Parser Generation by Example for LEGACY Pattern Languages

V. Zaytsev @ GPCE'17 @ SPLASH

Solution combines:

- …by example
- grammar inference
- parsing
- data binding

raincode LABS

compiler experts

Problem combines:

- fourth generation language
- bespoke compiler development
- bizarre notation

Notation sample

```
$$FILE 06/07/2017 23:59:59
$$FOO    ABCD      Y 06/07/2017 23:59:59 XYZ
 A 1 00010 00 0000 Y Y N Y NAMEA     NAMEB     S
 C 2 00015 02 0000 Y Y Y Y NAMEDDDD NAME EEE S
 F 5 00030 00 0020 Y N N Y NAMEG     NAMEH     S
$$BAR    EFGHKLMN Y 06/07/2017 23:59:59 N/A
 A LONGER_NAME_FOR_ENTITY                  999 10.0
 A ANSWER_TO_THE_ULTIMATE_QUESTION          42  7.5
```

No ready solution

- language is unknown ⇒ verbal documentation
- notation is unknown ⇒ no free parser/grammar
- position-oriented notation ⇒ no demand so no support
- incremental development ⇒ no academic interest
- error handling/reporting/recovery
- third party products are evil

BNF ⇒ *PCB?*

- **Patterns**
  - break a line into fields
- **Commitments**
  - demand additional structure from the fields
- **Bindings**
  - denote where processed fields go

Patterns

Patterns

```
A 1 00010 00 0000 Y Y N Y NAMEA    NAMEB    S
_ _ _____ __ ____ _ _ _ _ ________ ____ ___ S
A B C     D  E    F G H I J        K
```

Patterns

A

B

C

D

Commitments

```
A       (DLI|DB2|N/A)

B       [0-9A-Z ]+

C       [YN ]

D
(SYNC |ASYNC|EVENT|)
```

Postprocessing

```
A?T    (DLI)
A?T    (DB2)
A      (DLI|DB2|N/A)

B~     [0-9A-Z ]+

C?TF   [YN ]

D:Sync/Async/Event/Undefined
(SYNC |ASYNC|EVENT|)
```

Typing

```
bool   DLI  := A?T    (DLI)
bool   DB2  := A?T    (DB2)
            :- A      (DLI|DB2|N/A)

str Input   := B~     [0-9A-Z ]+

bool Flag   := C?TF   [YN ]

enum Synch:= D:Sync/Async/Event/Undefined
       (SYNC |ASYNC|EVENT|)
```

Enumeration bindings

```
enum   Module    := D:Main/Sub/Undefined [MS ]
```

```
public override string ToString()
     {
        return string.Format("  PC{0} {1} {2}      {3} {4}  {5}      {6}   {7} {8} {9} {10} {11} ",
             Cmps ? "CMPS" : Cics ? "CICS" : "   ",
             Input.PadRight(8, ' '),
             Output.PadRight(8, ' '),
             UnparseEnum(Module),
             UnparseEnum(Synchronisation),
             Database ? "DB2" : "N/A",
             UnparseEnum(Locality),
             Name.PadRight(8, ' '),
             Flag1 ? 'Y' : ' ',
             Flag2 ? 'Y' : ' ',
             Flag3 ? 'D' : ' ',
             Flag4 ? 'Y' : ' ',
             null);
     }
```

```
private string UnparseEnum(ModuleEnum x)
     {
        switch (x)
        {
             case ModuleEnum.Main:
                return "M";
             case ModuleEnum.Sub:
                return "S";
             case ModuleEnum.Undefined:
                return " ";
             default:
                throw new NotImplementedException(x +
                " is not supported by unparsing of " + Module);
        }
     }
```

Process

- Infer from codebase
  - commitments underspec: 000DD
  - bindings underspec: M/S
  - nominal underspec: Name1, Name2, Flag1, Flag2
- Joint design sessions

Aftermath

- Spec inferred “by example”
- Spec refined in collab with domain/legacy experts
- Easily adjusted multiple times
- Optimised parser and unparser generated
- Takes ~7 minutes to parse ~20k files (9135 kLOC, 2.3 GB)
- What can you learn?

Sources

- Title
  - https://doi.org/10.1145/937563.937566
  - https://doi.org/10.1145/362991.363001
  - https://doi.org/10.1145/361532.361539
  - https://doi.org/10.1145/355604.361597
  - https://doi.org/10.1145/3131851.3131868
  - http://amzn.to/2z36vlq
  - https://doi.org/10.1145/356635.356641
  - https://doi.org/10.1145/2483760.2483766
  - https://doi.org/10.1145/971258.971268
  -