Miking Workshop, 2023

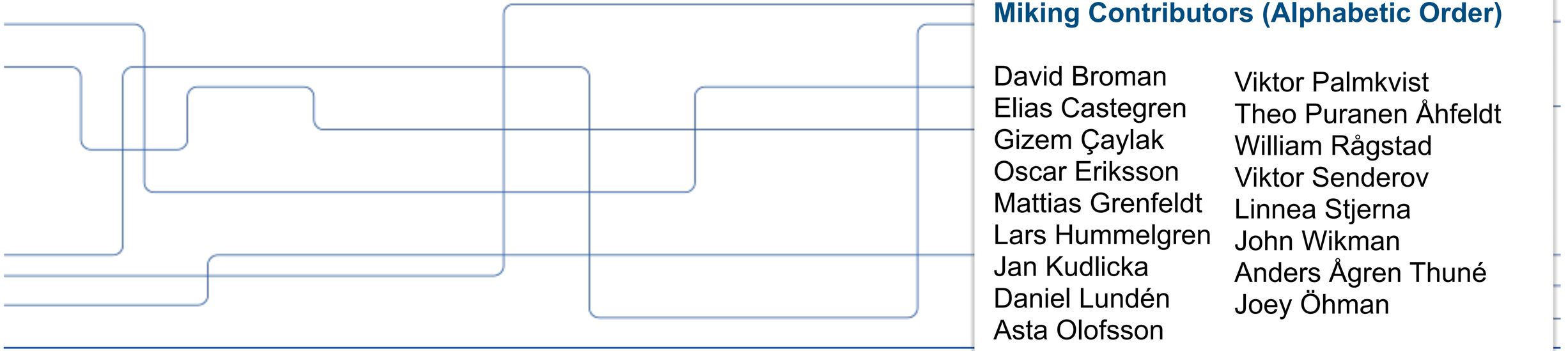
digital futures

Digital Futures Hub
Stockholm, November 23, 2023

David Broman
Professor, KTH Royal Institute of Technology
Visting Professor, Stanford University
Associate Director Faculty, Digital Futures

Vetenskapsrådet (VR)

Financially supported by the Swedish Foundation for Strategic Research.



Miking Contributors (Alphabetic Order)

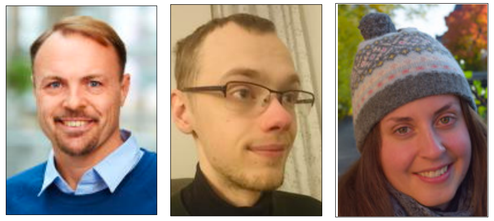
David Broman
Elias Castegren
Gizem Çaylak
Oscar Eriksson
Mattias Grenfeldt
Lars Hummelgren
Jan Kudlicka
Daniel Lundén
Asta Olofsson
Viktor Palmkvist
Theo Puranen Åhfeldt
William Rågstad
Viktor Senderov
Linnea Stjerna
John Wikman
Anders Ågren Thuné
Joey Öhman

Part I
Research Group

Part II
Workshop Overview

Part III
Overview of the Miking Framework

Part I
Research Group




Programming and Modeling Languages Group

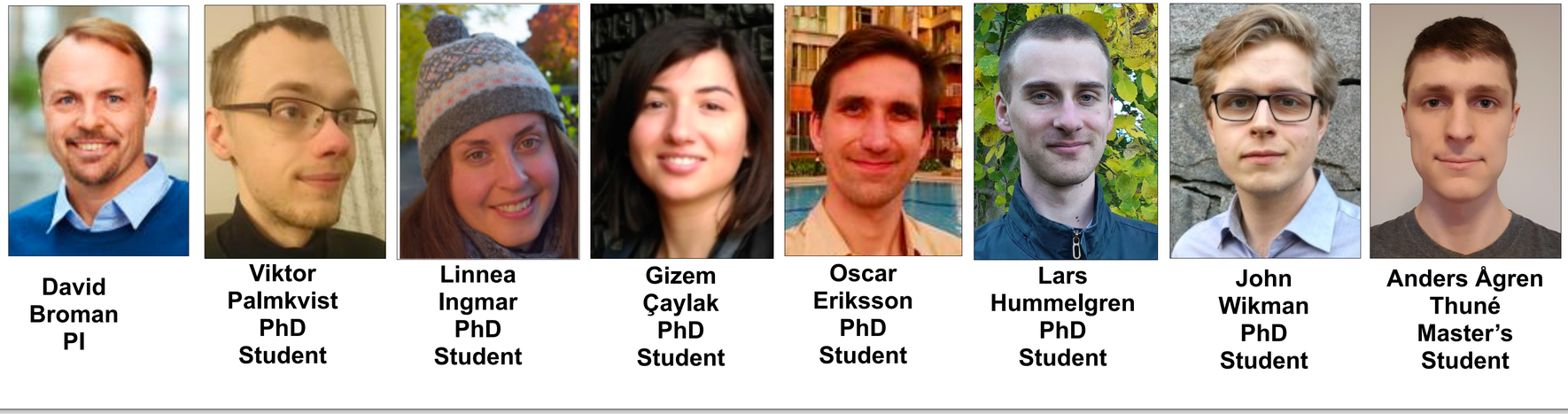
Probabilistic Machine Learning

Intersection of Research Areas

Cyber-Physical Systems

$\lambda^{<?>}$

Programming Languages and Compilers



David Broman PI

Viktor Palmkvist PhD Student

Linnea Ingmar PhD Student

Gizem Çaylak PhD Student

Oscar Eriksson PhD Student

Lars Hummelgren PhD Student

John Wikman PhD Student

Anders Ågren Thuné Master's Student

Why models?

