FLIN: A Functional Language for Interaction Nets

Ian Mackie
University of Sussex
Brighton, UK

Shinya Sato
Ibaraki University
Ibaraki, Japan

Marc Thatcher
University of Sussex
Brighton, UK

ABSTRACT

Interaction nets are a specific kind of graph rewriting. They have been used successfully as an implementation technique for the lambda calculus, but they are also a programming language in their own right. The purpose of this paper is to promote this idea by designing a new language for writing interaction net programs. We define the language together with a type system, and give a series of examples.

KEYWORDS

programming language, interaction nets, type systems

ACM Reference Format:
Ian Mackie, Shinya Sato, and Marc Thatcher. 2023. FLIN: A Functional Language for Interaction Nets. In *Proceedings of The 35th Symposium on Implementation and Application of Functional Languages (IFL 2023).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

While people often lean towards graphical formalisms over textual ones, the availability of practical examples for computer programming remains limited. Diagrams play a crucial role in enhancing the comprehension of algorithms and programs, effectively conveying intuitive insights and properties. Algorithm animation, as a pedagogical tool, facilitates the explanation of algorithms and, consequently, aids programmers in specifying, writing, and debugging. The purpose of this paper is to define a graphical programming language that can be written in a more traditional functional style: the functions are graphs.

Graph rewriting, grounded in logic, algebraic, and categorical foundations, provides a framework for understanding the dynamics of graphical systems. Specifically, they can model the behaviour of many implementations of functional programming languages, and give insight into the memory allocation (space usage) of a program.

Interaction nets are a particular form of graph rewriting. They lend themselves well to efficient implementation because aspects such as identifying a redex, which is often computationally expensive in general graph rewriting, is very straightforward. However, the killer app for interaction nets is the implementation of the λ-calculus, where optimal implementation has been achieved. However, it is also a programming/specification language very much like term rewriting systems: a user defined set of symbols and rewrite rules can be defined. Although work has been done to compile functional languages to interaction nets, it turns out that many interaction nets are functional programs already: there is a one-to-one correspondence.

The purpose of this paper is to define a new functional programming language for interaction nets, where we naturally capture the functional aspects naturally. Interaction nets are left-linear constructor systems, where patterns can be matched to depth one. Programming languages for interaction nets have typically been defined by flattening of the graphical representation. We give an example to illustrate the ideas.

```
add(x,y) >< Z => x~y
add(x,y) >< S(z) => x~S(a), z~add(a,y)
```

With some practice, it's possible to become fluent in this[1], but the functional representation is quite a bit more natural:

```
add Z y = y
add (S(x)) y = S(Add x y)
```

Both of these languages represent the same interaction net system, as shown in Figure 1.

The point here is that the functional language *is* the interaction net — this is not a compilation. This example works because the function is linear, and the pattern matching is depth 1. Not all programs fit this criteria, for example multiplication is not linear:

```
mult Z y = y
mult (S(x)) y = add (mult x y) y
```

This program does not correspond to an interaction net, but we can make it correspond with some work:

```
mult Z y = Z { eps~y }
mult (S(x)) y = add(mult x y1) y2 { delta(y1,y2) ~ y}
```

Now this corresponds directly to the interaction net rules shown in Figure 2, but we needed to introduce some additional features to the language to allow this. Specifically, we have introduced additional interaction net agents (eps and delta) and we have also introduced a new notation, where parts of the function are represented differently.

The full version of this paper defines a functional language that corresponds to interaction nets in a direct way. We give an operation semantics of this language, a translation of nets to this language, a type system and examples. The rest of full paper is organised as follows. In the next section we review interaction nets and the programming language inpla. Then we define a new language and give examples, a translation function from the language of interaction net and give a notion of type and type system.

IFL 2023, August 29–31, 2023, Braga, Portugal

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

[1] the authors have not yet reached this level

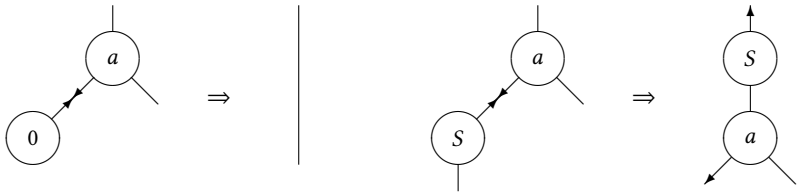

Figure 1: Example: addition

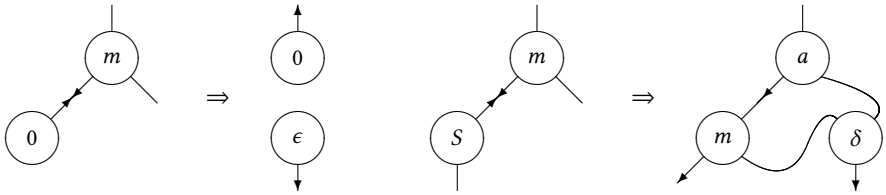

Figure 2: Example: multiplication