HIFIS | HELMHOLTZ FEDERATED IT SERVICES

RECOVERING KNOWLEDGE FROM OLD CODE

Fredo Erxleben, Helmholtz-Zentrum Dresden-Rossendorf

deRSE 2025 - 2025/02/27

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

SHORT INTRO

- RSE, trainer and consultant for HIFIS
- HIFIS offers RSE Consulting Service aimed at scientists in Helmholtz
- Wide range of covered topics
  - Licensing, publication, code review, project planning, ...

THE MISSION

- Use old code to match up crystal cross-sections
- Yields better results than other published algorithms.

How does it calculate its results?

THE MISSION

- Use old code to match up crystal cross-sections
- Yields better results than other published algorithms.

How does it calculate its results?

THE ISSUES

- Author is no longer available
- Few knowledgeable users left
- Unclear legal situation
- Sparse, outdated? documentation
- No version history

THE BIG L

- No license was initially given
- First code publication under CC 4.0
- Clarified intentions with author
- Check for potential license claims
  from former employer

→ Now under MIT

THE BIG L

- No license was initially given
- First code publication under CC 4.0
- Clarified intentions with author
- Check for potential license claims from former employer

→ Now under MIT

THE BIG R

- Code was not versioned at all
- Set up repository in GitLab (Helmholtz Codebase)
- Added relevant files for compilation
- Added verified documentation parts

→ Now under version control

THE MASTERPLAN

Extract math formulas from code

Create CI tests using known I/O as fixtures

Rewrite the code in a modern environment

Clear and easy.
What could possibly
go wrong?

A CLOSER LOOK AT THE CODE

- FORTRAN 77
- 10K+ lines
- Sparse documentation

Dealing with "historic" programming language:

- Transpile into something more "modern" or
- Learn how it works


```
      common /ngk/ ngk                                                  805
      character*4 nam1,nam2,text,ta1                                    806
      character*1 jsm                                                   807
      character* 10rs,rss                                               808
c                                                                       809
c cell parameters B-memory > work area                                  810
      if (ngk.gt.0) go to 10                                            811
      write (ioa,80)                                                    812
      n=0                                                               813
      return                                                            814
   10 write (io,70) (j,(dgg(k,j),k=1,3),(dggw(k,j),k=1,3),jsm(la(j)),   815
     1rss(j),nam1(j),(nam2(i,j),i=1,3),j=1,ngk)                         816
      if (io.ne.ioa) write (ioa,70) (j,(dgg(k,j),k=1,3),(dggw(k,j),k=1, 817
     13),jsm(la(j)),rss(j),nam1(j),(nam2(i,j),i=1,3),j=1,ngk)           818
      n=1                                                               819
      write (ioa,90)                                                    820
```

DOES IT COMPILE?

Challenge:

- There are a lot of compiler options for non-standard behaviour

Good news:

- Compiles out-of-the-box with minimal additional parameters

But: Minor warnings

```
gfortran -std=legacy -o piep17z piep17z.for
```

DOES IT COMPILE?

Challenge:

There are a lot of compiler options for non-standard behaviour

Good news:

Compiles out-of-the-box with minimal additional parameters

But: Minor warnings

```
gfortran -std=legacy -o piep17z piep17z.for
```

DOES IT RUN?

Yes. But:

- Needs the right files in the right place
- Needs domain expert to operate

```
~/piep-consult  ./piep17z
###################################
=======      P I E P      =======
======= VERSION 14-jun-17 =======
###################################
default parameters from file? (def.=yes)
parameter-file piep.par            ? (blank), otherwise name
-------------------------------------------------------------------------------
cell parameter file assigned: cell.dat           ,    61 sets
1st set read, unit: 20, file: cell.dat
-------------------------------------------------------------------------------
SAD data file: unit 30, file: sadm.dat           ,    17 sets
1st set loaded
-------------------------------------------------------------------------------
```

PEN & PAPER

Try the naïve approach…

Known: **input**, **commands**, **output**

- Learn which questions to tackle
- Figure out which techniques are promising

Reverse Engineering by hand does not scale!

PEN & PAPER

Try the naïve approach…
Known: **input**, **commands**, **output**

- Learn which questions to tackle
- Figure out which techniques are promising

Reverse Engineering by hand does not scale!

