

# 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



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



Visit us at [www.karlsruhe.dhbw.de/rahmlab](http://www.karlsruhe.dhbw.de/rahmlab)



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

## Application:

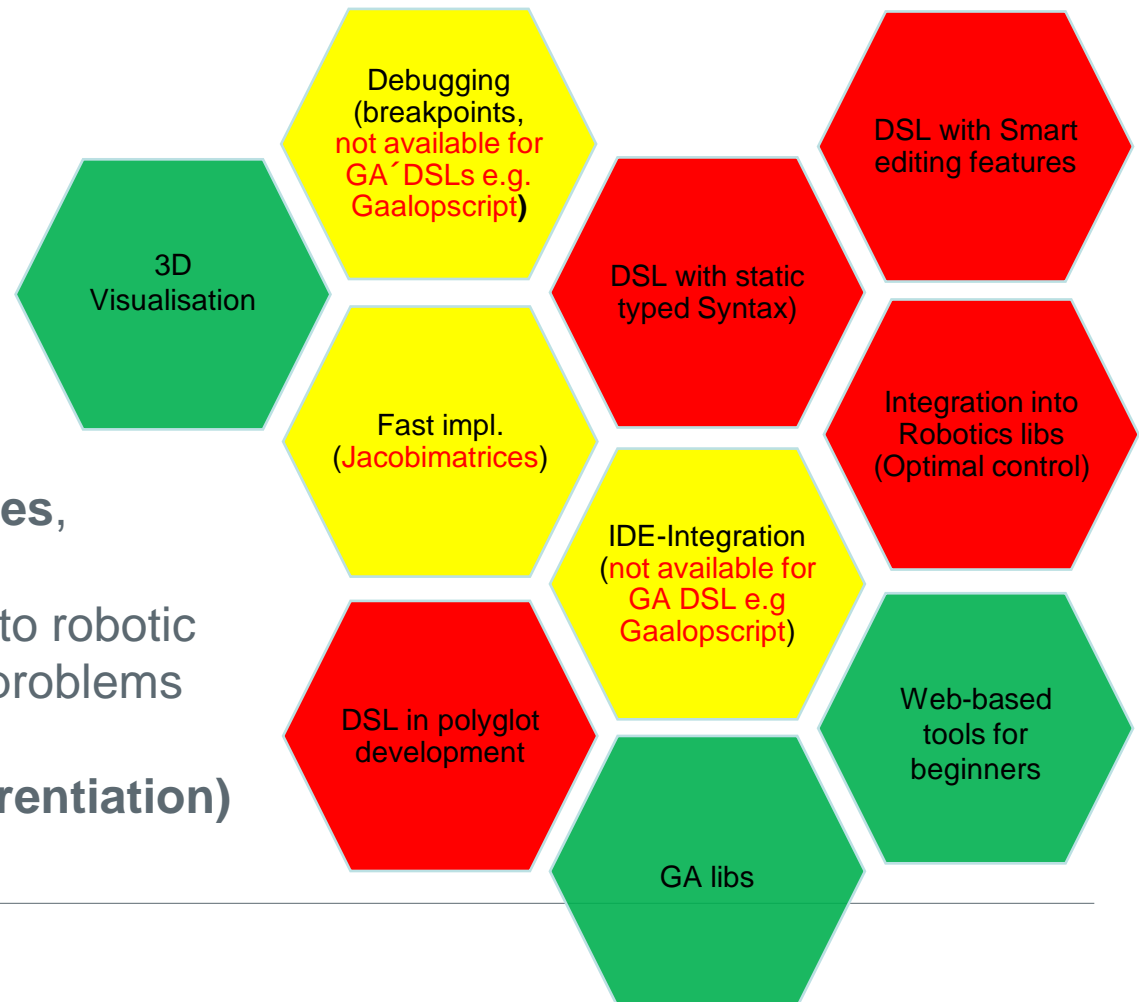
- Robotics: IK, RNEA, ...
- Optimal-Control

## 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)**





## 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 – Chosen 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.**

