# Translating Natural Language into Linear Temporal Logic

Menghan Chen

*under the supervision of Toryn Klassen, Prof. Sheila McIlraith, and Prof. Gerald Penn*

## 1 Introduction

The application of Artificial Intelligence to decision making is an active field of research that has been used to address a wide range of sequential decision problems. An increasing number of physical devices, from thermostats to robot vacuum cleaners, are now being controlled by AI-based sequential decision making systems. Rather than having to program these devices, our goal is to enable non-programmers to task, advise, and personalize these systems using a (perhaps controlled) natural language, obviating the need for a user to know how to program the system or to know details of how it operates. For instance, a homeowner might want to instruct their smart home system to *"Always make coffee at 7am on weekdays."*, to *"Turn on the air conditioner when it's above 28 degrees celcius unless I'm on vacation."* or that *"I prefer to run the dishwasher after 7pm."*.

Unfortunately, natural language, unlike a computer programming language, or other formal language, can be ambiguous. Our work resolves such ambiguity external to the system by translating natural language into a formal language that is unambiguous to the underlying sequential decision system. The formal language that we appeal to as our target language is Linear Temporal Logic (LTL) [1], a propositional language extended with temporal modalities that allow it to describe properties of a system over time. The vocabulary over which the LTL is defined is taken from the sequential decision making system – the actions and propositions that form the basis of this target system. The technical approach developed in this project is applicable to a diversity of sequential decision making systems, making them amenable to control and personalization by a non-programmer via natural language.

## 2 Approach

To enable users to advise, instruct and task sequential decision making systems, and more specifically to map from English to LTL, we appeal to techniques from AI knowledge representation and reasoning (KR) and from computational linguistics. Specifically, we appeal to Combinatory Categorial Grammars (CCG) [2] as a means of mapping English statements in the vocabulary of the sequential decision making systems to LTL statements in the same vocabulary.

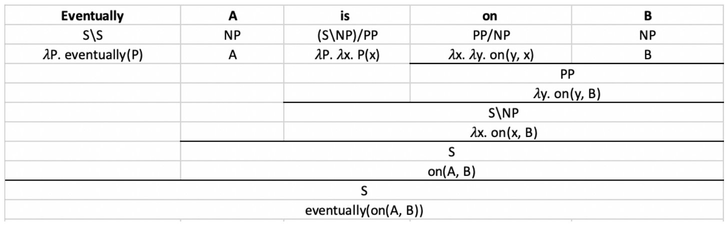
For example, *"Always make coffee at 7am on weekdays"* would be translated to the LTL formula "$\Box[\mathit{weekday} \wedge \mathit{7am} \rightarrow \mathit{make}(\mathit{coffee})]$". To narrow down the problem, we restrict the target language to the subset of LTL that captures natural instructions and personalizations, which in turn constrains the source language system to a controlled subset of English. For the purposes of this project, we elected to focus on sequential decision making systems synthesized from the Planning Domain Definition Language (PDDL) [3]. PDDL is a standardized AI planning language used in the International Planning Competition, and it describes the domain knowledge including objects, predicates, and actions as well as initial and goal state. In the long term, we hope to augment existing decision-making systems with our parser so that they can be used conveniently.

**Linear Temporal Logic**: LTL extends propositional logic with modalities that provide an intuitive but mathematically precise notation for expressing properties of a dynamical system that vary over time. LTL has been used to represent temporally extended goals and preferences in planning problems (e.g., [4, 5]). It contains formulae constructed from atomic propositions, the Boolean connectives and ($\wedge$), or ($\vee$), and negation ($\neg$), and the basic temporal modalities next ($\bigcirc$) and until ($U$). From these temporal operators, we can derive other useful temporal operators such as eventually ($\Diamond$) and always ($\Box$). The semantics of LTL is defined with respect to traces - infinite sequences of states, and in particular the truth assignments to the propositions that comprise the states.

