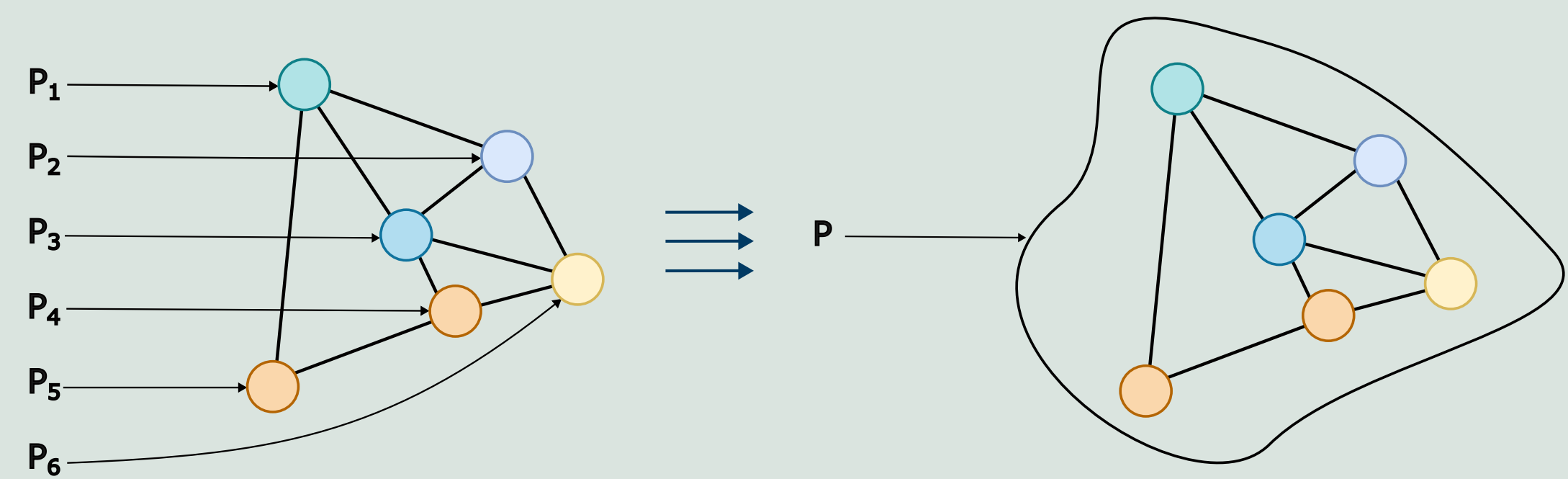


# Towards Multiplatform Self-Organizing Aggregate Computing Systems

Angela Cortecchia  
Fellow researcher @GARR & (soon) PhD student @UniBo

## Aggregate Computing



Aggregate Computing (AC) is a macro-programming approach that supports behavior compositionality for self-organizing systems. It is based on the Field Calculus abstractions, which operates in terms of computational fields.

AC aims to shift the the definition of the behavior from the single device to a collective behavior of heterogeneous devices.

## State of the art

FCPP

val y = minHood(nbr(id))  
val ids = nbr(id)  
// astonishing behavior  
minHood(ids) // -> returns local ID

Reified fields

Manipulating fields can facilitate the creation of more intricate and efficient programs.

FP

let n = nbr(id)  
let y = mux (n > 0.5) { [1,2] } else { 0 }  
y.first() //possible error at runtime

Static checking

Type checking at compile time can prevent unexpected errors at runtime.

Transparent alignment

Alignment can be performed behind the scenes by a compiler plugin.

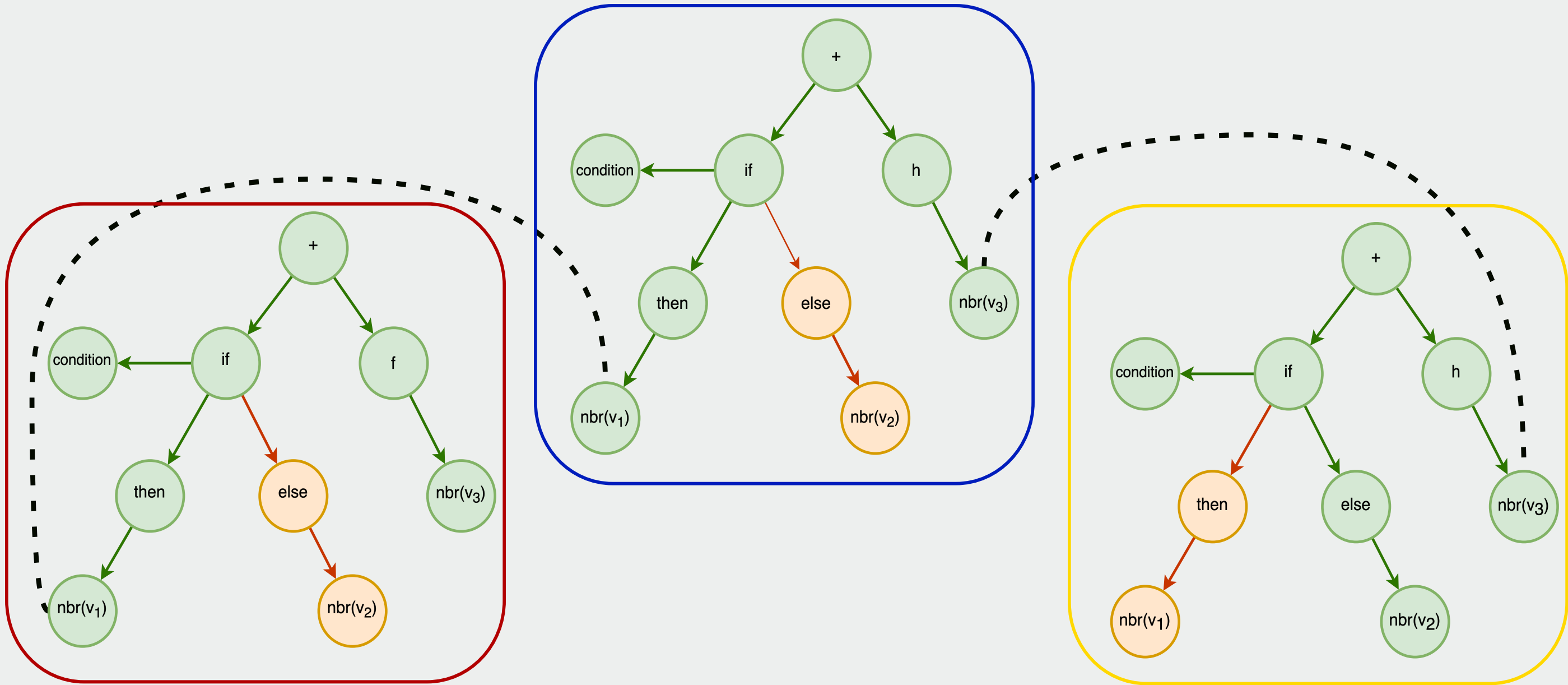
fun Aggregate<Double>.x() {  
 val n = neighboring(localID)  
 n.map { listOf(it, 2.0, 3.0) }.first() // Field<Double>  
 // Typed fields prevent wrong calls at compilation time  
 n.map { if (it > 0.5) listOf(1, 2) else 0 } // Field<Any>  
 .first() // Error: receiver type mismatch -- CORRECT!  
}

