The Semantics of Categorical Message Passing Language

Alexanna Little

August 2023

Pure 2023 Final Assignment: Research Findings and Synthesis
Research Supervisor: Dr. Robin Cockett

Abstract. The objective of this PURE 2023 project was to develop a document to introduce category theorists and functional programmers to the programming language Categorical Message Passing Language (CaMPL). The semantics of the programming language is formalized in category theory, so the document should be able to introduce functional programming to category theorists and programmers to category theory. This document will be posted to the CaMPL website `campl-ucalgary.github.io`.

1 Introduction

Categorical Message Passing Language (CaMPL) is a functional-style concurrent programming language which implements concurrency by message passing [CP09]. Functional programming is a type of programming in which computations are expressed as functions with input types and output types. In concurrent programming, programs have multiple interacting processes running in parallel. Message passing is a form of concurrency in which processes communicate with each other by passing messages along communication channels.

The semantics of CaMPL is formalized in category theory which is an area of structural mathematics that defines mathematical structures in terms of relationships, which are generally called maps. A categorical semantics of functional programming can be obtained by defining programs as maps from their input types to their output types. CaMPL programs have sequential functions, which are maps from

sequential input types to sequential output types, and concurrent processes which are maps from input polarity channel types to output polarity channel types. Processes can receive messages on a channel and use the information from the message as input for a function. Processes can also send the output from a function as a message along a channel for another process to use [CP09].

Non-determinism was added to CaMPL by introducing a programming feature called a `race`. This allows a process to act differently depending on which channel it receives input on first [Pon21]. Each channel the non-deterministic process is waiting for input from is connected to a different process, so the other processes are in a race to pass their message first. The categorical semantics for non-determinism in CaMPL was introduced in the author's previous PURE 2022 project [Lit22]. Formalizing and proving the correctness of the non-deterministic semantics has been a part of the author's work and research for the last year.

The objective of this PURE 2023 project was to develop a document to introduce CaMPL to people who might be interested in learning how to write programs with it. The intended audiences for the document are theoretical computer scientists and mathematicians interested in category theory and functional programming. The document will be posted to the CaMPL website, `campl-ucalgary.github.io`, so it will be available to anyone interested in learning about CaMPL, functional programming, and category theory.

2 Process/Methods

The CaMPL document was developed by studying literature from the fields of category theory and functional programming and synthesizing previous papers and reports from projects on CaMPL. Previous CaMPL projects described the language using different representations including category theory, programming syntax, and proof theory representations. All of the representations were brought together in the document to explain each programming feature in CaMPL. A review of the literature in category theory was especially important for formalizing the semantics of non-determinism in CaMPL because previous projects on CaMPL had not considered this.

A Curry-Howard-Lambek isomorphism is a method used in functional programming language development to connect the types of functions to propositions in logical proofs and objects in categories. This gives a categorical semantics of functional programming where programs are defined as maps from the object of their input types to the object of their output types. The different representations of CaMPL in category theory, programming syntax, and proof theory representations

constitute a Curry-Howard-Lambek isomorphism which means they are equivalent representations of CaMPL. This means that if one representation of the language is changed, the other representations also need to change. It is useful to have different but equivalent representations of CaMPL because they provide different intuitions for understanding the language and are better at developing different parts of the language.

The programming syntax of CaMPL is the representation of the language that is used to code programs written in CaMPL on a computer. Theoretical computer scientists and mathematicians interested in functional programming should be able to understand how the code for CaMPL is written because it is similar to other programming languages like Haskell and Scala. In previous projects on CaMPL, a programming syntax was created for each programming feature [Kum18]. In the case of non-determinism, the process command `race` was added so non-deterministic processes could be written as processes which include a race [Pon21]. For programmers interested in learning about category theory, the connection from the programming syntax to the proof theory representations and category theory by a Curry-Howard-Lambek isomorphism provides a great introduction. An introduction to the category theory used in the semantics was also included to accommodate programmers who are interested in learning about category theory. It was developed by studying literature in category theory including [Kel65], [Mac63], [CS97], [CP09].