Model

Model

Perform experiments

Model

Why?
- Get insights
- Cheaper
- Too dangerous
- May not exist
- Easier

System

Scientists

Our group:

Develop domain-specific modeling languages and compilers

Overall Research Challenge

Combine:
- high-level of abstraction modeling with
- automatically generated efficient compilers

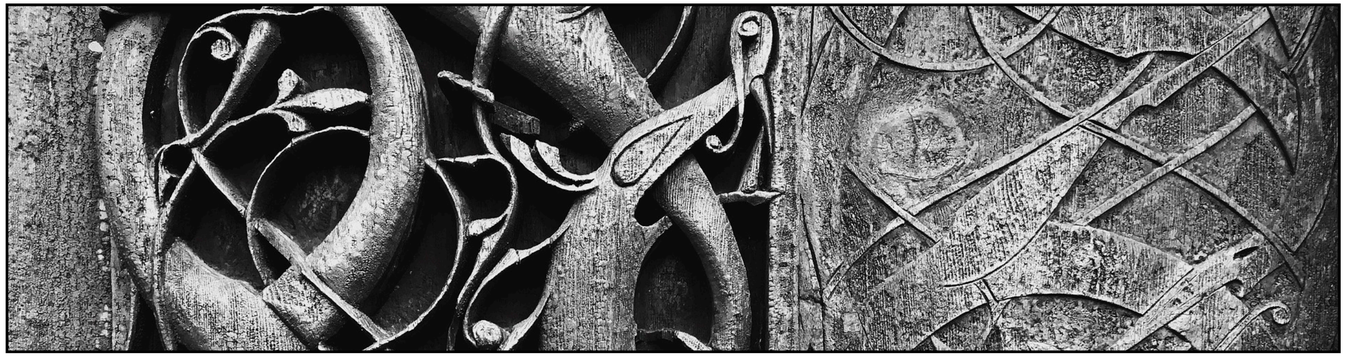
Engineers

Insights from (Lee, 2016) and (Cellier, 1996)

Part II
Workshop Overview

Agenda

Miking Workshop 2023



10.00 Registration

10.15 Welcome and introduction to Miking. Speaker: David Broman

11.00 Coffee break

11:15 Session 1: Types and Parsing in Miking

- Title: *Universal Collection Types*. Speaker: Viktor Palmkvist
- Title: *Towards LR parsing in Miking - key ideas and challenges*. Speaker: John Wikman
- Title: *A new polymorphic type system for Miking*. Speaker: Anders Ågren Thuné

12.00 Lunch

13.00 Hacking session 1: Getting started and playing around Organizers: The Miking core team

14:30 Session 2: Tuning and Code Generation

- Title: *Programming with Context-Sensitive Holes using Dependency-Aware Tuning*. Speaker: Linnea Stjerna
- Title: *Functional programming on the JVM*. Speaker: Asta Olofsson

15.00 Coffee break

15:30 Session 3: Domain-Specific Languages (DSLs) in Miking

- Title: *TreePPL - a new DSL in Miking for Phyologenetics*. Speaker: Viktor Senderov
- Title: *Real-time Probabilistic Programming, a DSL in Miking*. Speaker: Lars Hummelgren
- Title: *Equation-based modeling and efficient simulation in Miking*. Speaker: Oscar Eriksson

16.15 Hacking session 2: Try out your favorite DSL or hack on the compiler Organizers: The Miking core team

17.00 Conclusions and more happy hacking!

Part I
Research Group

Part II
Workshop Overview

Part III
Overview of the Miking Framework

Agenda

10.00 Registration

10.15 Welcome and introduction to Miking. Speaker: David Broman

11.00 Coffee break

11:15 Session 1: Types and Parsing in Miking

- Title: *Universal Collection Types*. Speaker: Viktor Palmkvist
- Title: *Towards LR parsing in Miking - key ideas and challenges*. Speaker: John Wikman
- Title: *A new polymorphic type system for Miking*. Speaker: Anders Ågren Thuné

12.00 Lunch

13.00 Hacking session 1: Getting started and playing around Organizers: The Miking core team

14:30 Session 2: Tuning and Code Generation

- Title: *Programming with Context-Sensitive Holes using Dependency-Aware Tuning*. Speaker: Linnea Stjerna
- Title: *Functional programming on the JVM*. Speaker: Asta Olofsson

15.00 Coffee break

15:30 Session 3: Domain-Specific Languages (DSLs) in Miking

- Title: *TreePPL - a new DSL in Miking for Phyologenetics*. Speaker: Viktor Senderov
- Title: *Real-time Probabilistic Programming, a DSL in Miking*. Speaker: Lars Hummelgren
- Title: *Equation-based modeling and efficient simulation in Miking*. Speaker: Oscar Eriksson

16.15 Hacking session 2: Try out your favorite DSL or hack on the compiler Organizers: The Miking core team

17.00 Conclusions and more happy hacking!

Agenda

13.00 Hacking session 1: Getting started and playing around Organizers: The Miking core team

14:30 Session 2: Tuning and Code Generation

- Title: *Programming with Context-Sensitive Holes using Dependency-Aware Tuning*. Speaker: Linnea Stjerna
- Title: *Functional programming on the JVM*. Speaker: Asta Olofsson