TOOL SUPPORT

Available tools:
- Debugger (gdb), Static Analysis (valgrind)
- Language-specific tools (commercial)
- kscope (discontinued)

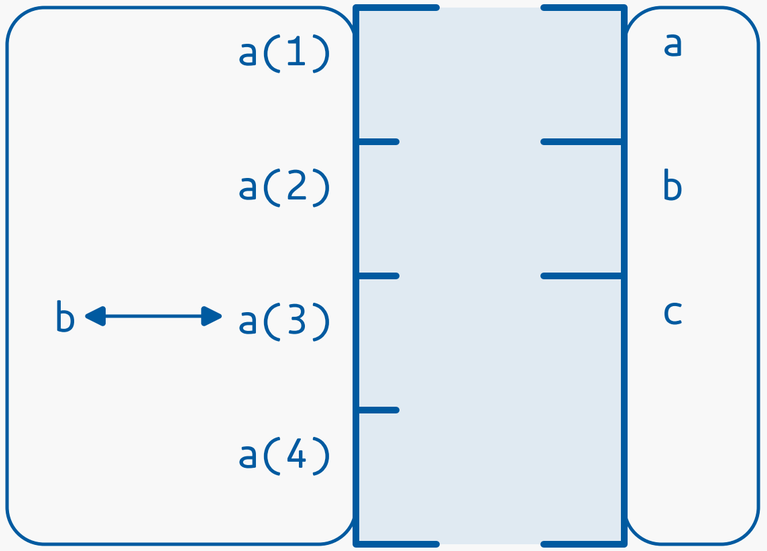
Having tools is nice, but which part of the code do we need to investigate?

→ Need something to annotate the code

SIDE NOTE: FUN FEATURES OF FORTRAN

- Split variable definitions
- *Common blocks + equivalence statements*
- Data type can depend on variable name
- Subroutines can return to place different from call
- Function return values via implicit variables
- …

→ Vastly different from currently established languages.



FEATURE WHISHLIST

- Get overview over code structure
- Identify variables that share memory
- Identify types, initial values and dimensions
- Allow to annotate everything in situ
  - Support for formulas
- Identify which lines generate a given output
- Vizualize the data flow between statements

More to come ...

FEATURE WHISHLIST

- Get overview over code structure
- Identify variables that share memory
- Identify types, initial values and dimensions
- Allow to annotate everything in situ
  - Support for formulas
- Identify which lines generate a given output
- Vizualize the data flow between statements

More to come ...

INITIAL STEPS

- Parse the code, build a OO-model
- Implement UI for explorative tasks
- Add algorithms for trivial reasoning

All we need is an EBNF for FORTRAN 77
Can't be that hard to find

PARSING FORTRAN 77

- The standard is a recommendation
- Many "optional" language features
- No complete EBNF to be found
- AI was not helpful

```
// Entry point for parsing
line                        : assignment | _statement | type_declaration
assignment                  : _variable_access  "=" _expression
type_declaration            : TYPE_NAME type_size? _comma_sequence{individual_declaration}
    type_size               : "*" (_expression | DUMMY_TYPE_SIZE)
    individual_declaration  : _variable_access type_size?
    // NOTE According to Section 4.14,
    // initializing variables in type statements is non-standard
```

→ Use LARK as parser
→ Write grammar matching required features
→ Adapt while learning about FORTRAN 77

→ Focus on features used in example code
→ Working for selected subset
→ ~200 rules / terminals

CURRENT STATE

Can parse the example code and build a model

```
statements OK 7718 / 7718 (100.0%)
statements XX 0 / 7718 (0.0%)
# Common blocks: 36
# Functions: 17
# Subroutines: 91
# Block data: 1
# Overall variables: 4806
```

CURRENT STATE

Can parse the example code and build a model

```
statements OK 7718 / 7718 (100.0%)
statements XX 0 / 7718 (0.0%)
# Common blocks: 36
# Functions: 17
# Subroutines: 91
# Block data: 1
# Overall variables: 4806
```

NEXT STEPS

Un-prototype:

- Clean up code and grammar
- Add doc, license, readme

Investigate model:

- Make explorable
- Make annotateable

HIFIS | HELMHOLTZ FEDERATED IT SERVICES

This was

RECOVERING KNOWLEDGE FROM OLD CODE

Part 1

Stay tuned!

Fredo Erxleben, f.erxleben@hzdr.de
https://hifis.net