**Planning Domain Definition Language (PDDL)**: PDDL [6] is a community standard for the representation of a planning domain model and specific planning instances. Sequential decision making systems are automatically synthesized from such models and instances. In PDDL, the state of the world is described in terms of predicates whose truth values change as the result of actions. Both predicates and actions are parameterized and held for object constants or quantified object types. The planning domain characterizes domain behavior in terms of such actions and how they affect the truth or falsity of predicates. It is these predicates, actions and object that collectively induce a vocabulary that is used by both our controlled natural language (English) and by our target (LTL) language to automatically induce a domain-specific language.

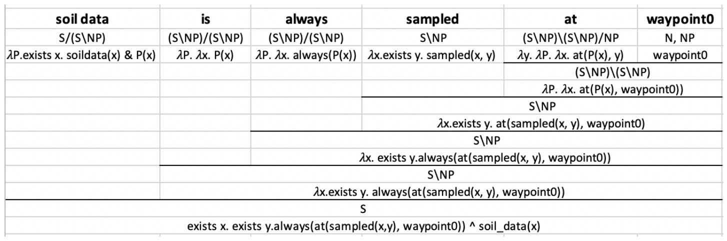
We define the target language for a particular planning problem as a subset of LTL constructed using the vocabularies from the corresponding PDDL description. The source language is a controlled natural language, which is like English but with the vocabulary restricted to terms relevant to the domain, and the grammar limited to relatively simple constructions that are easier to parse. For example, in the blocksworld domain, which is about stacking blocks, a source language instruction might be "Eventually A is on B", where A and B are both objects (blocks) and "on" is a property in the domain, denoted by the predicate "on". This instruction would be translated to an LTL formula such as "$\Diamond(on(A, B))$", which is a machine-understandable goal statement in LTL.

**Combinatory Categorial Grammars (CCG)**: CCG is a powerful (super-context-free) lexicalized grammar formalism that systematically defines how words combine to form phrases and sentences in a manner that emphasizes the semantic consequences of those combinations. The lexical categories have atomic categories including sentence (S), noun (N), and noun phrase (NP), and a set of derived functional categories constructed from the atomic ones, which build more primitive categories through combination with the arguments that they select for. Under CCG, every lexical element is assigned to one or more of these categories, each with a semantic meaning expressed as a term from the typed lambda-calculus. A universal set of CCG rules determines the way in which these constituents combine together. These can be thought of as algebraic rules of meaning combination and therefore every grammatical sentence corresponds to a CCG derivation tree that comes with its own composition of the meaning of the sentence. Some examples of such derivations are shown in Figure 1 and Figure 2. Here, the occurrences of '/' and '\' in functional categories specify arguments that must be located to the right and left, respectively of the functions. Since CCG binds syntax and semantics together, it is widely used in semantic analysis and parsing, including instruction execution and question answering. Thus, we elected to use CCG to understand and convert natural language into LTL.



| Eventually | A | is | on | B |
|---|---|---|---|---|
| S\S | NP | (S\NP)/PP | PP/NP | NP |
| λP. eventually(P) | A | λP. λx. P(x) | λx. λy. on(y, x) | B |
| | | | PP | |
| | | | λy. on(y, B) | |
| | | S\NP | | |
| | | λx. on(x, B) | | |
| | S | | | |
| | on(A, B) | | | |
| S | | | | |
| eventually(on(A, B)) | | | | |

**Figure 1:** Example CCG Derivation from the Blockworld Domain



| soil data | is | always | sampled | at | waypoint0 |
|---|---|---|---|---|---|
| S/(S\NP) | (S\NP)/(S\NP) | (S\NP)/(S\NP) | S\NP | (S\NP)\(S\NP)/NP | N, NP |
| λP.exists x. soildata(x) & P(x) | λP. λx. P(x) | λP. λx. always(P(x)) | λx.exists y. sampled(x, y) | λy. λP. λx. at(P(x), y) | waypoint0 |
| | | | | (S\NP)\(S\NP) | |
| | | | | λP. λx. at(P(x), waypoint0)) | |
| | | | S\NP | | |
| | | | λx.exists y. at(sampled(x, y), waypoint0) | | |
| | | S\NP | | | |
| | | λx. exists y.always(at(sampled(x, y), waypoint0)) | | | |
| | S\NP | | | | |
| | λx.exists y. always(at(sampled(x, y), waypoint0)) | | | | |
| S | | | | | |
| exists x. exists y.always(at(sampled(x,y), waypoint0)) ^ soil_data(x) | | | | | |

