Abstract of COMP 527 Final Project

Anthony Beaudry (261056660)
Claire Yang (260898597)
Tommy Zhou (260913606)

April 24, 2025

1 Motivation and Goal

First-Order Logic (FOL) extends propositional logic with quantifiers, enabling reasoning over arbitrary domains. However it's important to note that all the statements we proved in class and on assignments hold for any domain τ. The FOL system is powerful - it can express statements over a wide range of domains and can even capture some aspects of natural language.

However, FOL alone cannot directly express recursive structures or inductive reasoning. Our goal is to extend FOL with `list` types and recursion principles, making it suitable for expressing and reasoning about programs and data structures that rely on inductive definitions. This extension allows FOL to reason about finite sequences and recursive functions.

2 Related Work

We follow the same natural deduction system and format for FOL as introduced in class, which serves as the foundation for our extensio[1]. The Standard FOL[2] cannot represent data structures like lists or recursive functions while these features are essential to reason about programs and computations.

The inductive types and recursion has been explored in systems like Gödel's System T, which introduces primitive recursion as a logically grounded way to define functions over natural numbers [3][4]. We draw from the principles of System T in our recursion model. Additionally, Pfenning and Pientka's lecture notes [1], [5] provide the foundation for our natural deduction framework related to how natural numbers behave in the FOL system influenced how we structure rules for quantifiers and inductive types in this project.

3 Method of Approach

3.1 Constructing the System

We begin by reviewing the FOL system introduced in class, emphasizing its treatment of types, terms, and quantifiers. Based on the knowledge of using natural number within FOL, we modify the system to support list instead.

We define a new type constructor `list` τ along with two constructors: `nil` for the empty list, and `cons(h, t)` for non-empty lists. These constructors are syntactically similar to the encoding of natural numbers via 0 and suc.

We define an elimination rule for lists that captures the structure of inductive proofs: a base case for `nil`, and a step case that assumes the property for a list t and proves it for `cons(h, t)`. With this, we extend the term grammar to include a recursive operator rec, enabling the definition of terminating recursive functions over lists.

$$\begin{aligned}&\mathsf{rec}\ l\ \mathsf{with}\\ &\quad \mid f(\ \mathsf{nil}\ ) \to M_{\mathsf{nil}}\\ &\quad \mid f(\ \mathsf{cons}\ (h,t)) \to M_{\mathsf{cons}}\end{aligned}$$

We show that our recursive operator mirrors primitive recursion in System T.

3.2 Exploring the System

Our main works on the system are:

- A syntactic and semantic extension of FOL with list types, including formation, introduction, and elimination rules.
- A recursive term constructor rec for lists, grounded in the structure of System T, allowing for well-founded recursive definitions.
- A local soundness and completeness proof sketch for our list elimination rule, including detour elimination for `nil` and inductive reductions for `cons`.
- Introduce functions, properties, and theorems about lists, such as equity between lists and functions such as concatenation to demonstrating the expressive power of our extended system.
- Use Beluga to formally define the syntax of our extended FOL system, including the list type and recursive operator. We encode the rules and properties of our system and verify their correctness.
- Embedding of natural numbers into lists via unary representation, showing that nat can be simulated using list τ.

Together, these components form a simple, minium and expressive extension of FOL, allowing us to reason about list structures and define recursive functions while preserving logical consistency.

4 Statement of Contribution

Note: The supporting document is written in notes style with extra comments to better describe the system. We also included Beluga code to verify the correctness of our system.

Every one of us contributed to the project. Anthony mainly focused on the functions and properties of the system, Tommy mainly focused on the syntax and semantics of the system, and Claire mainly focused on the formalization of the system in Beluga.

References

[1] B. Pientka, "First-order logic - an extended discussion," in *Logic and Computation*.

[2] G. Dowek, "Gödel's system T as a precursor of modern type theory," working paper or preprint, 2006. [Online]. Available: https://inria.hal.science/hal-04046289.

[3] V. K. Gödel, "Über eine bisher noch nicht benützte erweiterung des finiten standpunktes," *Dialectica*, vol. 12, no. 3-4, pp. 280–287, 1958. DOI: https://doi.org/10.1111/j.1746-8361.1958.tb01464.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1746-8361.1958.tb01464.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1746-8361.1958.tb01464.x.

[4] J.-Y. Girard, "Une extension de ĭinterpretation de gödel a ĭanalyse, et son application a ĭelimination des coupures dans ĭanalyse et la theorie des types," in *Proceedings of the Second Scandinavian Logic Symposium*, ser. Studies in Logic and the Foundations of Mathematics, J. Fenstad, Ed., vol. 63, Elsevier, 1971, pp. 63–92. DOI: https://doi.org/10.1016/S0049-237X(08)70843-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0049237X08708437.

[5] F. Pfenning, "Lecture notes on natural numbers," in *Constructive Logic*. [Online]. Available: https://www.cs.cmu.edu/~fp/courses/15317-s23/lectures/11-nat.pdf.