Optimaiden implementation development

Saulius Gražulis

Vilnius, OPTIMADE workshop, 2025

Vilnius University, Life Sciences Centre, Institute of Biotechnology
Vilnius University, Faculty of Mathematics and Informatics, Institute of Informatics

Id: slides.tex 3127 2025-09-17 09:24:13Z saulius September 17, 2025

Desiderata

- Code works across language and library version changes (~~Python~~)
- Hight abstraction level, memory safe (~~C~~, C++ (?))
- Describes compiler-checked library interfaces (~~Perl~~, ~~PhP~~)

(Amiard et al. 2022)

Desiderata

- Code works across language and library version changes (~~Python~~)
- Hight abstraction level, memory safe (~~C~~, C++ (?))
- Describes compiler-checked library interfaces (~~Perl~~, ~~PhP~~)

(Amiard et al. 2022)

Code example

Space group symbol decoder

```
   C 2y (x-y, x+y, z),   C 2y (1/2*a - 1/2*b, 1/2*a + 1/2*b, c)
```

```
Get_Hall_Symbol_Inversions (Symbol, Pos, N_Inversions);
Get_Hall_Symbol_Centerings (Symbol, Pos, Centering, N_Centering);

for Axis_Number in 1..4 loop
   Get_Hall_Symbol_Rotation (Symbol, Pos, Symmetry_Operators,
                             N_Symmetry_Operators,
                             Preceeding_Axis_Direction,
                             Preceeding_Axis_Order, Axis_Number);
end loop;

Get_Change_Of_Basis (Symbol, Pos, Change_Of_Basis);

Apply_Change_Of_Basis
  (
   Symmetry_Operators, N_Symmetry_Operators,
   Centering, N_Centering,
   Inversions, N_Inversions,
   Change_Of_Basis
  );

Build_Group (Symmetry_Operators, N_Symmetry_Operators);
```

(Hall 1981a)
(Hall 1981b)

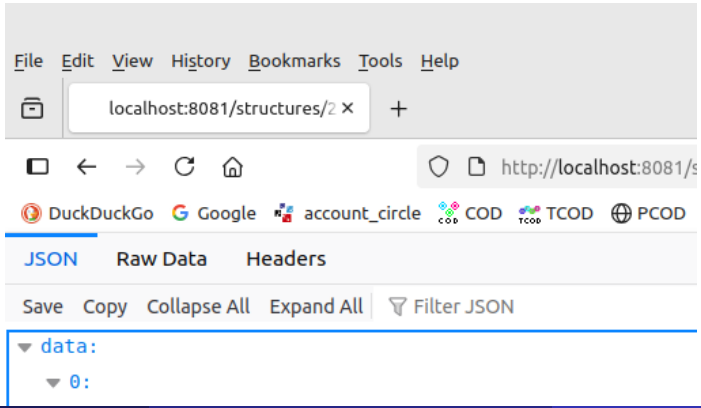
Code example

The interface

```
with Symmetry_Operations; use Symmetry_Operations;

package Hall_Symbol_Parser is

   Debug_Print_Matrices : Boolean := False;

   function Decode_Hall_Symbol (Symbol : in String)
                                return Symmetry_Operator_Array;

end Hall_Symbol_Parser;
```

```
saulius@starta optimaiden/ $ make -W src/optimaiden.adb
alr build
...
 Build finished successfully in 1.10 seconds.
```

```
saulius@starta optimaiden/ $ make -W src/optimaiden.adb
alr build
...
 Build finished successfully in 1.10 seconds.
```



Filter language

Tools at hand:

- AYacc :)
- AFelx :)
- Grammatiker (OPTIMADE EBNF $\rightarrow$ AYacc)
- GNAT (of course :)

Filter grammar

```
ebnf2ayacc Filter.ebnf > Filter.y
```

```
(* BEGIN EBNF GRAMMAR Filter *)
(* The top-level 'filter' rule: *)

Filter = [Spaces], Expression ;

(* Values *)

OrderedConstant = String | Number ;
UnorderedConstant = ( TRUE | FALSE ) ;

Value = ( UnorderedConstant | OrderedValue ) ;
```

```
%token ONLY_TOKEN
%token ANY_TOKEN

%token TRUE_TOKEN
%token FALSE_TOKEN

%token IDENTIFIER_TOKEN
%token NUMBER_TOKEN

%%

Filter : optional__Spaces Expression
;
OrderedConstant : String | Number
;
UnorderedConstant : grouped__TRUEs
;
Value : grouped__UnorderedConstants
;
```