The categorical semantics of CaMPL is the representation of the language from the perspective of category theory which allows one to mathematically prove the language has certain properties. In CaMPL, the channel types and concurrent processes are the objects and maps of a linearly distributive category [CP09]. The categorical semantics provided by a linearly distributive category ensure that processes cannot be connected by channels in a way that creates a cycle [Gir87]. A deadlock is when a cycle of processes are stuck waiting to receive a message from one another, and a livelock is when a cycle of processes are stuck in an endless loop passing messages that cause them to perform useless computations. A previous project on CaMPL showed that CaMPL programs have the property of progress, which is deadlock and livelock freedom [Lyb18]. Theoretical computer scientists and mathematicians interested in category theory should be able to understand why a linearly distributive category can be used to guarantee this property, and by connecting the programming syntax and proof theory representations to the category theory by a Curry-Howard-Lambek isomorphism, this part of the audience should be able to learn how to code in the language. When non-determinism was added to CaMPL, the categorical semantics needed to be changed so that non-deterministic processes could be represented. After the semantics for non-deterministic CaMPL

were formalized, it was important to show that the categorical semantics were still in a linearly distributive category and the property of progress was not lost. Other important programming features such as message passing, protocols and coprotocols also needed to be investigated to ensure they were not lost either. The literature in category theory studied to develop the categorical semantics for non-determinism in the CaMPL document included [JT84], [Cru08], [Geu92], [CS97].

The proofs of the sequent calculus representation of CaMPL programs use a circuit diagram notation. Circuit diagrams are a visual representation of CaMPL programs that can easily communicate how a program written in CaMPL is structured. It is especially useful for understanding the concurrent part of CaMPL because it emphasises the connections between different processes. The sequent calculus is a representation of CaMPL in linear logic which acts as a bridge between the categorical semantics and programming syntax. In the original paper that described CaMPL, the programming syntax was simply expressed as an annotated version of the sequent calculus.

3 Findings/Results

The document that will be posted on the website `campl-ucalgary.github.io` uses a Curry-Howard-Lambek isomorphism to comprehensively introduce CaMPL to a broader audience. It is over 40 pages in length and provides a detailed explanation of each programming feature available in CaMPL. Additionally, it formally describes the semantics of non-determinism and shows how important programming features from the original deterministic CaMPL semantics are lifted to the new non-deterministic CaMPL semantics. An example of how the CaMPL document explained different representations of CaMPL is included here.

The categorical semantics of a non-deterministic process was defined as the join of the different ways the process could act depending on which message it received first [Lit22]. This means a non-deterministic process which will act as either p or q depending on the outcome of the race is defined as $p \vee q$.

Suppose process $p \vee q$ has the following type $p \vee q : W \to A \bullet X, B \bullet Y$. Then $p \vee q$ has one input polarity channel $\gamma : W$ and two output polarity channels $\alpha : A \bullet X$ and $\beta : B \bullet Y$. The $\bullet$ type of an output polarity channel indicates that the process is waiting for input on that channel. In this case, the process is waiting for input on channels α and β.

Suppose $p \vee q$ races channels α and β, and the channel γ is not included in the race. If α wins the race, the process received the input it was waiting for on channel α first and will continue as $p : W \to A \bullet X, B \bullet Y$. If β wins the race, the

process received the input it was waiting for on channel β first and will continue as $q : W \to A \bullet X, B \bullet Y$. Regardless of the outcome of the race, the type of the process must be the same because channels cannot be duplicated, changed, or deleted.

Process $p \vee q$ depicted in the circuit diagram notation is shown in Figure 1. The channels included in the race are indicated by the purple dashed line connection to the outer box. On the inner boxes, the channel which causes the process to continue as that option is also indicated by the purple dashed line.

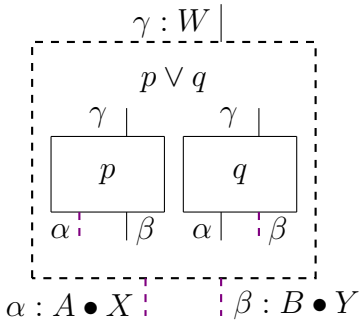

Figure 1: Process $p \vee q$ depicted as a circuit diagram.

