DHBW Karlsruhe

Towards an Integrated Development tool for GA and a symbolic CGA implementation based on CasADi for application in robotics

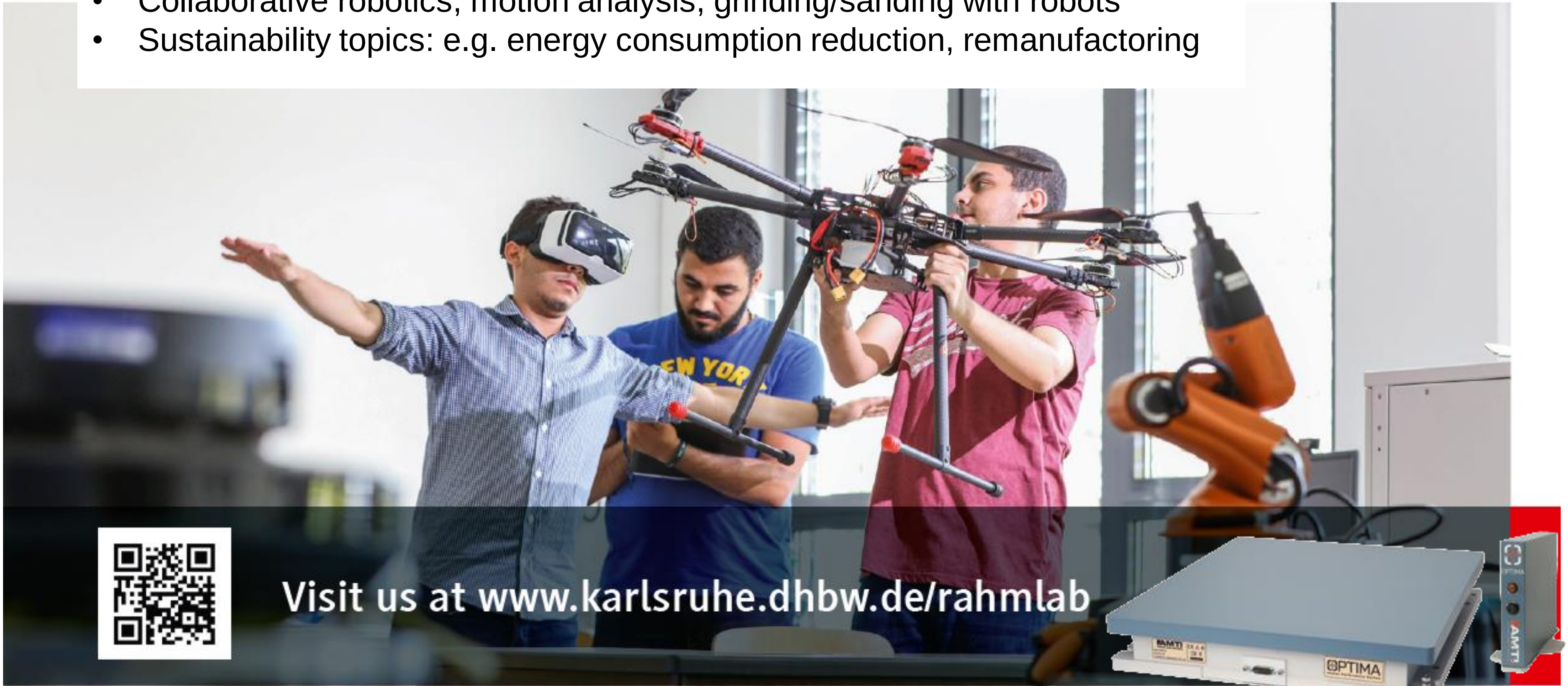
Oliver Rettig, Fabian Hinderer, Marcus Strand

Background

Robot and Human Motion Lab
RAHM-LAB
@ DHBW Karlsruhe

DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

- Collaborative robotics, motion analysis, grinding/sanding with robots
- Sustainability topics: e.g. energy consumption reduction, remanufactoring



Goal: Simplify usage of geometric algebra not only in robotics applications

Application:

