Sebastian: Spain.

[14] Tellex, S; Knepper, K; Li, A; Rus, D; and Roy, D. 2014 Asking for Help Using Inverse Semantics. Proceedings of Robotics: Science and Systems, Berkeley, CA

[15] Winograd, T. 1971 Procedures as a Representation for Data in a Computer Program for Understanding Natural Language, Technical Report 235, MIT AI

Appendix A: SemSpec example

Below is the Analyzer’s SemSpec output for the sentence: “Robot1, if the box near the green box is red, push it South!” In order to conserve space and also illustrate the entire constructional tree, many of the constructional roles and schemas have been collapsed. Small boxes with the same number denote items that are co-indexed (bound).



```
Addressed-ConditionalImperative
comma: 10  Comma
addr:  6   ROBOT1_CAP
core:  7   ConditionalImperative
           c:     30  Comma
           core:  32  SimpleImperative
                      m:  33  EventDescriptor
                              profiledParticipant: 13
                              profiledProcess:  142  CauseEffect
                                                     protagonist2:    145  RD
                                                     process1:        158  ForceApplication
                                                     actionary:       148
                                                     p-features:      161  ProcessFeatures
                                                     process2:        147  MotionPath
                                                     affectedProcess: 147
                                                     causalProcess:   158
                                                     protagonist:     13
                                                     complexKind:     144
                                                     causalAgent:     13
                                                     affectedEntity:  145
                                                     ed:              33
                              eventForm:        141
                              e-features:       143
                              eventProcess:     142
                      vp: 29  ActiveTrans_CauseChangeLocation_causeMotion
                      end: 140
           bool:  42  Declarative
           m:     4   ComplexED
           then2: 40
           cond:  41  If
           end:   34
m:     1   DiscourseElement
           content:    4
           speaker:    12  RD
           mood:       8   "Conditional-Imperative"
           addressee:  13
```

Appendix B: n-tuple example

“Robot1, if the box near the green box is red, push it South!”

Below is a representation of the N-Tuple; the actual Python code is shown in the supplementary materials. This particular n-tuple contains multiple nested n-tuples (a condition and a command), because a conditional command is relatively complex.

Return_type: error_descriptor,
Predicate_type: conditional
Parameters:

- **Kind**: Conditional
- **Condition**:
  - Protagonist: *Object-Descriptor:*
    - Type: box
    - Givenness: uniquely-Identifiable
    - *Location-Descriptor:*
  - Relation: Near
    - *Object-Descriptor:*
      - Type: box
      - Givenness: Uniquely-Identifiable
      - Color: green
  - Predication: (Color: Red)
  - Kind: Query
  - Action: be
- **Command**:
  - Kind: cause
  - Causal-Process:
    - Protagonist: Robot1_instance
    - Control_State: Ongoing
    - Speed: 0.5
    - Distance: (units: square, value: 8)
    - Acted-Upon: *Object-Descriptor:*
      - Type: Box
      - Givenness: Uniquely-Identifiable
      - *Location-Descriptor*:
      - Relation: Near
        - *Object-Descriptor:*
          - Type: box
          - Givenness: Uniquely-Identifiable
          - Color: green
    - Kind: Execute
    - Action: Force-Application
  - Affected-Process:
    - Direction: None
    - Protagonist: *Object-Descriptor*
      - Type: Box
      - Givenness: Uniquely-Identifiable
      - *Location-Descriptor:*
      - Relation: Near
        - *Object-Descriptor:*
          - Type: box
          - Givenness: Uniquely-Identifiable
          - Color: green
    - Heading: South
    - Control_state: Ongoing
    - Speed: 0.5
    - Distance: (units: square, value: 8)
    - Kind: Execute
- Causer: Robot1_instance
- Action: push_move