Ada compilation

```
for I in 1 .. Argument_Count loop
    YYInput_Definition.Start_Parsing (Argument (I));
    Put_Line ("Parsing string: """ & Buffer & """");
    YYParse;
end loop;
```

```
saulius@starta optimaiden/ $ ./bin/parse_filter 'nelements > 3 AND _cod_formula ST
Parsing string: "nelements > 3 AND _cod_formula STARTS WITH "C6""
>>> Token: IDENTIFIER_TOKEN, "nelements"
>>> Token: NUMBER_TOKEN, "3" ( 3.00000E+00)
>>> Getting number:  3.00000E+00
>>> Token: IDENTIFIER_TOKEN, "_cod_formula"
```

Future plans

- Connect to SQL (Ada libraries exist – report by Yaroslav Rozdobudko)
- Generate serving code from the formal OPTIMADE spec. (Swagger? Y.R.);
- Use formal property definitions to generate responses (A.M.)
- Split the task into Ada packages :);

Conclusions

- Some aspects of OPTIMADE implemenation (e.g. filter grammar) can be generated automatically from the *formal* OPTIMADE spec. in any language;
- Ada gives just enought tools to implement OPTIMADE within reasonable time and with reasonable effort.
- Ada allows us to produce a protable and maintainable implementation of OPTIMADE;

Acknowledgements

VU LSC IBT (KICIS)

Andrius Merkys
Antanas Vaitkus
Algirdas Grybauskas

VU MIF II (FMG)

Linas Laibinis
Karolis Petrauskas
Irus Grinis
Haroldas Giedra

QM community

Linas Vilčiauskas
Vytautas Žalandauskas
Lukas Razinkovas
Nicola Marzari
Giovanni Pizzi
Matthew Evans
Rickard Armiento

COD Advisory board

Daniel Chateigner
Robert T. Downs
Werner Kaminsky
Armel Le Bail
Luca Lutterotti
Peter Moeck
Peter Murray-Rust
Miguel Quirós

Funding:

Data Center for Machine Learning and Quantum Computing in Natural and Biomedical Sciences, Research Council of Lithuania Project No.: S-A-UEI-23-11

CECAM and MARVEL are kindly acknowledged for the financial support of the OPTIMADE workshop.

Useful links

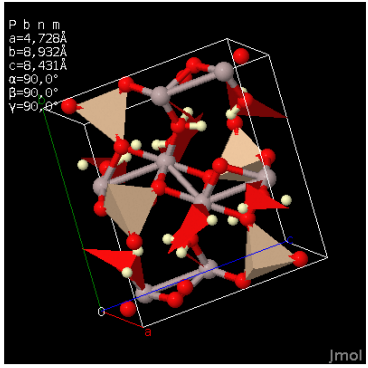
Ada code and Ada books

Code: https://github.com/sauliusg/

Ada books: https://bookauthority.org/books/best-ada-books

Open-access Ada books from AdaCore: https://learn.adacore.com/

Thank you!



Coordinates	2207377.cif
Original IUCr paper	HTML

http://en.wikipedia.org/wiki/Topaz

http://www.crystallography.net/2207377.html

References I

Amiard, Raphaël et al. (Apr. 2022). *Learning Ada*. Ed. by Richard Kenner. URL: https://learn.adacore.com/pdf_books/learning-ada.pdf WEB: https://learn.adacore.com/. AdaCore. URL: https://learn.adacore.com/pdf_books/learning-ada.pdf.

Gražulis, Saulius et al. (2009). "Crystallography Open Database – an open-access collection of crystal structures". In: *Journal of Applied Crystallography* 42, pp. 726–729. DOI: 10.1107/S0021889809016690. URL: http://dx.doi.org/10.1107/S0021889809016690.

Hall, S. R. (July 1981a). "Space-group notation with an explicit origin". In: *Acta Crystallographica Section A* 37.4, pp. 517–525. DOI: 10.1107/s0567739481001228.