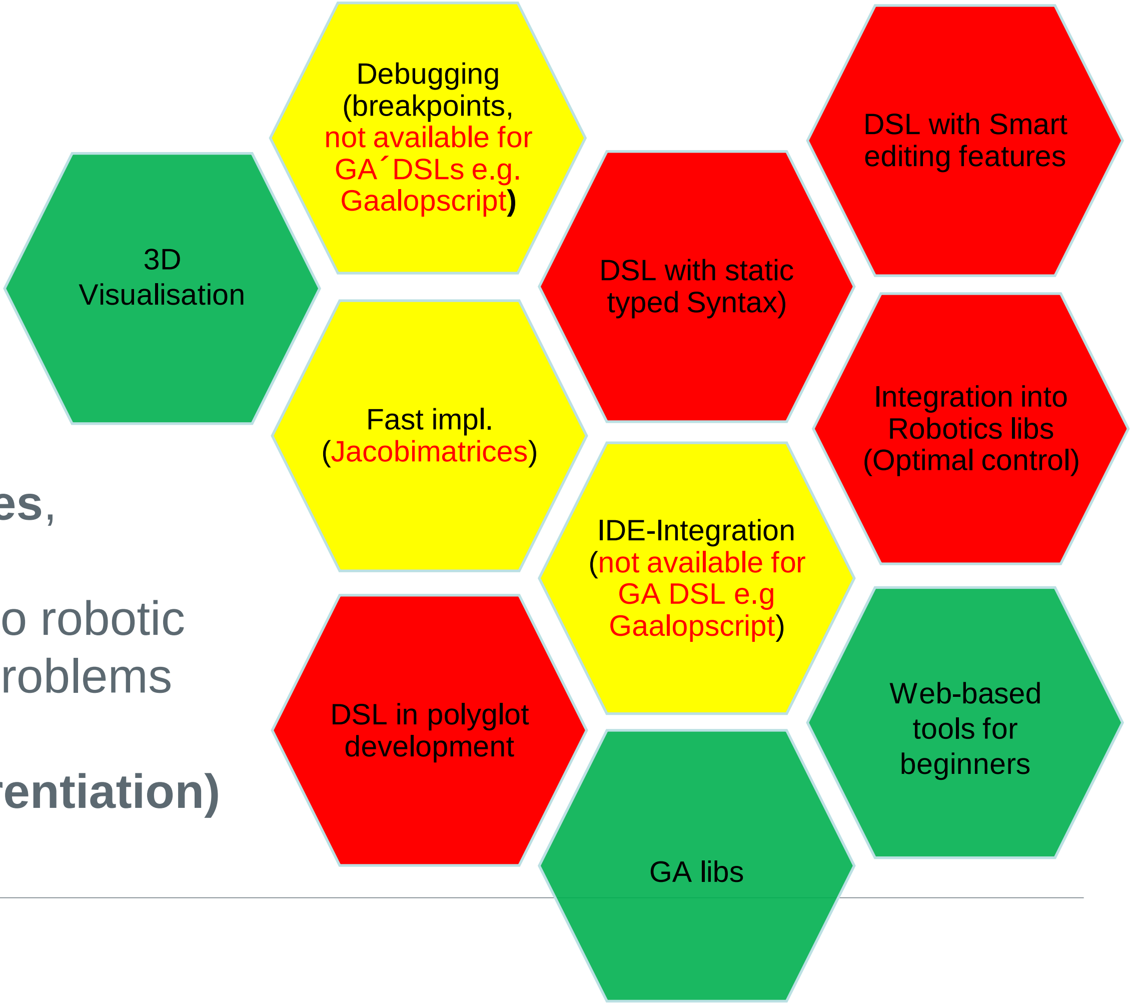
- Robotics: IK, RNEA, …
- Optimal-Control

Available:

- many good GA libs and tools

Missing:

- DSL (statically typed)
- Tooling (**Smart editing features**, **debugging**, 3d visualization)
- Software architecture fitting into robotic tools to solve optimal control problems
- **Jaccobian/Hessian (automatic/algorithmic differentiation)**


3D Visualisation
Debugging (breakpoints, not available for GA´DSLs e.g. Gaalopscript)
DSL with Smart editing features
DSL with static typed Syntax)
Fast impl. (Jacobimatrices)
Integration into Robotics libs (Optimal control)
IDE-Integration (not available for GA DSL e.g Gaalopscript)
DSL in polyglot development
Web-based tools for beginners
GA libs

How to reach the goal: Simplify usage GA in robotics apps

How to get tooling support?

? Integrate a GA lib into a good supported general programming language (Python, C++, Julia…)?

? Create your own DSL from scratch and implement the complete tooling yourself

? Create your own DSL based on modern software-technology-stack for creating programming languages

How to reach good integration of GA lib into robotics appl.?

- fast symbolic expression based GA implementation which allows Jacobian/Hessian calculation by automated differentiation
- to solve nonlinear optimization problems e.g. by Optimal-Control or Model-Predictive-Control

? Create it from the scratch?

? Create it based on an existing modern software-technology-stack

Overview – Choosen open-source technologies

Apache NetBeans

language-server-protocol

Typical 100.000 lines of code for state-of-the-art IDE-support for a new programming language. With usage of GraalVMLSP only 10.000.