## Alignment

In Aggregate Computing, devices communicate between them without an explicit notion of sending messages, thanks to "alignment".

How?

Two (ore more) devices are identified as "aligned" when they have reached the same point in the program. Devices considered within the same neighborhood will be able to communicate with each others if they are indeed running the same program.



## Idea

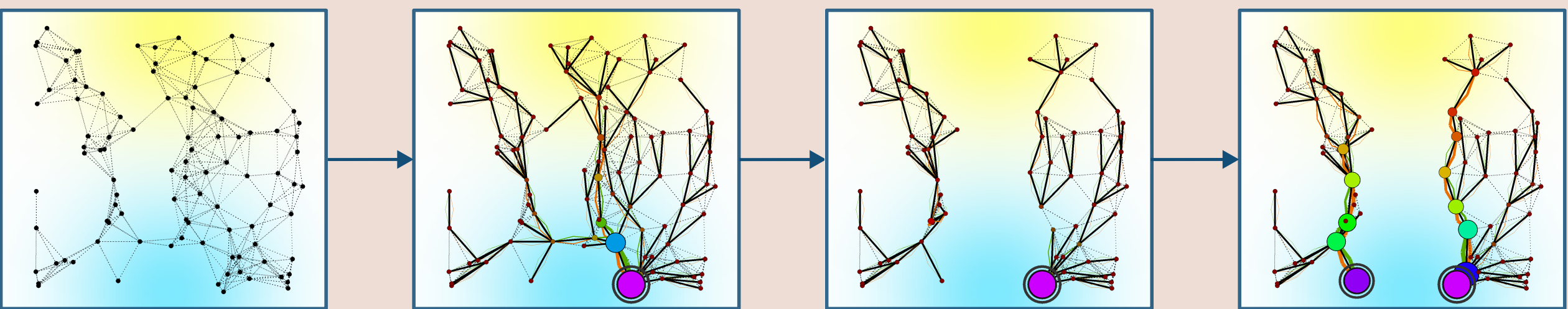
Kotlin Multiplatform allows multiplatform programming and modifications at compiler level. Leveraging on those technologies allowed the creation of an internal DSL prototype for aggregate programming, called **Collektive**.



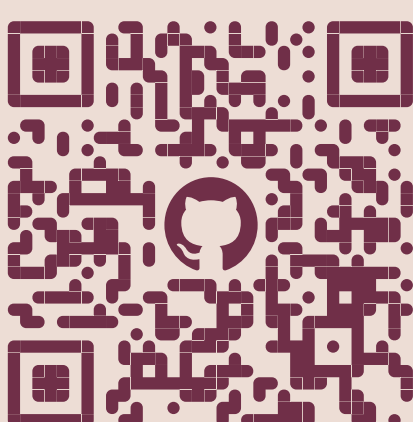
Language	DSL type	JVM	JS	Wearables	Reified Fields	Transparent Alignment	Automatic Complete Alignment
Proto	external	X	X	X	✓	✓	✓
Protelis	external	✓	X	X	✓	✓	✓
ScaFi	internal (Scala)	✓	✓	~	X	~	X
FCPP	internal (cpp)	X	X	~	✓	X	X
Collektive	internal (Kotlin)	✓	✓	✓	✓	✓	✓

✓ Supported  
X Not supported  
~ Partially supported

## Application Example



Used for a generalization of the Vascular Morphogenesis Controller algorithm: from a single node it is able to create structures based on the envirnoment's information. This approach can be applied to vascular tissues, organization management, robot swarms and others.



## Future Works

- Ensuring comprehensive evaluation through simulations execution on different platforms.
- Development of a standard library, with support for reusable building blocks.
- Study and development of collective operating systems with Aggregate Computing.