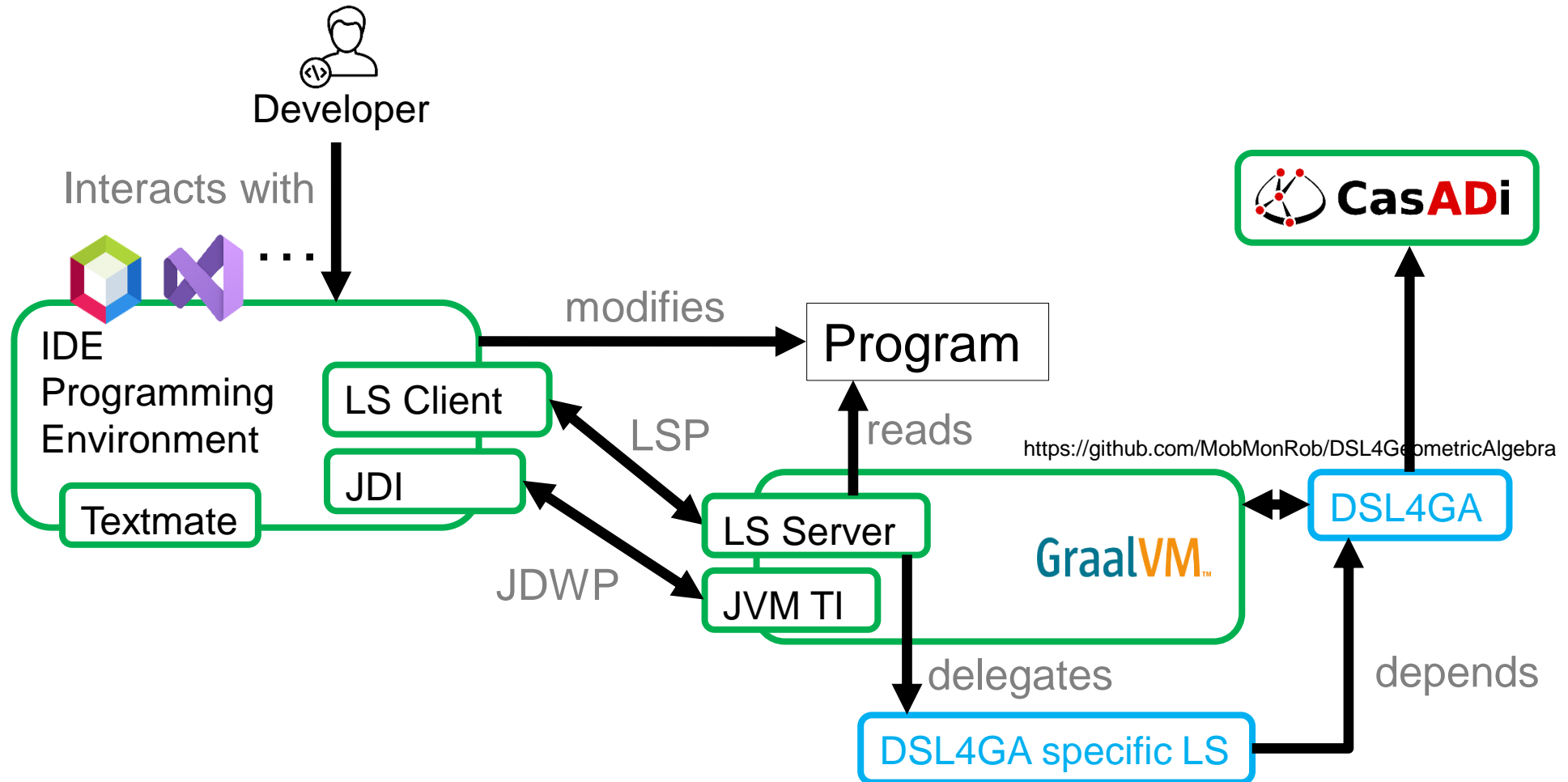


CasADi

# 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 languageagnostic 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
  - Evaluation in its own VM or exported to standalone c-code.
-

## Overview – software architecture





## DSL – status

### 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)

```
fn main(P1, P2, P3, PIC) {
  :L01 := ( $\epsilon_0^{\epsilon_3^{\epsilon_i}}$ )*
  :L12 := (P1^P2^ $\epsilon_i$ )*
  :L23 := (P2^P3^ $\epsilon_i$ )*
  :P0 :=  $\epsilon_0$ 
  a1 :=  $\epsilon_2$ 
  b1 := -PIC
  N1 :=  $\epsilon_1^{\epsilon_2}$ 
  x1 := (a1^b1)/N1
  y1 := a1·b1;
  // comment
  alpha := atan2(y1,x1)
  L01, L12, L23, alpha
}
```