**Figure 2:** Example CCG Derivation from the Rover Domain

As shown above, the first example comes from a simplified domain while the second one is more complicated with both forward/backward application and composition, as well as type raising. The derivation yields a sentence with a compositional interpretation and finally an LTL formula, which demonstrates the use case of CCG to translate natural language into LTL in the real-world problem setting.

## 3 Discussion

The notion of translating from English to LTL to communicate with agents has been examined previously. LTLMoP [7] defines a controlled language and its semantics of the sentences in the language with respect to LTL formulae. The NL2KR [8] platform introduces an interactive multi-stage learning algorithm that, given a user-provided initial lexicon and examples, employs inverse lambda and generalization to learn the semantics. Similar tasks that translate natural language directives into temporal and dynamic logic representations were also presented in [9].

We experimented with the Clark and Curran Parser [10], a large-scale parser that is widely used in semantic tasks. It is trained on CCGbank [11], which is extracted from the Penn TreeBank from the Wall Street Journal (WSJ). We tested the C&C parser on several PDDL domains, including the Rovers domain, which was setup as planetary rovers problems to navigate rovers on a planet surface, finding samples and communicating to lander. Several challenges presented themselves. One difficulty we encountered was that the many PDDL domains are poorly encoded from a KR perspective and not amenable to easy translation. In particular, predicate names are often hyphenated words, rather than being parameterized. For example, the Rovers domain has predicates "$communicate\_soil\_data(X)$" and "$communicate\_rock\_data(X)$" rather than "$communicate(X, Y)$". Also, given the nature of our project, most of the sentences we tested on are imperative sentences. Yet the training samples from WSJ lack this kind of data and fail to yield the correct CCG categories. Therefore, construction of a CCG based on the selected domain abstractions associated with LTL is required, and this process requires in-depth knowledge of CCG. We focused on the Rovers domain and constructed sentences and their CCG derivations accordingly, but further construction and testing are needed. Moreover, we want to automate the process of constructing CCG derivations from the desired domains and make it a systematic process. Potential works include formatting the PDDL domains so that the predicates are fully parameterized and finding ways to automatically construct derivations from the domains using both domain and linguistic knowledge.

## References

[1] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series).* The MIT Press, 2008.

[2] M. Steedman, *The Syntactic Process.* Cambridge, MA, USA: MIT Press, 2000.

[3] A. Gerevini and D. P. Long, "Plan constraints and preferences in PDDL 3 the language of the fifth international planning competition," 2005.

[4] J. A. Baier and S. A. McIlraith, "Planning with first-order temporally extended goals using heuristic search," in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI06)*, Boston, MA, July 2006, pp. 788–795.

[5] J. A. Baier, F. Bacchus, and S. A. McIlraith, "A heuristic search approach to planning with temporally extended preferences," *Artificial Intelligence*, vol. 173, no. 5-6, pp. 593–618, 2009.

[6] A. E. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos, "Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners," *Artificial Intelligence*, vol. 173, no. 5, pp. 619 – 668, 2009, advances in Automated Plan Generation.

[7] C. Finucane, G. Jing, and H. Kress-Gazit, "LTLMoP: Experimenting with language, temporal logic and robot control," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 1988–1993.

[8] N. Vo, A. Mitra, and C. Baral, "The NL2KR platform for building natural language translation systems," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers).* Association for Computational Linguistics, 2015, pp. 899–908.

[9] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn, "What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 4163–4168.

[10] S. Clark and J. R. Curran, "Parsing the WSJ using CCG and log-linear models," in *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ser. ACL '04. Stroudsburg, PA, USA: Association for Computational Linguistics, 2004.

[11] J. Hockenmaier and M. Steedman, "CCGbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank," *Computational Linguistics*, vol. 33, no. 3, pp. 355–396, 2007.