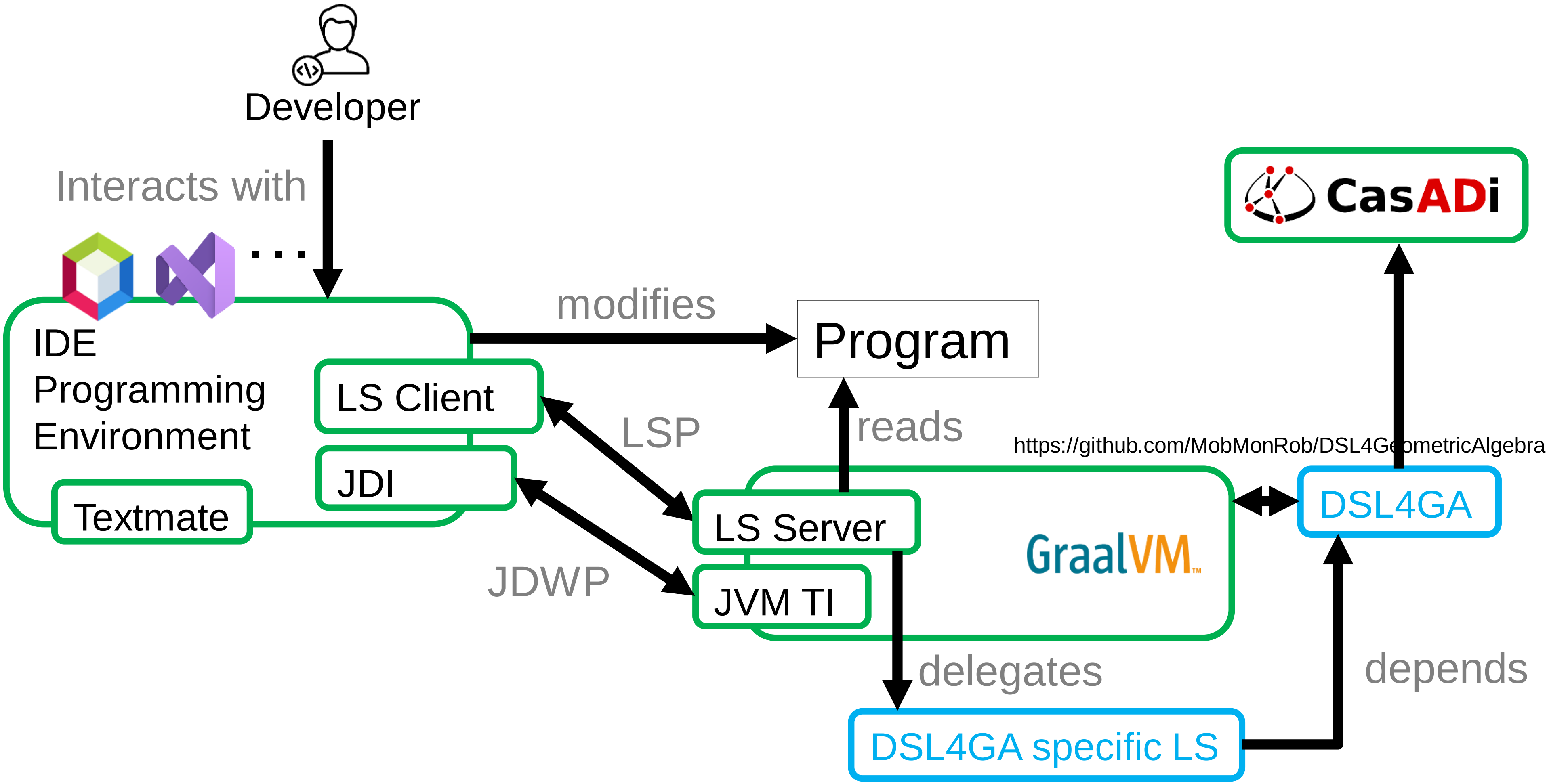
CasADi

GraalVM™

- High-performance, polyglot virtual machine
- JIT- and Ahead-of-time-compiler
- Comes with Truffle, a language implementation framework
- Truffle provides an API for program instrumentation
- based on this API, GraalVM provides various lang-uageagnostic tools such as debuggers and profilers

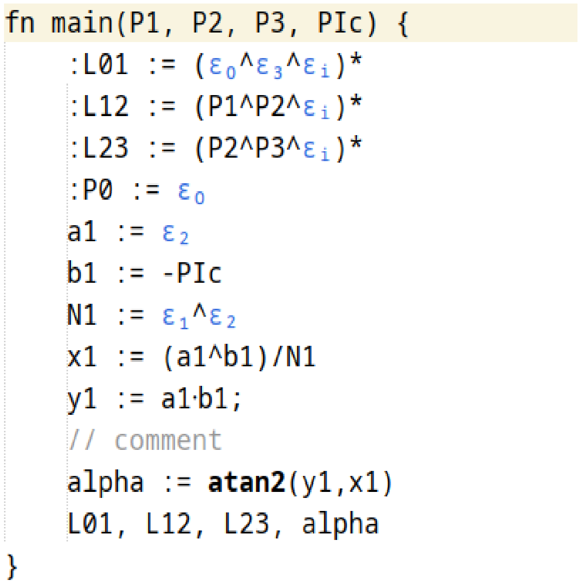
- Symbolic framework implementing forward and reverse mode of algorithmic differentiation on expression graphs to construct gradients, large-and-sparse Jacobians and Hessians
- Evalution in its own VM or exported to standalone c-code.

