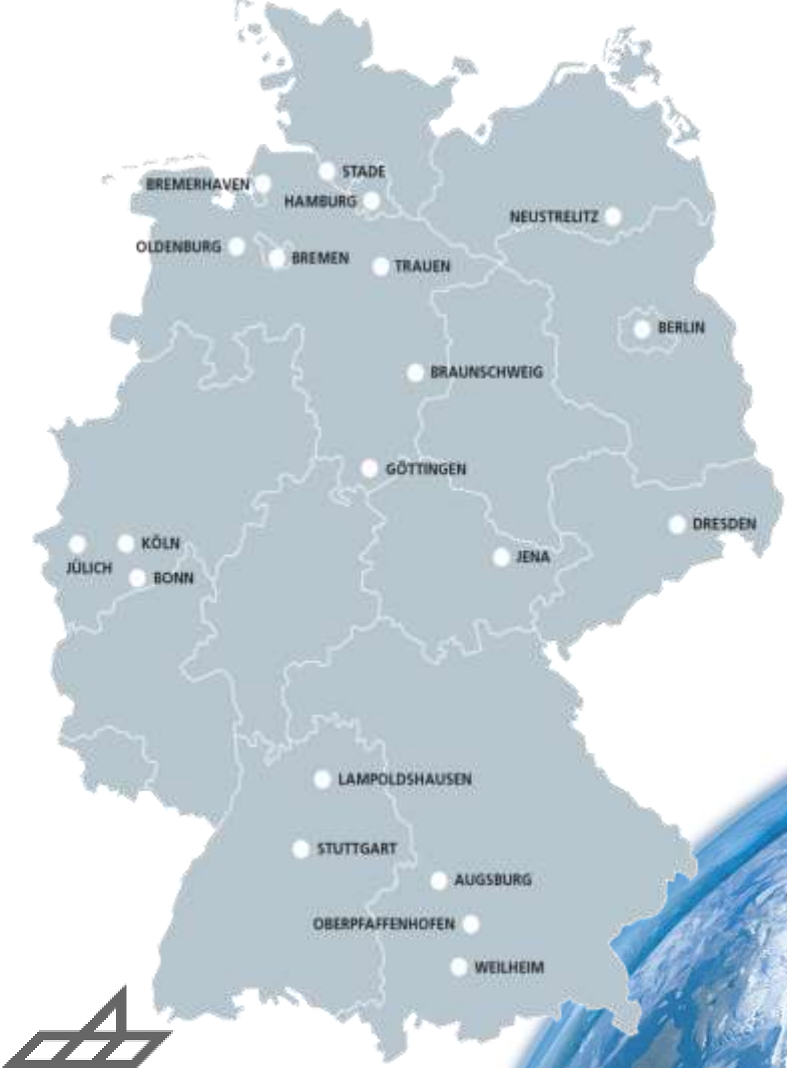
F2x - Automated FORTRAN wrapping without limits

EuroSciPy, 30./31.08.2018, Trento, Italy

Michael Meinel (@led02)

Simulation and Software Technology, Berlin

German Aerospace Center
Deutsches Zentrum für Luft- und Raumfahrt (DLR)



Approx. 8,000 Employees

39 Institutes and Facilities

20 Locations

HQ: Cologne, Germany

D-CMET

Overview

- Motivation for a new Fortran wrapper
- Implementation Details
  - Parser
  - Templates
- Experiments and Usage
- Benchmarks
- Outlook

Project „BACARDI“
Debris in Space

- Cooperation with German Space Operation Center (GSOC)

- Incresing amount of debris especially in LEO
    - Dangerous for exiting missions
    - Space problem for upcoming missions

➔ Develop catalogue system
- Usage of existing flight dynamic models (implemented in Fortran)

Why not f2py?

