Università di Genova | DIPARTIMENTO INFORMATICA, BIOINGEGNERIA, ROBOTICA E INGEGNERIA DEI SISTEMI

Computer Science Workshop
PhD program in Computer Science and Systems Engineering

Enhanced Regular Corecursion for Data Streams

Pietro Barbieri

Introduction

As we venture deeper into the Internet of Things (IoT) era, stream processing is becoming increasingly important. With our recent work, we propose a stream calculus to lay the foundations of a tool for testing of IoT systems and real time analysis of unbound data series.

Main Objectives

- Develop a calculus to define and manipulate infinite streams
- Provide procedures to check well-definedness and equality of streams
- Achieve a good compromise between expressive power and decidability

State of the Art

Two complementary approaches to manipulate streams:

Lazy Evaluation

Streams defined by arbitrary functions and inspected as much as needed.

Pros

- Widely known and well-established solution for stream processing
- Supports both regular (cyclic) and non-regular streams

Cons

- Operations that need to inspect the whole stream cannot be computed
- Allows the definition of undefined streams due to high expressive power

Regular Corecursion

Streams are represented by finitary equational systems. Non-termination of recursive stream functions is avoided by keeping track of already processed calls.

Pros

- The entire stream can be inspected because it is finitely represented by an equational system

Cons

- Fails to model non-regular streams

Our Solution

- Enhances regular corecursion
  - Not only constructors are allowed in equations defining streams
  - Besides regular streams, supports also a subset of non-regular streams
- Provides procedures to check well-definedness and equality of streams

Examples

Simple cyclic streams

```
repeat(n) = n:repeat(n)   //n:n:n:n:...
one_two() = 1:2:one_two()   //1:2:1:2...
```

Note: `[+]` `[*]` `[/]` are pointwise operations on streams, `^` computes the tail.

Non-regular streams

```
nat() = 0:(nat()[+]repeat(1))
fib() = 0:1:(fib()[+]fib()^)
```

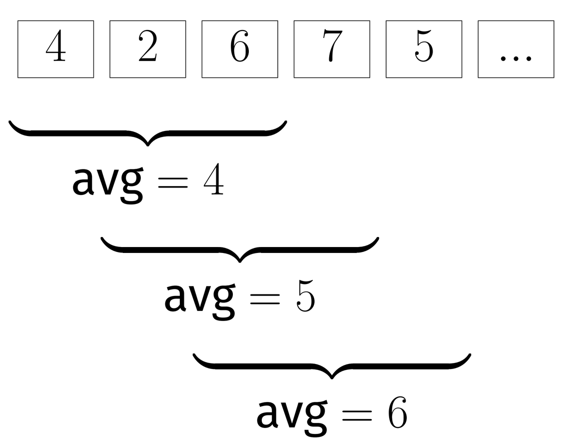
nat() ⟶ | 0 | 1 | 2 | 3 | ... |

fib() ⟶ | 0 | 1 | 1 | 2 | ... |

Functions for stream processing

```
aggr(n,s) = if n<=0 then repeat(0)
            else s[+]aggr(n-1,s^)
avg(n,s) = aggr(n,s)[/]repeat(n)
```

Below you find an example of execution of avg over a window of size 3



Checking equality of streams

- The stream of all ones `s = 1:s` is equal to its tail:

`s == s^` → `s == (1:s)^` → `s == s`

- If we add more ones to stream `s` we still get `s` as a result:

`1:1:s == 1:s` → `1:s == s` → `1:s == 1:s` → `s == s`

Forthcoming Research

- Make function definitions more *flexible*
  The user is allowed to specify the behaviour in presence of a cycle
- Introduce a static type system to prevent runtime errors

Reference papers:

Davide Ancona, Pietro Barbieri, Elena Zucca. Enhanced Regular Corecursion for Data Streams. ICTCS '21

Davide Ancona, Pietro Barbieri, Elena Zucca. Enhancing Expressivity of Checked Corecursive Streams. FLOPS '22

CONTACTS

Davide Ancona
Davide.Ancona@unige.it

Elena Zucca
Elena.Zucca@unige.it

Pietro Barbieri
pietro.barbieri@edu.unige.it

CSW