Overview – software architecture



https://github.com/MobMonRob/DSL4GeometricAlgebra

DSL – status



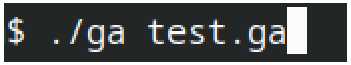
```
fn main(P1, P2, P3, PIc) {
    :L01 := (ε₀^ε₃^εᵢ)*
    :L12 := (P1^P2^εᵢ)*
    :L23 := (P2^P3^εᵢ)*
    :P0 := ε₀
    a1 := ε₂
    b1 := -PIc
    N1 := ε₁^ε₂
    x1 := (a1^b1)/N1
    y1 := a1·b1;
    // comment
    alpha := atan2(y1,x1)
    L01, L12, L23, alpha
}
```

Syntax

- Symbols (unicode representations)
- function defs (multiple result values)
- Build-in functions
- Assignments

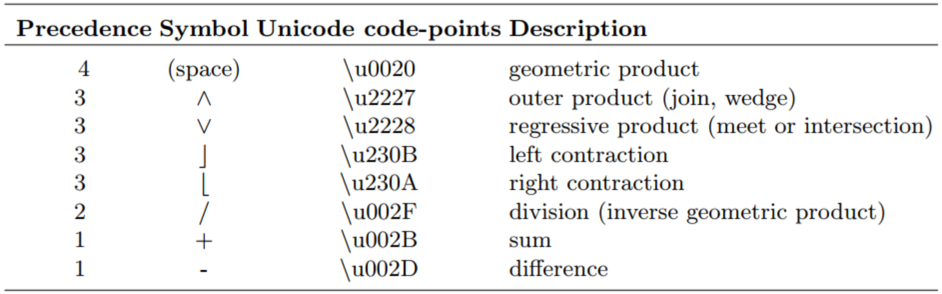
Usage

- Java integration (via annotation)
- Command line execution



```
$ ./ga test.ga
```

- Polyglot inside GraalVM
- Creation of c-code (CasADi)



Precedence	Symbol	Unicode code-points	Description
4	(space)	\u0020	geometric product
3	∧	\u2227	outer product (join, wedge)
3	∨	\u2228	regressive product (meet or intersection)
3	⌋	\u230B	left contraction
3	⌊	\u230A	right contraction
2	/	\u002F	division (inverse geometric product)
1	+	\u002B	sum
1	-	\u002D	difference

Smart editing features

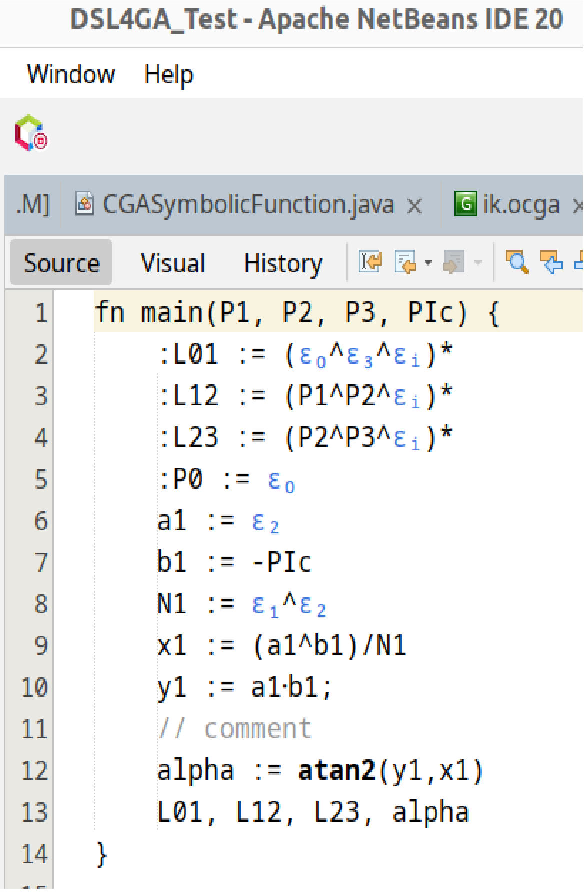
GraalVM‘s Language Server (generic, language-agnostic)

- ✓ Text Document Synchronization
- ✓ Hover Provider
- ✓ Completion Provider
- ✓ Signature Help Provider
- ✓ Code Action Provider (refactoring, quick fixes)
- ✓ CodeLens Provider (links in-between the source code)
- ✓ Execute Command Provider (key-bindings, e.g. command to uncomment a line)