Precedence	Symbol	Unicode	code-points	Description
4	(space)	\u0020		geometric product
3	^	\u2227		outer product (join, wedge)
3	v	\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

The screenshot shows the Apache NetBeans IDE 20 interface. The title bar reads "DSL4GA\_Test - Apache NetBeans IDE 20". The window title is ".M] CGASymbolicFunction.java x ik.ocga x". The editor shows a function definition in a DSL-like language:

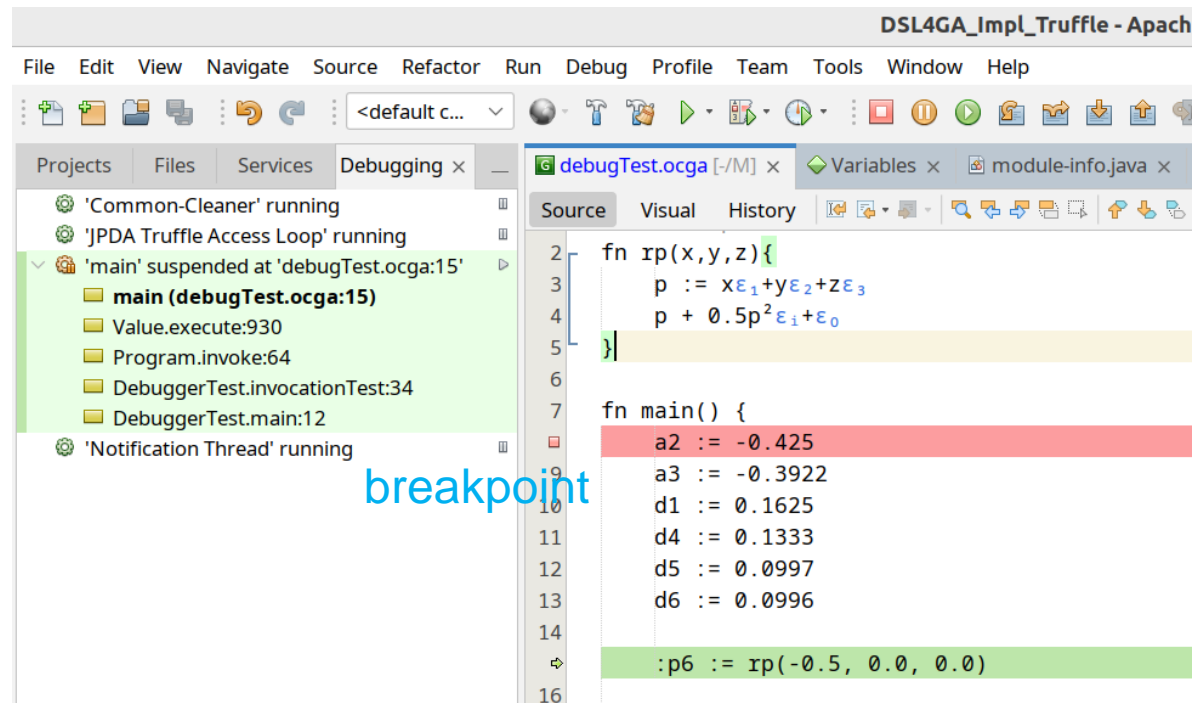
```
1 fn main(P1, P2, P3, PIc) {
2     :L01 := ( $\epsilon_0^{\wedge} \epsilon_3^{\wedge} \epsilon_i$ ) *
3     :L12 := (P1^P2^ $\epsilon_i$ ) *
4     :L23 := (P2^P3^ $\epsilon_i$ ) *
5     :P0 :=  $\epsilon_0$ 
6     a1 :=  $\epsilon_2$ 
7     b1 := -PIc
8     N1 :=  $\epsilon_1^{\wedge} \epsilon_2$ 
9     x1 := (a1^b1)/N1
10    y1 := a1·b1;
11    // comment
12    alpha := atan2(y1,x1)
13    L01, L12, L23, alpha
14 }
```