Pro

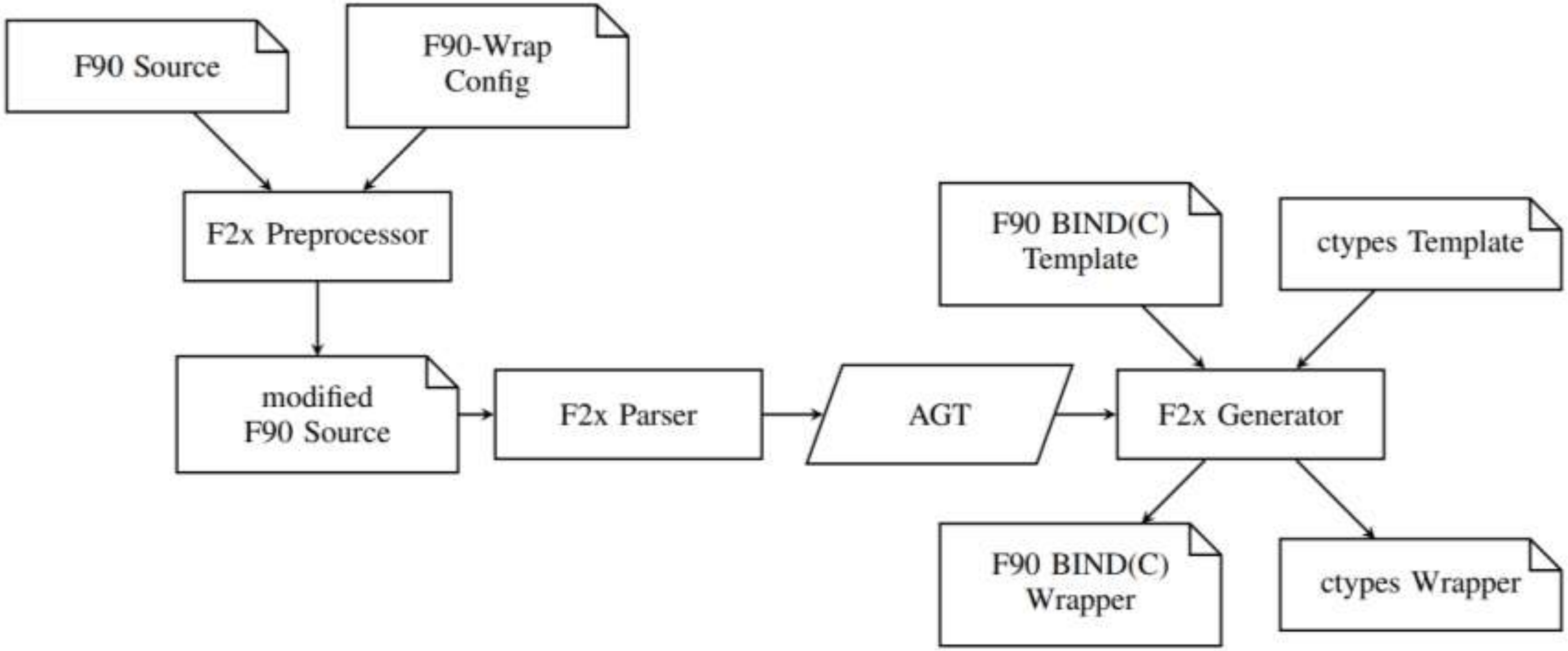
- Standard solution
- Stable
- Highly automated
- Easy to use (for simple cases)

Contra

- No derived type support
- Wrapper code highly compiler/version dependent
- Hard to extend
- Supports only Python

F2x Implementation

Overview



F2x Implementation
Parser

- Full LALR(1) parser using plyplus
  - Fortran not exactly a LALR(1) grammar…
  - plyplus discontinued by author ☹
  - Successor „lark“ nor suitable ☹

- Grammar borrowed from OpenFortran Project

- AST transformed into „AGT“
  - Decoupling
  - Streamlining of templates

- Other approaches (ANTLR) in planning

F2x Implementation
Template-based Code Generation

- Approach from *Model Driven Development*
- „Transformation of a model instance into text (source code) using templates."
- Input
  - (Model)
  - Model Instance
  - Template
- Output
  - Source code
- Support for new target (C#, Java, …): Write new Template!
  - Lots of knowledge in templates → non-trivial ☹