— (Nov. 1981b). "Space-group notation with an explicit origin; erratum". In: *Acta Crystallographica Section A* 37.6, pp. 921–921. DOI: 10.1107/s0567739481001976.

Spare slides

Just in case someone asks...

Ada CIF parser

Multi language programming

`cod-tools` CIF parser: svn://www.crystallography.net/cod-tools

```
procedure Parse_Cif_From_File_With_Error_Code (Filename : char_array;
                                               CO : Cif_Option_T)
      with Import => True,
      Convention => C,
      External_Name => "parse_cif_from_file_with_error_code";


task Cif_Parser_Task is
   entry Begin_Parsing (Cif_Filename : Unbounded_String;
                        Cif_Options : Cif_Option_T);
end Cif_Parser_Task;
```

Ada popularity

Growing ?!

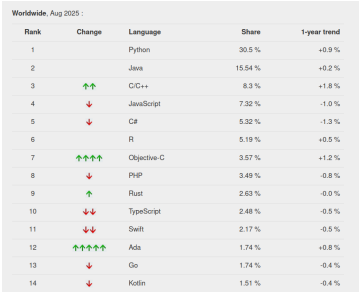
(All URLs accessed 2025-08-21)

“Why is 40-year-old programming language Ada hot again?”
https://www.developer-tech.com/news/why-is-40-year-old-programming-language-ada-hot-again/

“2025: ‘Golden Oldie’ Ada Hits Popularity Milestone”
https://www.techrepublic.com/article/news-tiobe-analysis-july-2025/

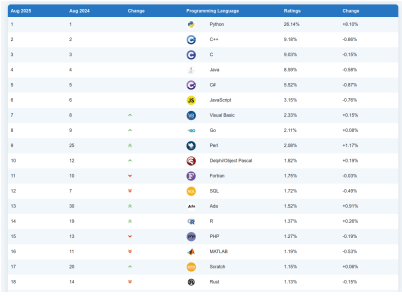
https://pypl.github.io/PYPL.html

https://www.tiobe.com/tiobe-index/



Worldwide, Aug 2025 :

Rank	Change	Language	Share	1-year trend
1		Python	30.5 %	+0.9 %
2		Java	15.54 %	+0.2 %
3	↑↑	C/C++	8.3 %	+1.8 %
4	↓	JavaScript	7.32 %	-1.0 %
5	↓	C#	5.32 %	-1.3 %
6		R	5.19 %	+0.5 %
7	↑↑↑↑	Objective-C	3.57 %	+1.2 %
8	↓	PHP	3.49 %	-0.8 %
9	↑	Rust	2.63 %	-0.0 %
10	↓↓	TypeScript	2.48 %	-0.5 %
11	↓↓	Swift	2.17 %	-0.5 %
12	↑↑↑↑↑	Ada	1.74 %	+0.8 %
13	↓	Go	1.74 %	-0.4 %
14	↓	Kotlin	1.51 %	-0.4 %



Aug 2025	Aug 2024	Change	Programming Language	Ratings	Change
1	1		Python	26.14%	+8.13%
2	2		C++	9.18%	-0.88%
3	3		C	9.03%	-0.15%
4	4		Java	8.89%	-0.68%
5	5		C#	5.52%	-0.87%
6	6		JavaScript	3.15%	-0.76%
7	8	↑	Visual Basic	2.33%	+0.15%
8	9	↑	Go	2.11%	+0.08%
9	25	⇑	Perl	2.08%	+1.17%
10	12	↑	Delphi/Object Pascal	1.82%	+0.19%
11	10	↓	Fortran	1.79%	-0.03%
12	7	⇓	SQL	1.72%	-0.49%
13	38	⇑	Ada	1.52%	+0.91%
14	19	⇑	R	1.37%	+0.28%
15	13	↓	PHP	1.27%	-0.19%
16	11	⇓	MATLAB	1.19%	-0.53%
17	28	↑	Scratch	1.15%	+0.06%
18	14	⇓	Rust	1.13%	-0.15%