DSL4GA specific Language Server

Textmate based Syntax-Highlighting

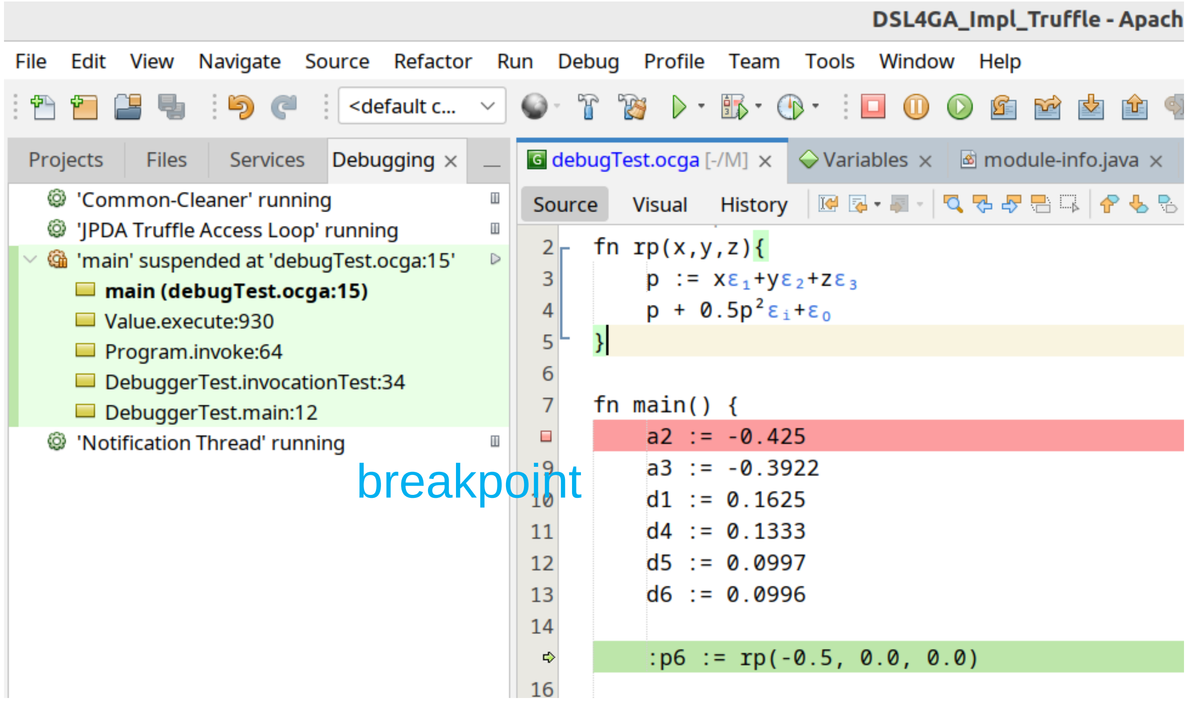
More powerful Syntax-Highlighting based on the anlr



```
fn main(P1, P2, P3, PIc) {
    :L01 := (ε₀^ε₃^εᵢ)*
    :L12 := (P1^P2^εᵢ)*
    :L23 := (P2^P3^εᵢ)*
    :P0 := ε₀
    a1 := ε₂
    b1 := -PIc
    N1 := ε₁^ε₂
    x1 := (a1^b1)/N1
    y1 := a1·b1;
    // comment
    alpha := atan2(y1,x1)
    L01, L12, L23, alpha
}
```

Debugging

- ✓ *tested with the Netbeans IDE*
- ✓ *breakpoints, variables/watches*
- ✓ *3d visualization of the scope („:“ syntax)*
- ✓ *polyglot stacktrace*
- ✓ *Chrome Debugger*
- ✓ *VS but not tested*

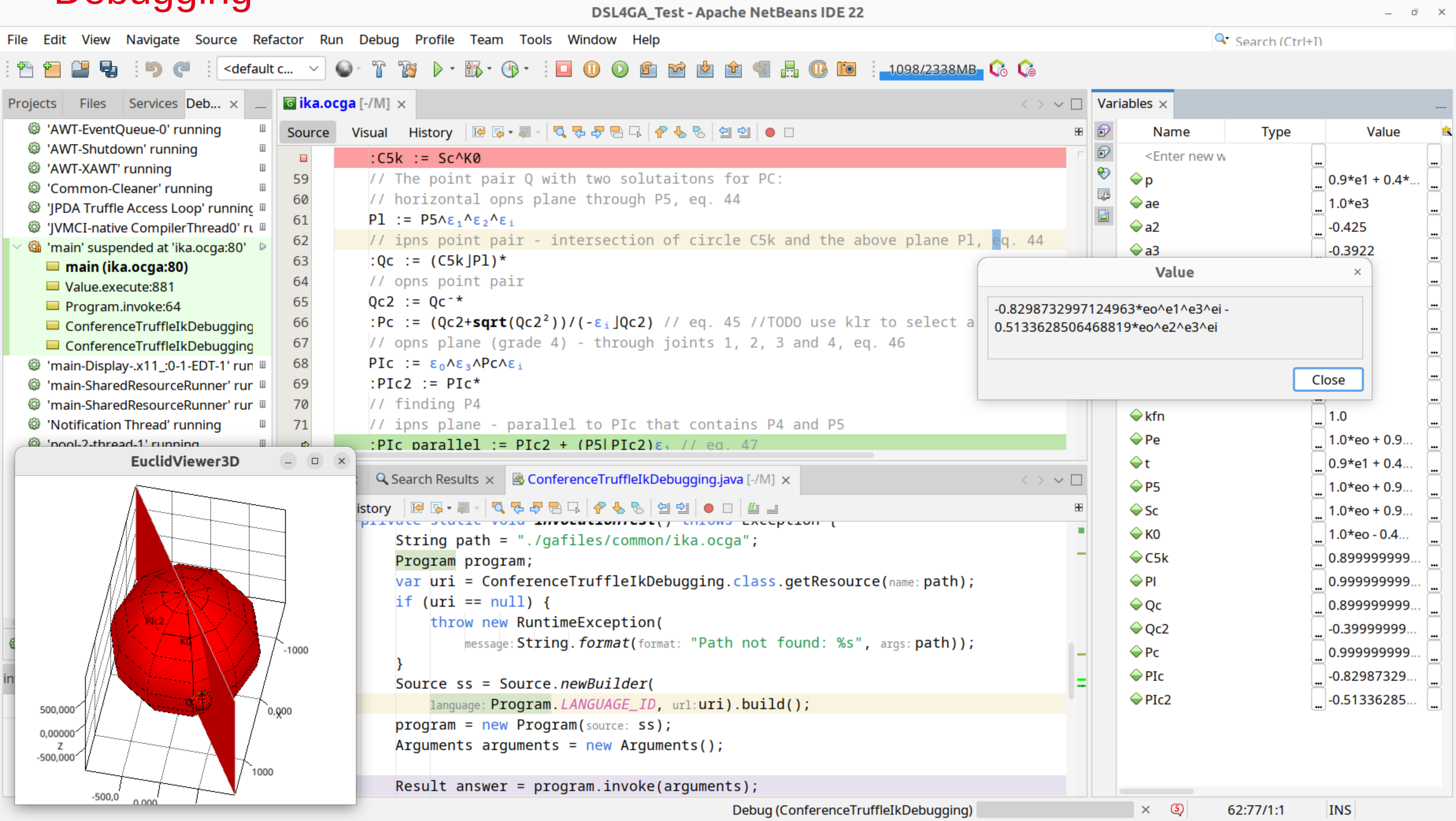


```
'Common-Cleaner' running
'JPDA Truffle Access Loop' running
'main' suspended at 'debugTest.ocga:15'
    main (debugTest.ocga:15)
    Value.execute:930
    Program.invoke:64
    DebuggerTest.invocationTest:34
    DebuggerTest.main:12
'Notification Thread' running
```

```
fn rp(x,y,z){
    p := xε₁+yε₂+zε₃
    p + 0.5p²εᵢ+ε₀
}

fn main() {
    a2 := -0.425
    a3 := -0.3922
    d1 := 0.1625
    d4 := 0.1333
    d5 := 0.0997
    d6 := 0.0996

    :p6 := rp(-0.5, 0.0, 0.0)
```