F2x Implementation

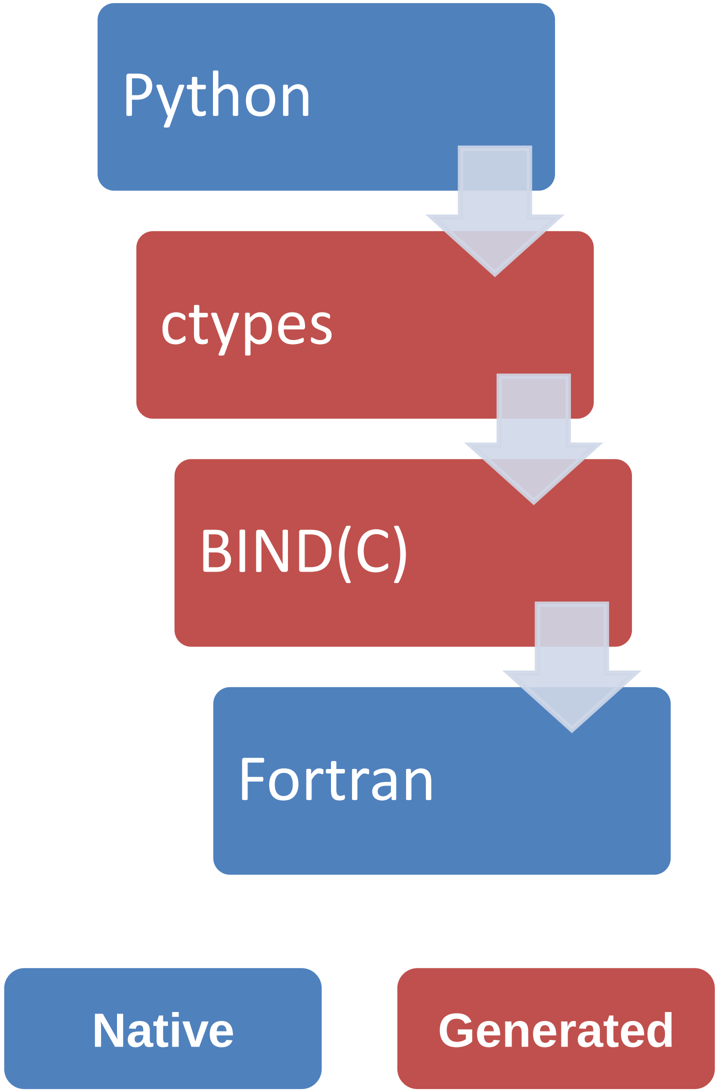
Results

- Two wrapping layers
  - Compiler-neutral interface
  - Use of compiler-neutral interface (from Python, C#, …)

- Different templates
  - Fortran – BIND(C)*°
  - Python – ctypes*°, Cython
  - C – Longjmp error handling*°
  - C# – Pinvoke

* currently in production use

° part of F2x distribution



F2x Implementation
Example

example.f90

```
MODULE EXAMPLE
    TYPE, PUBLIC :: POLYNOM
        INTEGER :: DEGREE
        REAL(8), DIMENSION(:), &
            ALLOCATABLE :: COEFF
    END TYPE
    PUBLIC EVAL_AT
CONTAINS
    FUNCTION EVAL_AT(P, X)
        REAL(8) :: EVAL_AT
        TYPE(POLYNOM), INTENT(IN) :: P
        REAL(8), INTENT(IN) :: X
        [...]
    END FUNCTION
END
```

