LOGIC OF COMBINATORY LOGIC

SILVIA GHILEZAN

Models of computation can be broadly classified into three categories: sequential models, functional models and concurrent models. *Combinatory logic* (CL) is a well-known functional model of computation which accomplishes the idea to eliminate the need for quantified variables in mathematical logic. It is based on combinators, which are higher order functions, and uses only function application. Combinators where invented a century ago, in the early 1920s, by Moses Schönfinkel. The foundations of combinatory logic were established by Haskell Curry in the 1930s and it has been developing ever since. A recent comprehensive overview can be found in [6]. CL is of the same expressive power as the λ-calculus, the functional model of computation introduced by Alonso Church in the 1930s. With respect to this CL captures all computable functions without using bound variables and the obstacles that emerge from bound variables in λ-calculus are completely avoided.

Typed Combinatory logic is a system in which application is controlled by types assigned to terms. Following the first type system of CL, called *simply types*, various type systems emerged, such as intersection types, dependent types, polymorphic types found their primary applications in programming languages, automated theorem provers, proof assistants, program synthesis. Over time the range of applications has been widened to machine learning, artificial intelligence, cognitive representation, natural language and physics. New applications urge for further research and development of the theory and reasoning about CL both typed and untyped.

Various extensions of combinatory logic, both untyped and typed, have been considered in order to obtain formalisms capable to express new features and paradigms. The extension of the syntax with new constructors such as pairs, records, variants, among others, enables building compound data structures, better organization of data and dealing with heterogeneous collections of values. Extending the syntax with a new operator such as a probabilistic operator shifts the computation to a new paradigm, called probabilistic computation. Herein, the approach we are interested in combines typed combinatory logic with classical propositional logic.

In this talk we discuss and present an extension of the simply typed combinatory logic which is a classical propositional logic for reasoning about combinatory logic. It is called *Logic of Combinatory logic*, denoted by LCL and introduced in [4]. We define its syntax, axiomatic system and semantics. The logic LCL, can be explained in two ways:

- LCL is a logic obtained by extending the simply typed combinatory logic with classical propositional connectives, and corresponding axioms and rules;
- LCL is a logic obtained from classical propositional logic by replacing propositional letters with type assignment statements $M : \sigma$, i.e. typed combinatory terms, where M is a term of the combinatory logic, and σ is a simple type.

The LCL *axiomatic system* has two kinds of axioms:

- non-logical axioms concerned with features of combinatory logic;
- logical axioms concerned with logical connectives.

The LCL *semantics* is based on the notion of applicative structures extended with special elements corresponding to the primitive combinators. It is inspired by the applicative structures introduced in [5], [3] and [2].

As a first result we prove the *soundness and completeness of the equational theory of untyped CL* with respect to the proposed semantics. As a second result we show the *soundness and completeness of the LCL axiomatization* with respect to the proposed semantics. The completeness theorem is proved by an adaptation of the Henkin-style completeness method. We first prove that every consistent set can be extended to a maximal consistent set, which is further used to introduce the notion of a canonical model. We prove then that every consistent set is satisfiable. As a consequence we have the completeness theorem: whenever a formula is a semantical consequence of a set of formulas, it is also a deductive consequence of that set. Further, we prove soundness and completeness of CL, which yields a *new semantics of* CL. In addition, we prove that LCL is a conservative extension of the simply typed CL.

We will conclude the talk with discussing related work ([1]) and possible applications in automated reasoning and knowledge representation.

The present talk is based on joint work with Simona Kašterović ([4]).

References

[1] Michael Beeson. Lambda logic. IJCAR 2004, volume 3097 of *Lecture Notes in Computer Science*, pages 460–474, 2004.

[2] Silvia Ghilezan, Simona Kašterović. Semantics for combinatory logic with intersection types. *Frontiers in Computer Science*, volume 4, 2022. DOI: `https://doi.org/10.3389/fcomp.2022.792570`

[3] Simona Kašterović, Silvia Ghilezan. Kripke semantics and completeness for full simply typed lambda calculus. *Journal of Logic and Computation Volume* 30, issue 8: 1567–1608, 2020. DOI: `https://doi.org/10.1093/logcom/exaa055`

[4] Simona Kašterović, Silvia Ghilezan. Logic of Combinatory Logic. *C*oRR, abs/2212.06675, 2022. DOI: `https://doi.org/10.48550/arXiv.2212.06675`

[5] John C. Mitchell and Eugenio Moggi. Kripke-style models for typed lambda calculus. *Annals of Pure and Applied Logic*, 51(1-2):99–124, 1991.

[6] Stephen Wolfram. *Combinators: A Centennial View*. Wolfram Media, Incorporated, 2021.

University of Novi Sad and Mathematical Institute of the Serbian Academy of Sciences and Arts, Serbia
Email address: `gsilvia@uns.ac.rs`