15.00 Coffee break

10.00 Registration

10.15 Welcome and introduction to Miking. Speaker: David Broman

11.00 Coffee break

11:15 Session 1: Types and Parsing in Miking

- Title: *Universal Collection Types*. Speaker: Viktor Palmkvist
- Title: *Towards LR parsing in Miking - key ideas and challenges*. Speaker: John Wikman
- Title: *A new polymorphic type system for Miking*. Speaker: Anders Ågren Thuné

12.00 Lunch

15:30 Session 3: Domain-Specific Languages (DSLs) in Miking

- Title: *TreePPL - a new DSL in Miking for Phyologenetics*. Speaker: Viktor Senderov
- Title: *Real-time Probabilistic Programming, a DSL in Miking*. Speaker: Lars Hummelgren
- Title: *Equation-based modeling and efficient simulation in Miking*. Speaker: Oscar Eriksson

16.15 Hacking session 2: Try out your favorite DSL or hack on the compiler Organizers: The Miking core team

17.00 Conclusions and more happy hacking!

Agenda

10.00 Registration

10.15 Welcome and introduction to Miking. Speaker: David Broman

11.00 Coffee break

11:15 Session 1: Types and Parsing in Miking

- Title: *Universal Collection Types*. Speaker: Viktor Palmkvist
- Title: *Towards LR parsing in Miking - key ideas and challenges*. Speaker: John Wikman
- Title: *A new polymorphic type system for Miking*. Speaker: Anders Ågren Thuné

12.00 Lunch

13.00 Hacking session 1: Getting started and playing around Organizers: The Miking core team

14:30 Session 2: Tuning and Code Generation

- Title: *Programming with Context-Sensitive Holes using Dependency-Aware Tuning*. Speaker: Linnea Stjerna
- Title: *Functional programming on the JVM*. Speaker: Asta Olofsson

15.00 Coffee break

15:30 Session 3: Domain-Specific Languages (DSLs) in Miking

- Title: *TreePPL - a new DSL in Miking for Phyologenetics*. Speaker: Viktor Senderov
- Title: *Real-time Probabilistic Programming, a DSL in Miking*. Speaker: Lars Hummelgren
- Title: *Equation-based modeling and efficient simulation in Miking*. Speaker: Oscar Eriksson

16.15 Hacking session 2: Try out your favorite DSL or hack on the compiler Organizers: The Miking core team

17.00 Conclusions and more happy hacking!

Part II
Overview of the Miking Framework

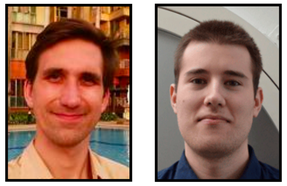
Miking (the Meta vIKING)

Objectives:

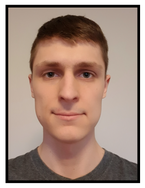
- Platform for constructing heterogeneous domain-specific modeling languanges (DSLs)
- Polymorphic static type system (based on FreezeML).
- Bootstrapping compiler
- Target constrained real-time systems as well as offline distributed computations
- Efficient compiler - different target platforms
- Research platform
- Open source (MIT license)

The Vision of Miking

David Broman. **A Vision of Miking: Interactive Programmatic Modeling, Sound Language Composition, and Self-Learning Compilation.** In Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2019), Athens, Greece, ACM, 2019.








Interactive Programmatic Modeling (Part I)

Input Data

Programmatic Model

Interactive Views

Miking Framework

Efficient Compilation (Part III)

Partial Evaluation

Residual Program

Online learning feedback

Hetero-geneous Execution Platforms

Offline learning feedback

Sound Language Composition (Part II)

Language Fragments




Related Work

Compiler construction

- Standard Lex, Yacc (external DSL)
- JastAdd (Ekman & Hedin, 2007)

Embedded DSLs

- Haskell DSELs, e.g., Fran (Ellito & Hudak, 1997), Lava (Bjesse et al. 1998, FHM (Nilsson et al., 2003)
- Scala, e.g. Lightweight modular staging (Rompf and Odersky, 2010)
- Shallow embedding and PE (Leißa et al., 2015)