breakpoint

Debugging



DSL4GA_Test - Apache NetBeans IDE 22
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default c...
1098/2338MB
Projects Files Services Deb...
ika.ocga [-/M]
Source Visual History
Variables
EuclidViewer3D
PIc2
K0
-1000
0,000
1000
500,000
0,00000
Z
-500,000
-500,0
0,000
Search Results
ConferenceTruffleIkDebugging.java [-/M]
istory
Debug (ConferenceTruffleIkDebugging)
62:77/1:1
INS

'AWT-EventQueue-0' running
'AWT-Shutdown' running
'AWT-XAWT' running
'Common-Cleaner' running
'JPDA Truffle Access Loop' running
'JVMCI-native CompilerThread0' ru
'main' suspended at 'ika.ocga:80'
main (ika.ocga:80)
Value.execute:881
Program.invoke:64
ConferenceTruffleIkDebugging
ConferenceTruffleIkDebugging
'main-Display-.x11_:0-1-EDT-1' run
'main-SharedResourceRunner' rur
'main-SharedResourceRunner' rur
'Notification Thread' running
'pool-2-thread-1' running

```
:C5k := Sc^K0
// The point pair Q with two solutaitons for PC:
// horizontal opns plane through P5, eq. 44
Pl := P5^ε₁^ε₂^εᵢ
// ipns point pair - intersection of circle C5k and the above plane Pl, eq. 44
:Qc := (C5k⌋Pl)*
// opns point pair
Qc2 := Qc⁻*
:Pc := (Qc2+sqrt(Qc2²))/(-εᵢ⌋Qc2) // eq. 45 //TODO use klr to select a
// opns plane (grade 4) - through joints 1, 2, 3 and 4, eq. 46
PIc := ε₀^ε₃^Pc^εᵢ
:PIc2 := PIc*
// finding P4
// ipns plane - parallel to PIc that contains P4 and P5
:PIc parallel := PIc2 + (P5⌊PIc2)εᵢ // eq. 47
```

Value

-0.8298732997124963*eo^e1^e3^ei -
0.5133628506468819*eo^e2^e3^ei

Close

Name	Type	Value
<Enter new w		
p		0.9*e1 + 0.4*...
ae		1.0*e3
a2		-0.425
a3		-0.3922
kfn		1.0
Pe		1.0*eo + 0.9...
t		0.9*e1 + 0.4...
P5		1.0*eo + 0.9...
Sc		1.0*eo + 0.9...
K0		1.0*eo - 0.4...
C5k		0.899999999...
PI		0.999999999...
Qc		0.899999999...
Qc2		-0.39999999...
Pc		0.999999999...
PIc		-0.82987329...
PIc2		-0.51336285...