example.py

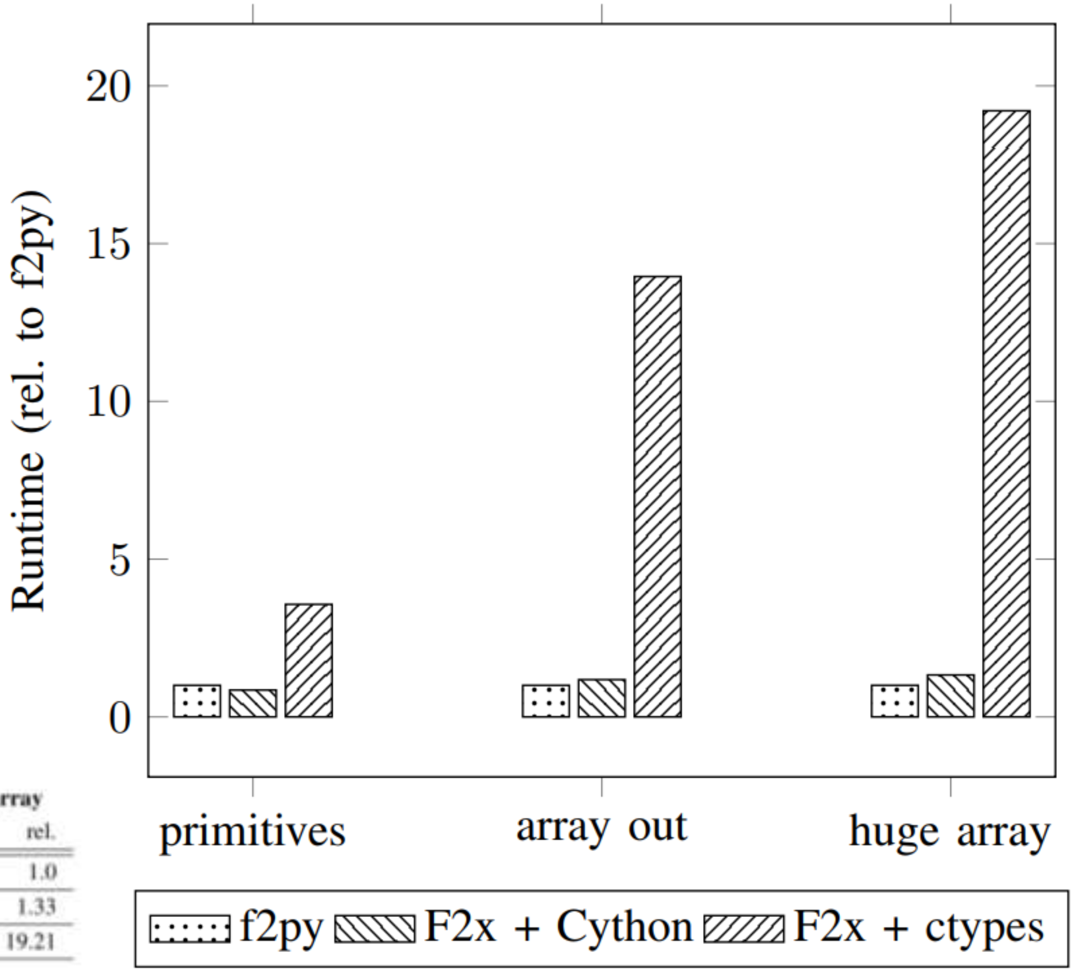
```
from example_glue import POLYNOM, EVAL_AT

# p: f(x) = 3.4 * x^3 + 1.2 * x.
p = POLYNOM(DEGREE=3,
            COEFF=[0, 1.2, 0, 3.4])
c = EVAL_AT(p, 5.6)

# Update offset of p.
p.COEFF[0] = -c
assert abs(EVAL_AT(p, 5.6)) < 0.0001
```

Usage in DLR

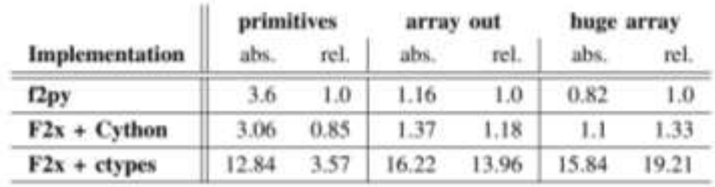
- Developed for BACARDI
  - Usage of Fortran flight dynamic codes from Python
- Used in other projects at GSOC
  - Usage of Fortran flight dynamic
  - Initial plan to implement templates for C#, Java, …
  - Provide different (RPC) interfaces from Python (REST, MQTT, …)
- Usage for production in different frameworks
  - Good testers and lots of feedback

Benchmark Results

Comparison with f2py



From my PyHPC paper
(Review pending…)



Implementation	**primitives**		**array out**		**huge array**	
	abs.	rel.	abs.	rel.	abs.	rel.
f2py	3.6	1.0	1.16	1.0	0.82	1.0
F2x + Cython	3.06	0.85	1.37	1.18	1.1	1.33
F2x + ctypes	12.84	3.57	16.22	13.96	15.84	19.21

https://doi.org/10.5281/zenodo.1405459

Status and Outlook

- Available under Apache License 2.0
- https://github.com/DLR-SC/F2x

- Support for many Fortran constructs
- Independent of compiler (tested with GNU and Intel)

- Templates:
  - BIND(C)
  - ctypes (slow!)

- Lot of improvemets plannded (but only little manpower)
  - Callbacks
  - Better parser
  - ~~Cython~~ CFFI template
  - Automatic module compilation…

Thank you!

What are your questions?

https://github.com/DLR-SC/F2x

E-Mail: michael.meinel@dlr.de

Twitter: @led02

Knowledge for Tomorrow