Preprocessing and template metaprogramming

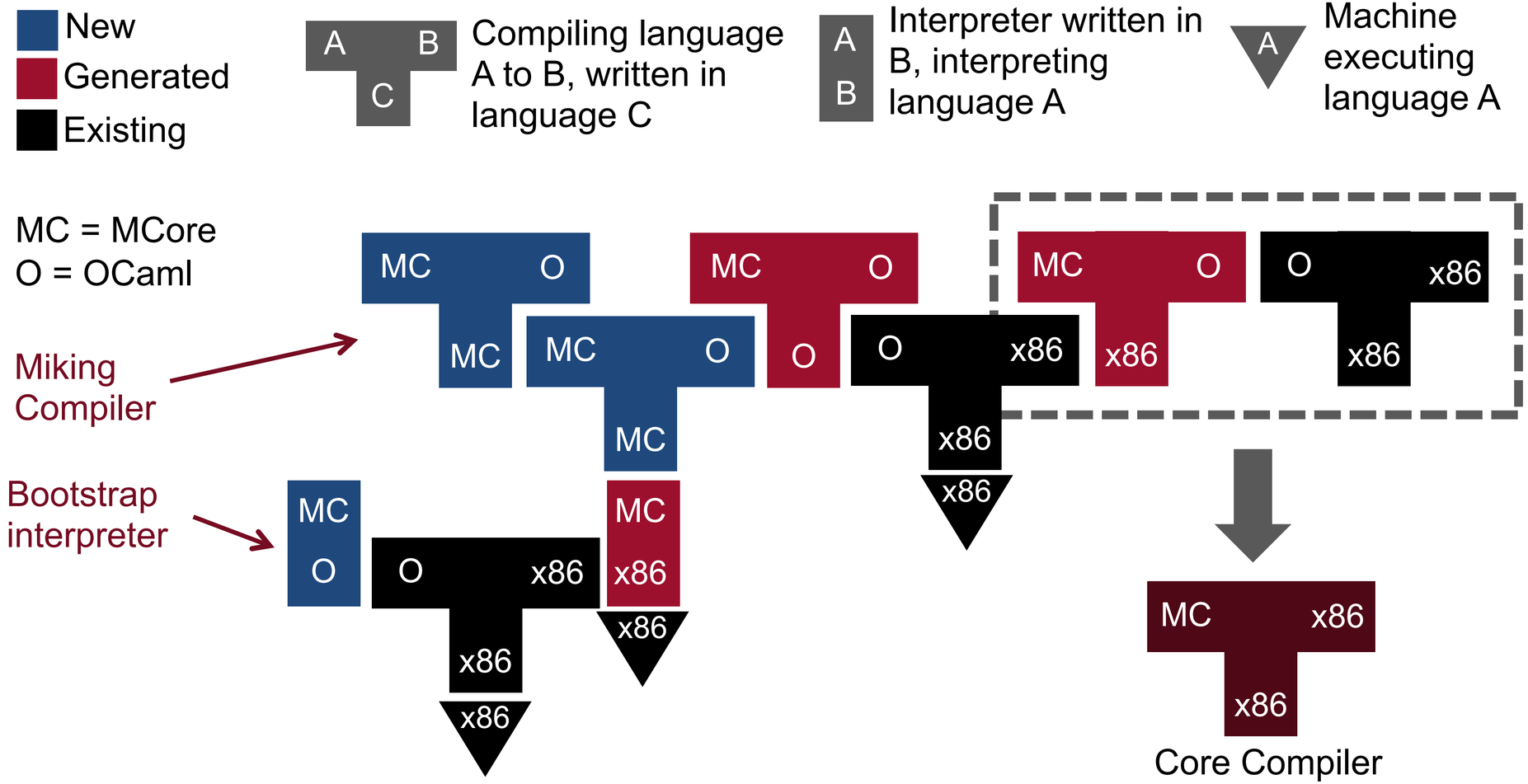
- C++ Templates (Veldhuizen, 1995)
- Template Haskell (Sheard & Peyton Jones, 2002)
- Stratego/XP (Bravenboer et al., 2008)

Language Workbenches and Languages for creating languages

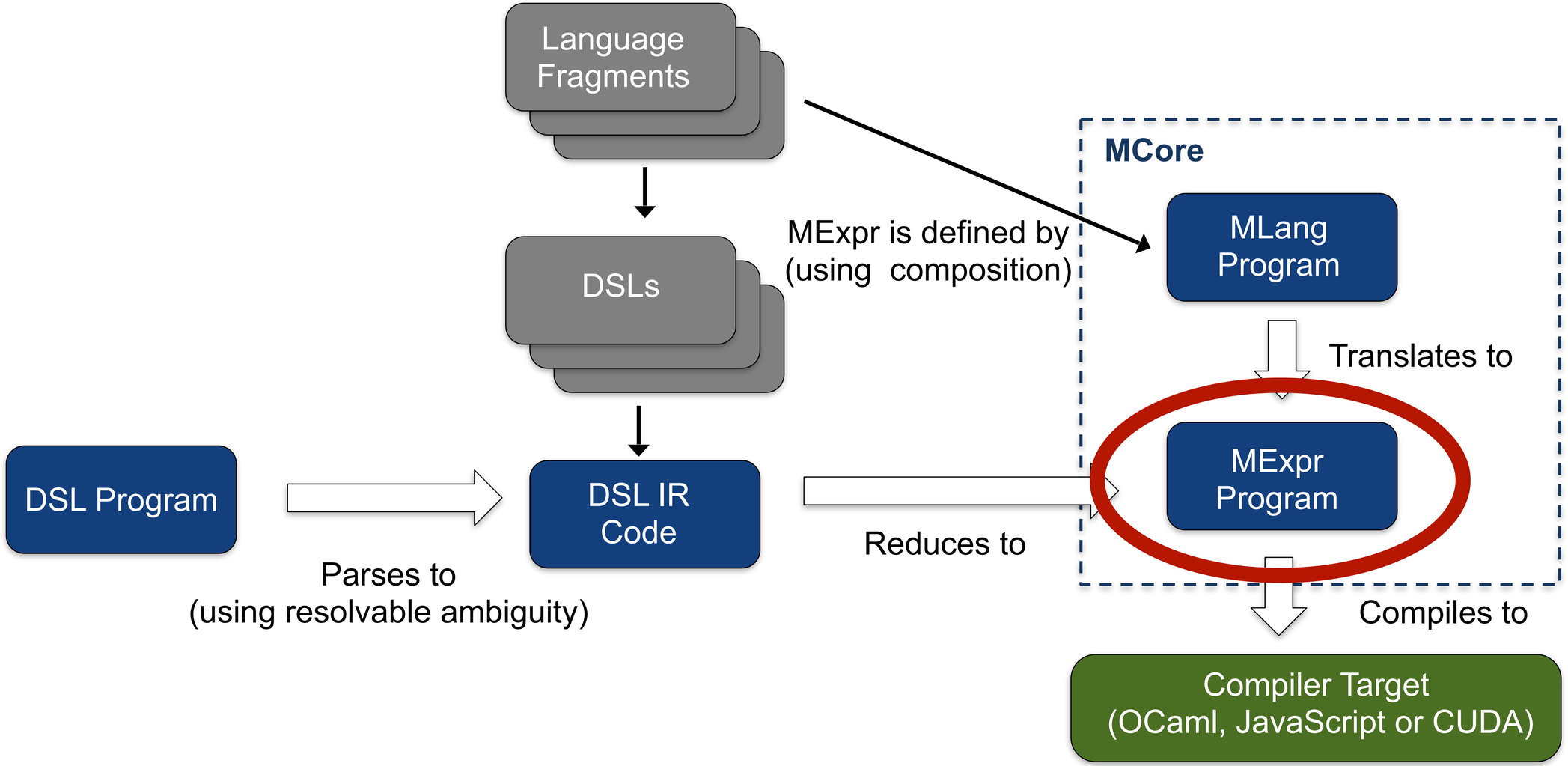
- SugarJ, MPS, Spoofax, RASCAL, MetaEdit+, Enso¯ , Racket etc.