Example: dihedral angle calculation

The assignment

```
Write a Perl*) program that calculates torsion angles for DNA and RNA
macromolecules. Compute the following angles: alpha, beta.

*) Note: Ada programs are acceptable; in that case all native Perl
   features mentioned in this specification SHOULD be replaced by the
   corresponding Ada features.

NB: for this particular assignment, you MAY NOT use external libraries
to read PDB files or calculate dihedral angles; these functions MUST
be implemented in the program.
```

NOTE: Students could chose a set of atoms for their individual assignment

Full assignment text: https://tinyurl.com/yp24aad4

Example: dihedral angle calculation

The vector algebra code

```
type Vector_3D is array (1 .. 3) of Long_Float;

function "-" (V, W : Vector_3D) return Vector_3D is
begin
   return (V(1) - W(1),  V(2) - W(2), V(3) - W(3));
end;
```

Example given to the students

```
type Vector_3D is record
   X, Y, Z : Float;
end record;

function "-" (V1, V2 : Vector_3D) return Vector_3D is
begin
    return (V1.X - V2.X, V1.Y - V2.Y, V1.Z - V2.Z);
end;
```

Version designed by the students (Martynas Mažuolis)

Example: dihedral angle calculation

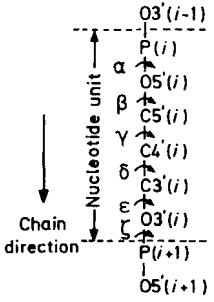
Angle definition and calculation

```
Function Dihedral_angle (Atoms: Chemical_Atom_Set) return float is
        Vector1 : Vector_3D := to_vector(Atoms.Atom_Array(1)) - to_vec..
        Vector2 : Vector_3D := to_vector(Atoms.Atom_Array(2)) - to_vec..
        Vector3 : Vector_3D := to_vector(Atoms.Atom_Array(3)) - to_vec..
        Angle1 : Bond_Angle := Angle(Vector1, Vector2);
        Angle2 : Bond_Angle := Angle(Vector2, Vector3);
        Normal1 : Vector_3D := Cross(Vector1, Vector2);
        Normal2 : Vector_3D := Cross(Vector2, Vector3);
        NX: Vector_3D := Cross(Normal1, Normal2);
        Sign : float := (if Vector2*NX >= 0.0 then 1.0 else -1.0);
        Dihedral : Float := Angle(Normal1, Normal2)*Sign;

        begin
        pragma Assert(Atoms.N>=4);
    return Dihedral;

        end;
```

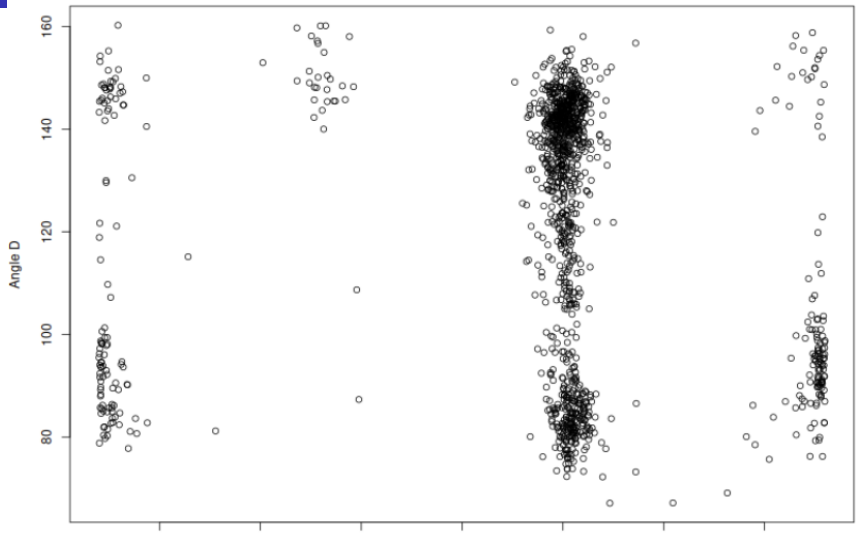
Version designed by the students (Viktorija Jugai)

Example: dihedral angle calculation

Results on the whole PDB



(IUPAC-IUB, **IIJCBN1983**)



Result computed by Aušrinė Zeleckytė