Process $p \vee q$ expressed as the join, $\vee$, of processes p and q by the $\vee$ rule in the sequent calculus is shown in Figure 2. If α wins the race, the process will continue as p, and if β wins the race, the process will continue as q.

$$\frac{\{p :: \ \ | \, \gamma : W \Vdash \alpha : A \bullet X, \beta : B \bullet Y, \qquad q :: \ \ | \, \gamma : W \Vdash \alpha : A \bullet X, \beta : B \bullet Y\}}{p \vee q \ \ \textit{where} \ \ \alpha \mapsto p, \ \beta \mapsto q :: \ \ | \, \gamma : W \Vdash \alpha : A \bullet X, \beta : B \bullet Y} \vee$$

Figure 2: The $\vee$ rule in the sequent calculus.

Process $p \vee q : W \to A \bullet X, B \bullet Y$ can be equivalently expressed in the programming syntax as `proc pvq :: | W => Get(A|X), Get(B|Y)` as shown in Figure 3. Process `pvq` has one input polarity channel `gamma` of type `W` and two output polarity channels `alpha` of type `Get(A|X)` and `beta` of type `Get(B|Y)`. If `alpha` wins the race, the process will continue as `p`, and if `beta` wins the race, it will continue as `q`.

```
proc pvq :: | W => Get(A|X), Get(B|Y) =
 | gamma => alpha, beta -> do
      race
           alpha -> p( gamma | => alpha, beta)
           beta -> q( gamma | => alpha, beta)
```

Figure 3: The `race` process command in CaMPL programming syntax.

4 Conclusion

The website has been created as a debut of the CaMPL project to general audiences. A version of the CaMPL compiler will be available to be downloaded so that anyone who is interested in CaMPL can learn the language and write interesting programs. The currently available documentation for CaMPL is quite specialized in either category theory or functional programming. The document created in this PURE 2023 project will be posted to the Documentation section of the website for CaMPL and is intended to bridge the gap between audiences that may be interested in CaMPL but come from different academic backgrounds in theoretical computer science and mathematics. This document is also a stepping stone for the author on the way to creating published work about CaMPL.

5 References

[CP09] J. R. B. Cockett and Craig Pastro. "The Logic of Message Passing". In: *Science of Computer Programming* 74.8 (2009), pp. 498–533. DOI: https://doi.org/10.48550/arXiv.math/0703713.

[Cru08] G. S. H. Cruttwell. "Normed Spaces and the Change of Base for Enriched Categories". Provided by the author. Halifax, Nova Scotia: Dalhousie University, Dec. 2008.

[CS97] J. R. B. Cockett and R. A. G. Seely. "Weakly Distributive Categories". eng. In: *Journal of Pure and Applied Algebra* 114.2 (1997), pp. 133–173. ISSN: 0022-4049.

[Geu92] Herman Geuvers. "Inductive and Coinductive types with Iteration and Recursion". In: *BRA-LF meeting, Edinburgh, Scotland, May 1991*. 1992, pp. 1–25.

[Gir87] J.-Y. Girard. "Linear logic". In: *Theoretical Computer Science* 50.1 (1987), pp. 1–102.

[JT84] Andre Joyal and Myles Tierney. *An extension of the Galois theory of Grothendieck*. eng. Vol. 309. Memoirs of the American Mathematical Society. Providence, R.I: American Mathematical Society, 1984. ISBN: 0821823124.

[Kel65] G. M. Kelly. "Tensor products in categories". In: *Journal of Algebra* 2.1 (1965), pp. 15–37. ISSN: 0021-8693.

[Kum18] Prashant Kumar. "Implementation of Message Passing Language". Provided by the author. MA thesis. Calgary, Alberta: University of Calgary, Feb. 2018.

[Lit22] Alexanna Little. "Semantics for Non-Determinism in the Categorical Message Passing Language". PURE Final Assignment: Research Findings and Synthesis, University of Calgary. Aug. 2022.

[Lyb18] Reginald Lybbert. *Progress for the Message Passing Logic*. Undergraduate Thesis, University of Calgary. 2018.

[Mac63] Saunders MacLane. "Natural Associativity and Commutativity". In: *Rice Institute Pamphlet - Rice University Studies* 49.4 (1963), pp. 28–46.

[Pon21] Jared Pon. *Implementation Status of CMPL*. Undergraduate Thesis Interim Report, University of Calgary. 2021.