Miking

Bootstrapping the Miking Compiler



Overview of the Toolchain



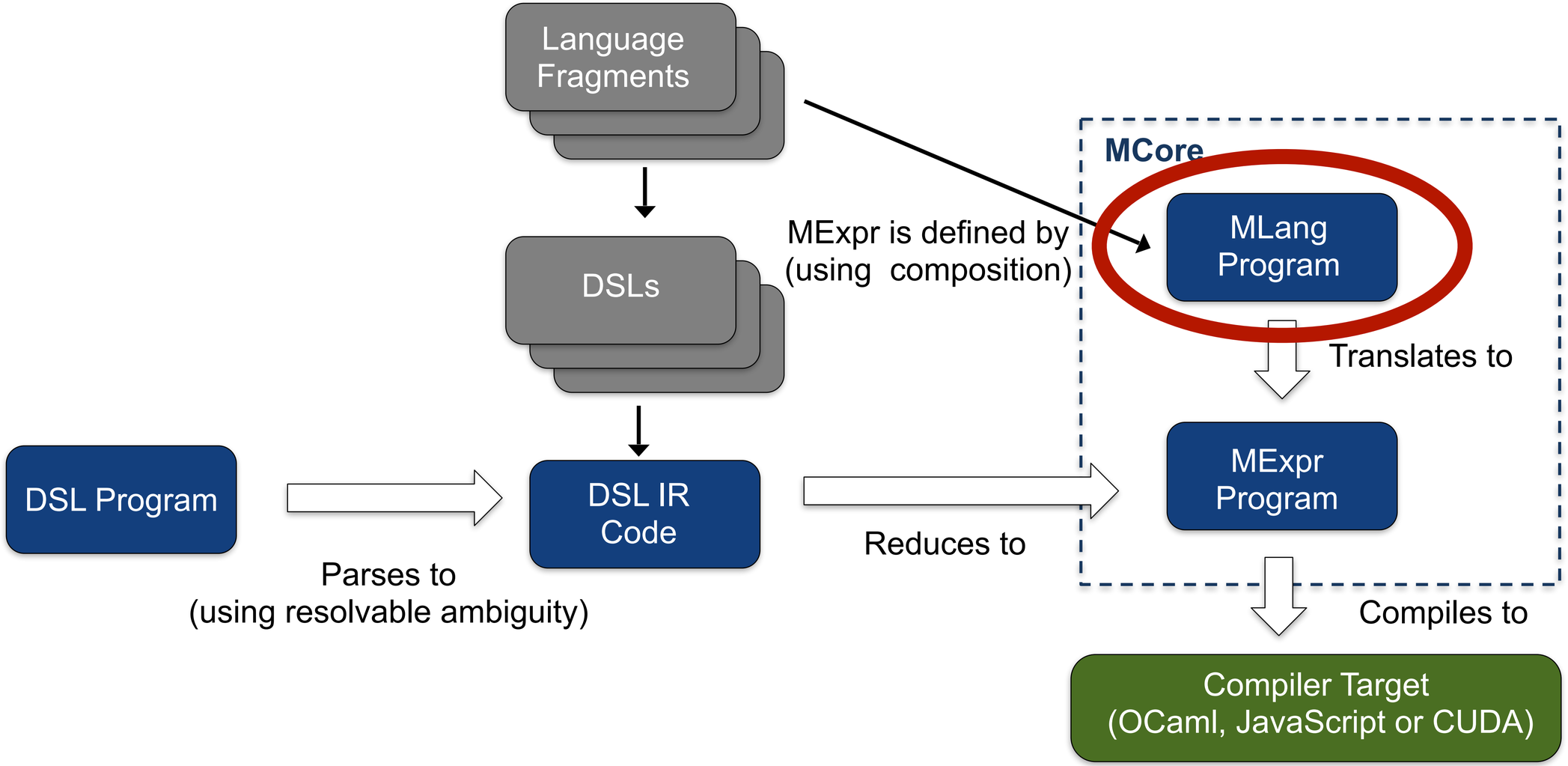
MExpr - the Miking IR

```
type Tree in
con Node : (Tree,Tree) -> Tree in
con Leaf : (Int) -> Tree in


recursive
  let count = lam tree.
    match tree with Node (left,right) then
      addi (count left) (count right)
    else match tree with Leaf v then
      v
    else error "Unknown node"
in


let tree3 = Node(Node(Leaf(3),Node(Leaf(2),Leaf(6))),Leaf(12)) in
utest count tree3 with 23 in
()
```