## Debugging

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



# Debugging

DSL4GA\_Test - Apache NetBeans IDE 22

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

1098/2338MB

ika.ocga [-/M] x

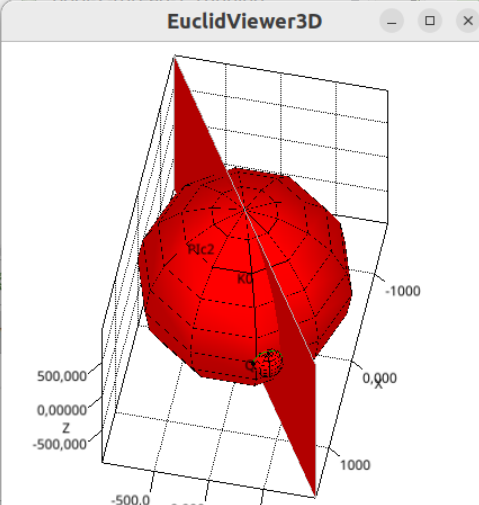
```

:CSk := Sc^K0
// The point pair Q with two solutaitons for PC:
// horizontal opns plane through P5, eq. 44
P1 := P5^ε1^ε2^ε1
// ipns point pair - intersection of circle C5k and the above plane P1, eq. 44
:Qc := (C5kJP1)*
// opns point pair
Qc2 := Qc-*
:Pc := (Qc2+sqrt(Qc2^2))/(-ε1JQc2) // eq. 45 //TODO use klr to select a
// opns plane (grade 4) - through joints 1, 2, 3 and 4, eq. 46
PIC := ε0^ε3^APc^ε1
:PIC2 := PIC*
// finding P4
// ipns plane - parallel to PIC that contains P4 and P5
:PIC parallel := PIC2 + (P5|PIC2)^ε1 // eq. 47
    
```

Variables x

Name	Type	Value
<Enter new w		
p		0.9*e1 + 0.4*...
ae		1.0*e3
a2		-0.425
a3		-0.3922
Value		
-0.8298732997124963*eo^e1^e3^ei - 0.5133628506468819*eo^e2^e3^ei		
Close		
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		-0.399999999...
Pc		0.999999999...
PIC		-0.82987329...
PIC2		-0.51336285...

EuclidViewer3D



Search Results x ConferenceTruffleIkDebugging.java [-/M] x

```

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);
    
```