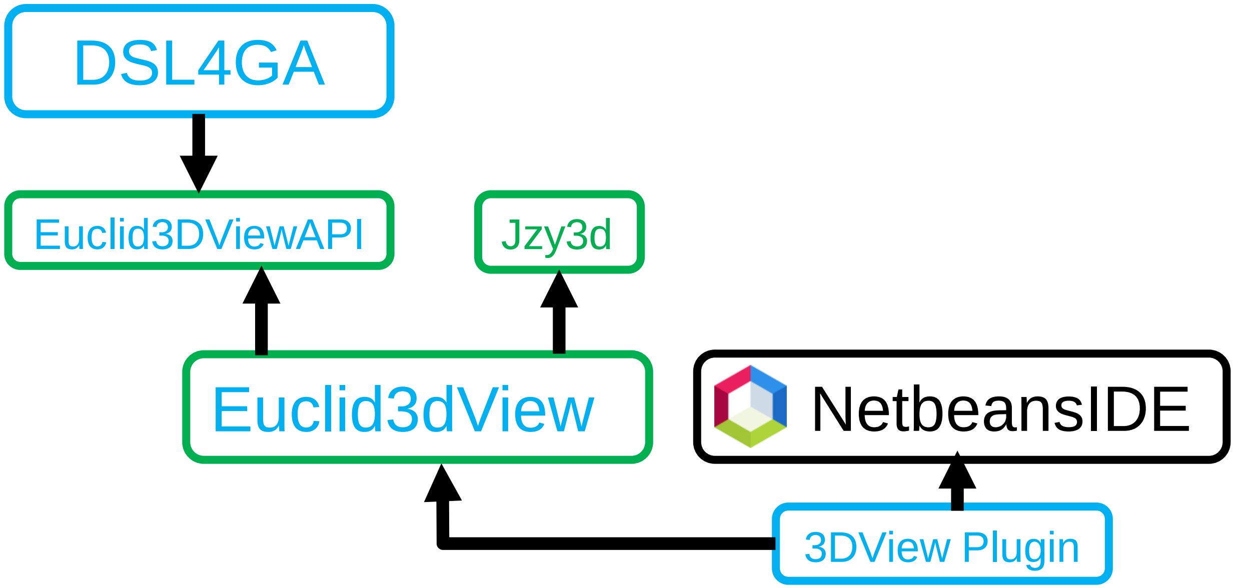
```
private static void invocationTest() throws Exception {
    String path = "./gafiles/common/ika.ocga";
    Program program;
    var uri = ConferenceTruffleIkDebugging.class.getResource(name: path);
    if (uri == null) {
        throw new RuntimeException(
            message: String.format(format: "Path not found: %s", args: path));
    }
    Source ss = Source.newBuilder(
        language: Program.LANGUAGE_ID, url: uri).build();
    program = new Program(source: ss);
    Arguments arguments = new Arguments();

    Result answer = program.invoke(arguments);
```

3D Visualisation

- API to plugin visualizers
- Default: Euclid3dView
  - ✓ Plugin to integrate in the Netbeans IDE
  - ✓ can be used with every IDE
  - – too slow for animations, no scene graph
  - ✓ Visualize robots
  - ✓ and skeletons

? Ganja.js

○ Webots



DSL4GA

Euclid3DViewAPI

Jzy3d

Euclid3dView

NetbeansIDE

3DView Plugin

CasADi based GA implementations

Truffle based AST implementation

- The magic of out-of-the-box tooling support needs the AST is based on Truffle-API
- All program statements (representing GA expressions) are immediately executed by CasADi, if the running programm reaches them
- The CasADi-Wrapper functions (or default Java objects) are invoked a lot

➢ Slow execution, fast compilation

Fast AST implementation

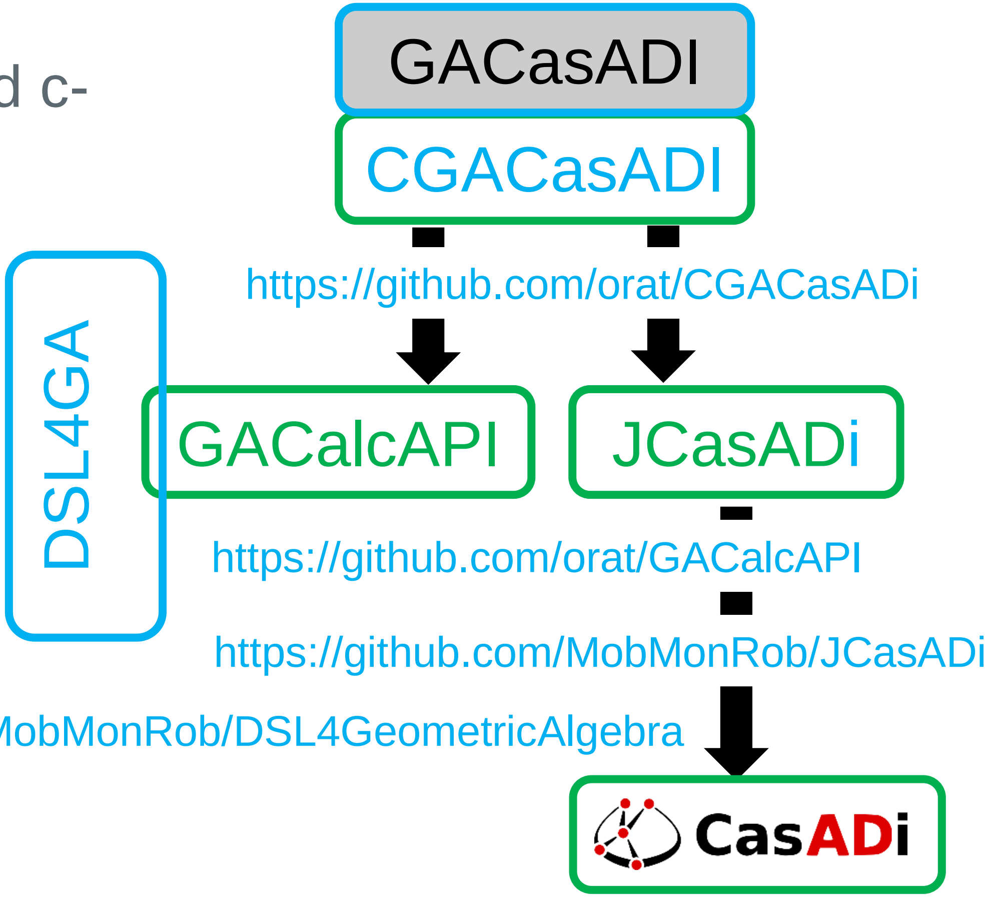
- creates a CasADi AST representation before the program is executed
- automatic resuse parts of the AST – optimization
- Automatic unrolling of loops in into matrices is possible
- CasADi can create automatically optimized c-Code
- code generation with parallelisation is available