Features

- Functional Intermediate language
- Polymorphic Type System - statically typed with type inference
- Intermediate representation - concrete syntax very close to abstract syntax
- Small but complete. Eager, includes references, arrays, sequences, algebraic data types, pattern matching, etc.

Overview of the Toolchain



MLang: Language Fragments and Composition

`syn`: defines extensible constructors

`sem`: define extensible functions

```
lang Arith
  syn Expr =
  | Num Int
  | Add (Expr, Expr)

  sem eval =
  | Num n -> Num n
  | Add (e1,e2) ->
    match eval e1 with Num n1 then
      match eval e2 with Num n2 then
        Num (addi n1 n2)
      else error "Not a number"
    else error "Not a number"
end
```

```
lang MyBool
  syn Expr =
  | True()
  | False()
  | If (Expr, Expr, Expr)

  sem eval =
  | True() -> True()
  | False() -> False()
  | If(cnd,thn,els) ->
    let cndVal = eval cnd in
    match cndVal with True() then eval thn
    else match cndVal with False() then eval els
    else error "Not a boolean"
end
```

Independent language fragment, using the same `syn` and `sem` names

```
mexpr
use Arith in
utest eval (Add (Num 2, Num 3)) with Num 5 in
()
```

`use`: using a language fragment in an expression

```
lang ArithBool = Arith + MyBool

mexpr
use ArithBool in
utest eval (Add (If (False(), Num 0, Num 5), Num 2))
  with Num 7 in
()
```

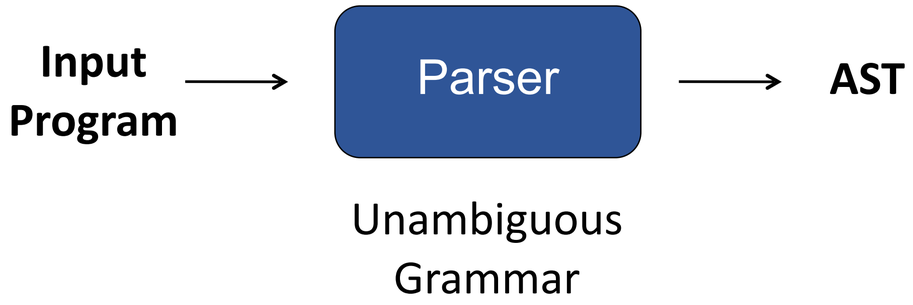
Composing together language fragments

Features

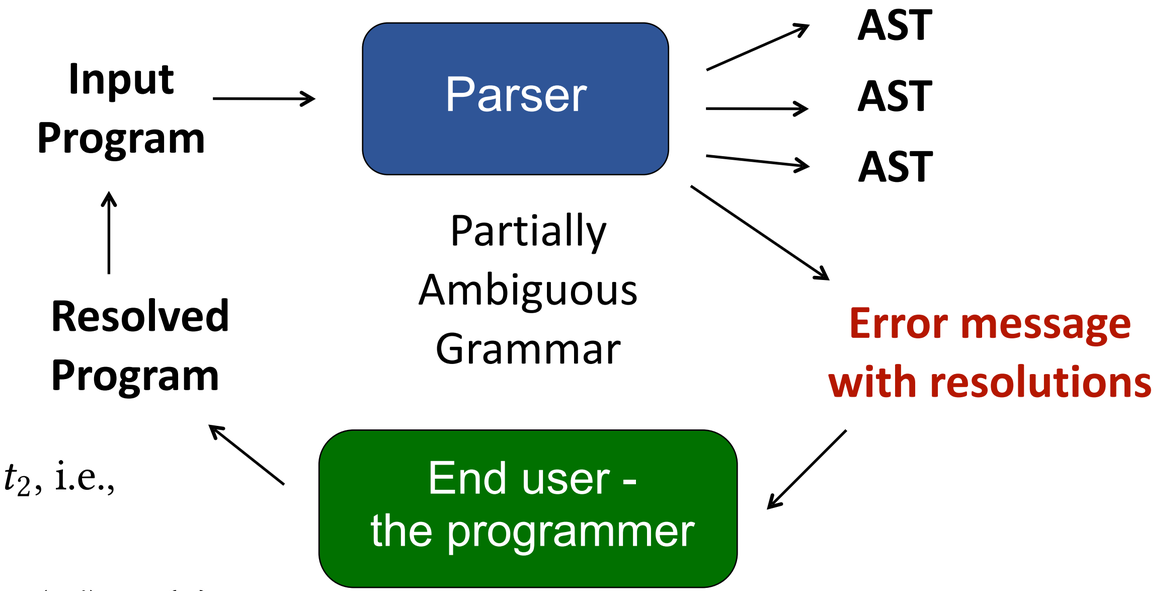
- Order-independent pattern matching composition
- Many semantic functions, e.g. ANF transformation, CPS transformation, lambda lifting, symbolizer, etc.