Debug (ConferenceTruffleIkDebugging) x 62:77/1:1 INS

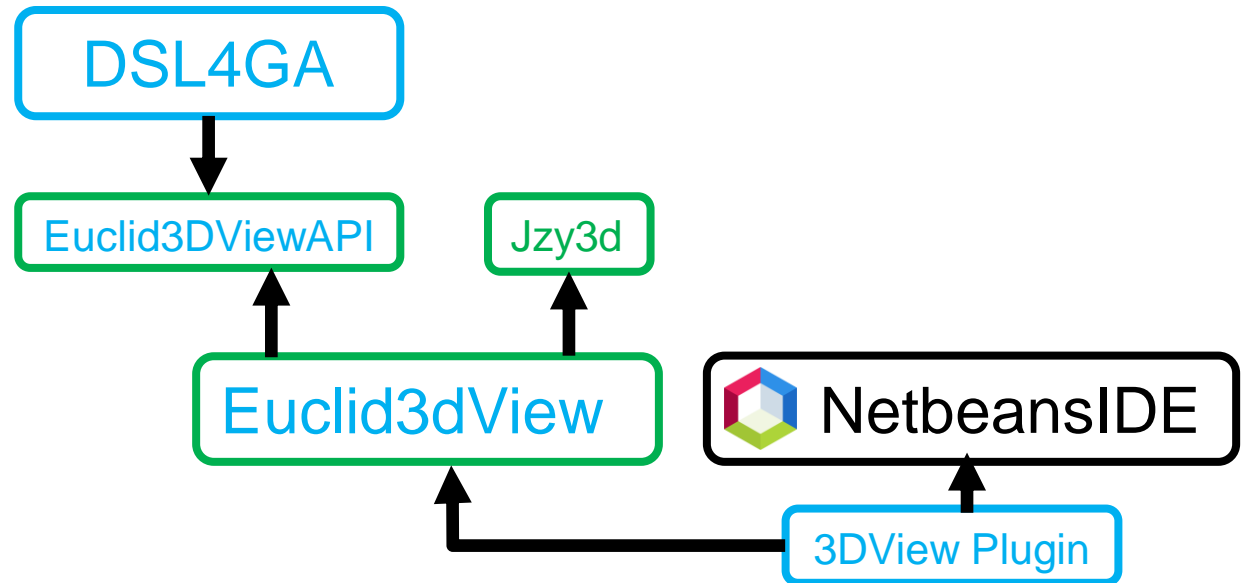


## 3D Visualisation

- 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





## 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 reuse 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**

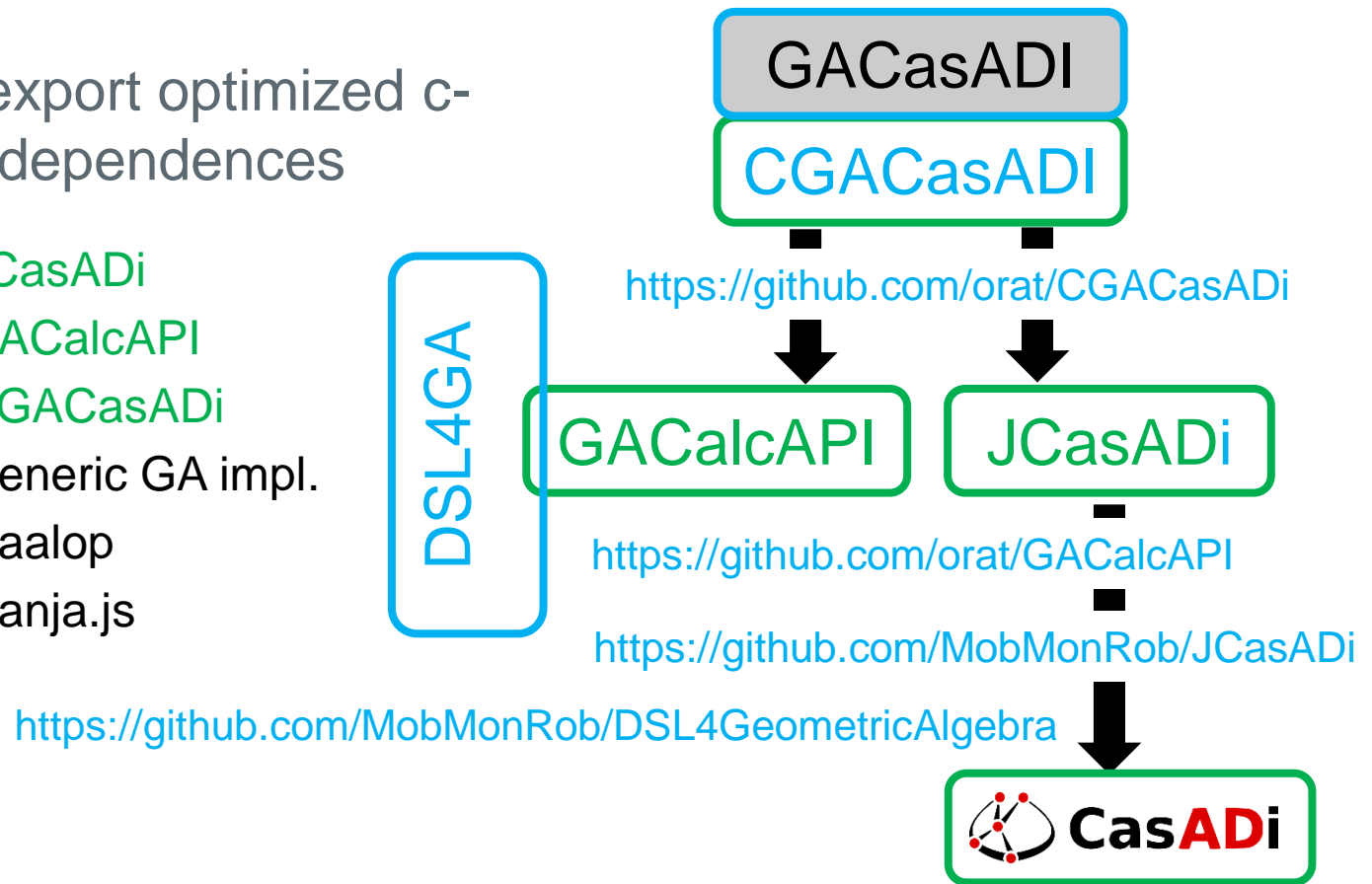
---



## 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





## Conclusion

- ✓ The feasibility of creating a programming tool chain with a DSL for GA, based on **Truffle/GraalVM** is shown
  - ✓ Base functionality (debugging, syntax-highlighting, ...) could 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 ambiguous 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, εii, ε2i, ε3i}
```

```
ipns_plane p1 := nxε1+nyε2+nzε3+dεi // plane with normalvector  
ipns_line l:= ...  
ipns_fpoint fp := p1^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 components 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

---