➢ Fast execution, long compile time

CasADi based GA implementations

- CasADI AST is created via JCasADi
- CasADI can export optimized c-code without dependences

✓ JCasADi
✓ GACalcAPI
✓ CGACasADi
○ Generic GA impl.
○ Gaalop
? Ganja.js



GACasADI

CGACasADI

https://github.com/orat/CGACasADi

DSL4GA

GACalcAPI

JCasADi

https://github.com/orat/GACalcAPI

https://github.com/MobMonRob/JCasADi

https://github.com/MobMonRob/DSL4GeometricAlgebra

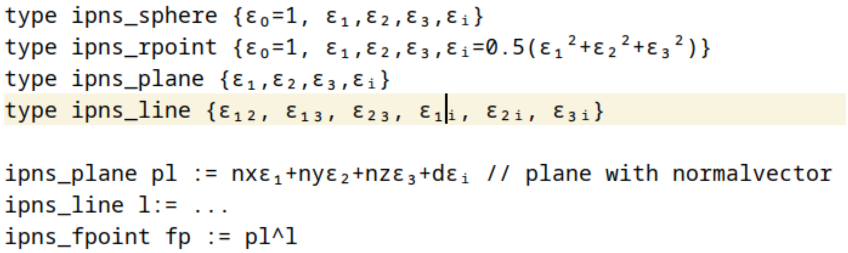
CasADi

Conclusion

- ✓ The feasibility of creating a programming tool chain with a DSL for GA, based on **Truffle/GraalVM** is shown
- ✓ Base functionality (debugging, syntax-highlighing, …) cauld be implemented with less lines of code
- ✓ The build-in GraalVM-LS brings smart editing features out-of-the-box
- – Truffle-based (CasADi) implementation of the AST seems not to be fast enough for our robotics application
- • A symbolic based implementation based on the CasADi AST is fast and support **Jacobians** and **Hessians** out of the box
- ○ It is ambigous how to handle best further hand-in-hand development of truffle-based and fast GA implementation

DSL – extention of the syntax

- with respect to easy creation a fast CasADI AST representation
- following **DOP** (data orientated) instead of OOP pattern: 1. model the data, 2. data is immutable, 3. validate at the boundary, 4. make illegal states unrepresentable
- **keyword to define GA models**
- (static) multivector **subtypes**
- multidim. arrays of multivectors
- records (named tuples)
- if statements, for-loops
- polyglot API (import/export func.)



```
type ipns_sphere {ε₀=1, ε₁,ε₂,ε₃,εᵢ}
type ipns_rpoint {ε₀=1, ε₁,ε₂,ε₃,εᵢ=0.5(ε₁²+ε₂²+ε₃²)}
type ipns_plane {ε₁,ε₂,ε₃,εᵢ}
type ipns_line {ε₁₂, ε₁₃, ε₂₃, ε₁ᵢ, ε₂ᵢ, ε₃ᵢ}

ipns_plane pl := nxε₁+nyε₂+nzε₃+dεᵢ // plane with normalvector
ipns_line l:= ...
ipns_fpoint fp := pl^l
```

CasADi based implementations: Next steps

Truffle based AST implementation

- Impl. of GraalVM Polyglott API (ganjs.js 3d Visualisation?)

Fast implementation:

- „Hyperwedge“ impl.
- Loop unrolling by CasADi based parallelization (map)
- Symbolic optimization of math expressions with Maxima (Computer algebra system)
- Precompile CasADi into LLVM bitcode (by GraalVM toolchain)
- …

Next Milestone

- Bundle of all componentes together into a single plugin for the NetbeansIDE
- Completion of some features, testing, bug fixing
- Configuration of Windows-Build (JCasADi)

Discussion

- DSL Syntax extentions?
- Naming? Current idea „Gaazelle“
- Benchmarks? Quantifying runtime speed?
- Comparison with other ga-libs?
- Recommendations for further development?

Thank you for your attention

Oliver Rettig

Oliver.Rettig@dhbw-karlsruhe.de