Statically Resolvable Ambiguity

Traditional Approach



Resolvable Ambiguity



Definition 2.1. A program p is *ambiguous* if $\exists t_1, t_2 \in parse(p).\ t_1 \neq t_2$, i.e., it can parse as at least two distinct ASTs.

Definition 2.2. A program p is *resolvable* if $\forall t \in parse(p).\ \exists p'.\ parse(p') = \{t\}$.

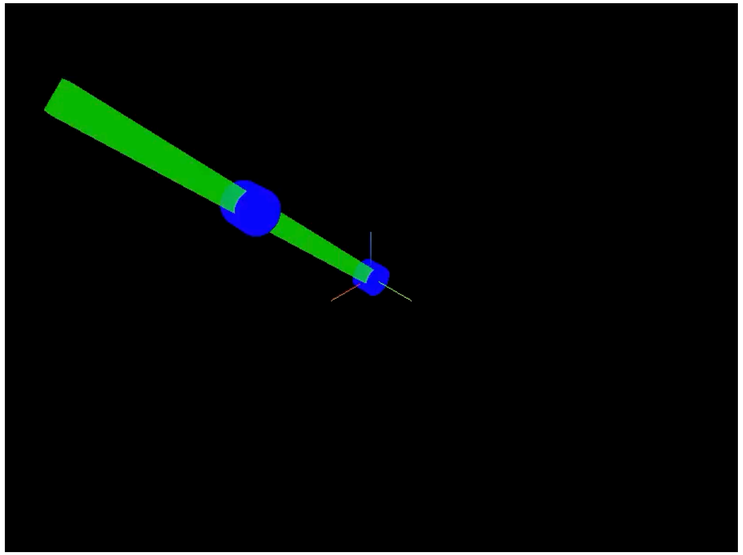
The Static Resolvable Ambiguity Problem:
Statically guarantee that the end user can resolve all accepted programs.

Viktor Palmkvist, Elias Castegren, Philipp Haller, and David Broman. **Resolvable Ambiguity: Principled Resolution of Syntactically Ambiguous Programs**. In Proceedings of International Conference on Compiler Construction (CC), ACM 2021.

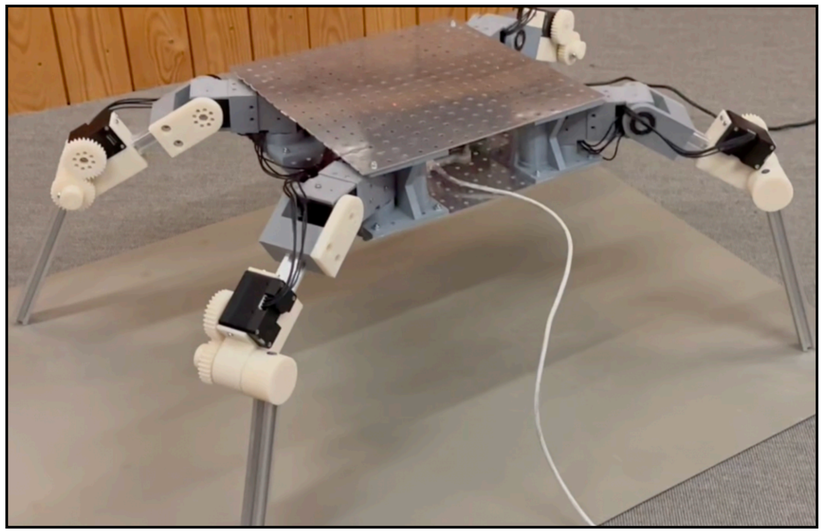
Viktor Palmkvist, Elias Castegren, Philipp Haller, and David Broman. **Statically Resolvable Ambiguity**. In Proceedings of the ACM on Programming Languages, Issue POPL, ACM 2023.

Ongoing Application Areas

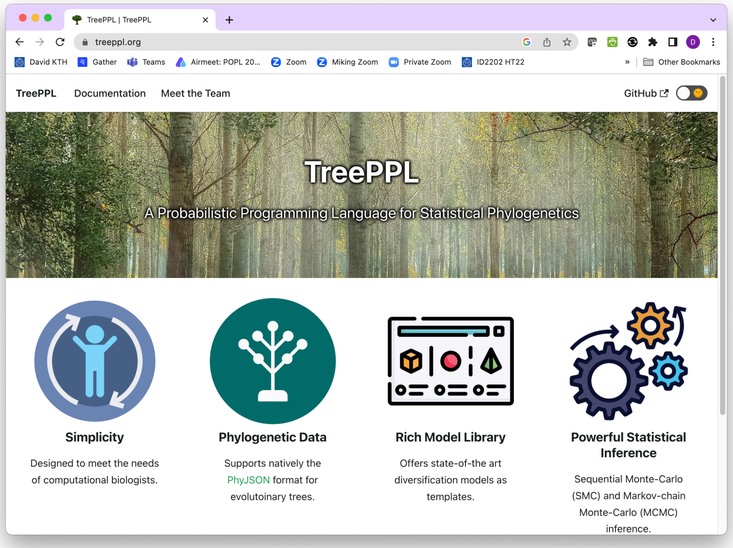
Equation-Based Modeling and Physical Simulation



Robotics and CPS

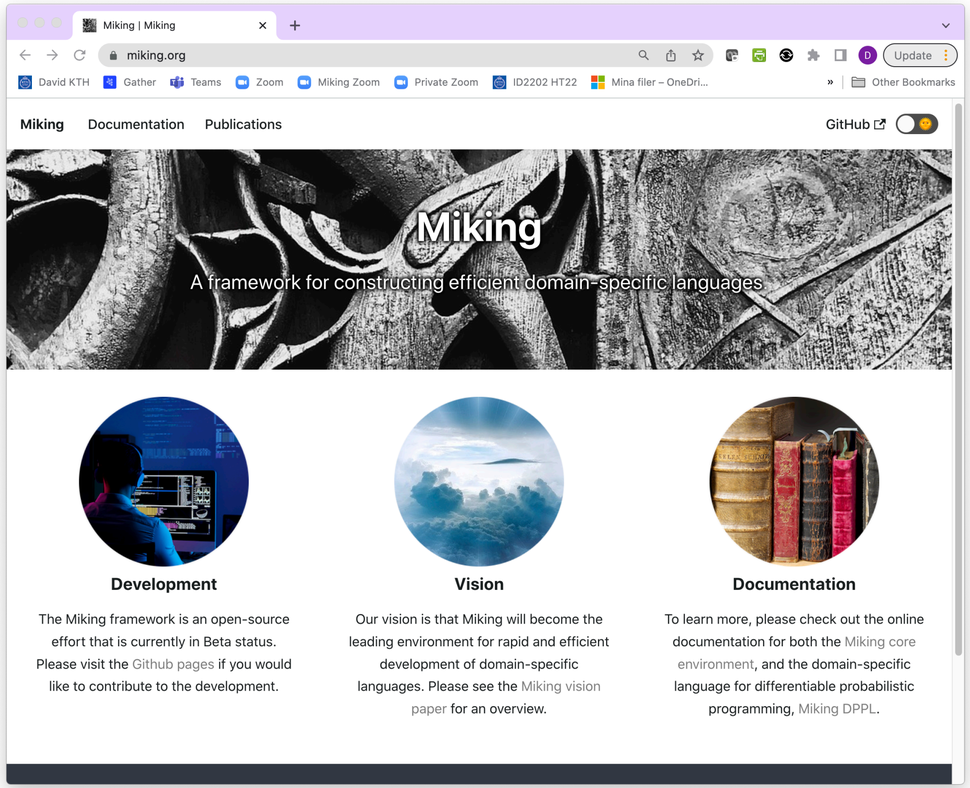


TreePPL - a DSL for phylogenetics

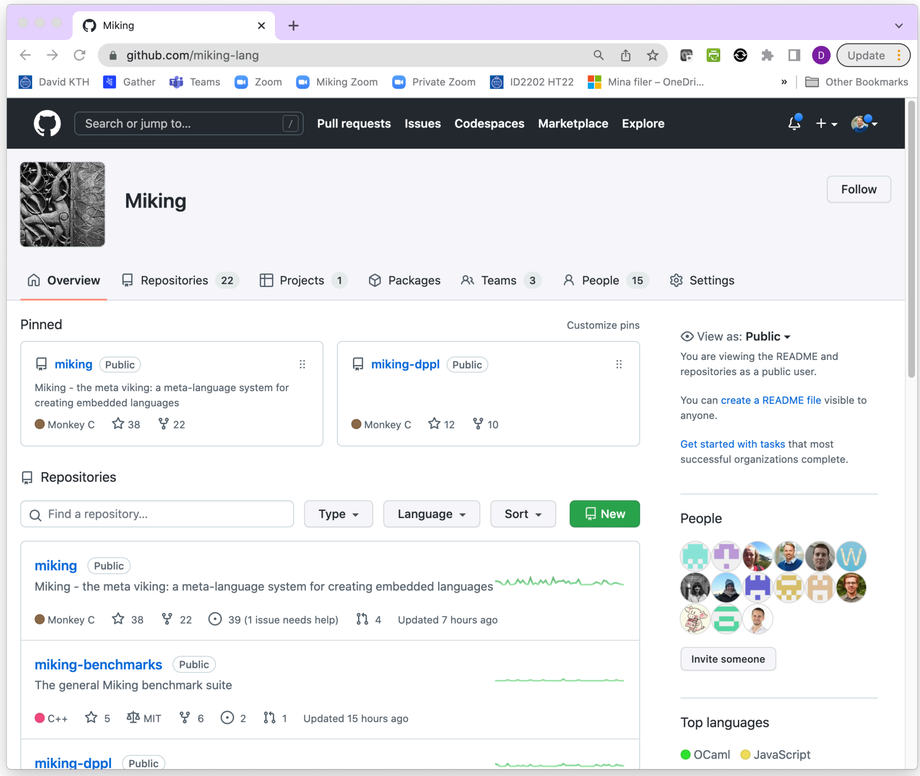



```
def model2 =
        world -- RevoluteJoint(yhat, q0_1) --
        Bar(1.5 * l, q0_1) --
        RevoluteJoint(yhat, q0_1) -- Bar(l, q0_2) -- f1
```

Open Source - MIT license



www.miking.org



https://github.com/miking-lang

Part I
Research Group

Part II
Workshop Overview

Part III
Overview of the Miking Framework

Getting involved

- Thesis research project
- Extending standard library
- Examples and documentation
- Fixing issues

Thanks for listening!