Error Recovery in PEGs

Michael Rüfenacht

m.ruefenacht@students.unibe.ch

Roadmap

1. Prerequisites
2. Goals
3. Problems
4. Conclusion

Who is who?

1. Prerequisites

1. Prerequisites

Parsing Expression Grammars

start	←	value
value	←	string
		/ integer
		/ float
		/ boolean
		/ null
		/ object
		/ array
object	←	‘ { ‘ properties **?** ‘ } ’
properties	←	property (‘ , ’ property)*****
property	←	key ‘ : ‘ value

1. Prerequisites

Parsing Expression Grammars

```
start      ← value
value      ← string
           / integer
           / float
           / boolean
           / null
           / object
           / array
object     ← '{' properties? '}'
properties ← property (',' property)*
property   ← key ':' value
```

Parsing Expression Grammars

```
start       ←  value

value       ←  string
            /  integer
            /  float
            /  boolean
            /  null
            /  object
            /  array

object      ←  '{' properties? '}'

properties  ←  property (',' property)*

property    ←  key ':' value
```

1. Prerequisites

Parsing Expression Grammars

```
start       ←  value

value       ←  string
            /  integer
            /  float
            /  boolean
            /  null
            /  object
            /  array

object      ←  '{' properties? '}'

properties  ←  property (',' property)*

property    ←  key ':' value
```

Parsing Expression Grammars

```
start       ←  value

value       ←  string
            /  integer
            /  float
            /  boolean
            /  null
            /  object
            /  array

object      ←  '{' properties? '}'

properties  ←  property (',' property)*

property    ←  key ':' value
```

1.23

Parsing Expression Grammars

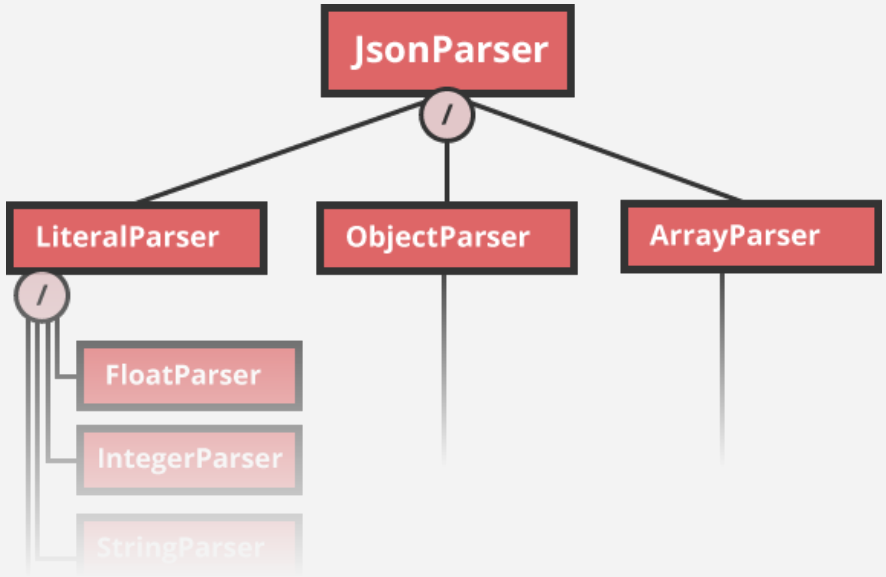
```
start       ← value
value       ← string
              / integer
              / float
              / boolean
              / null
              / object
              / array
object      ← '{' properties? '}'
properties  ← property (',' property)*
property    ← key ':' value
```

1.23

1. Prerequisites

PetitParser

- **parser combinator** framework
- following the **PEG** formalism



1. Prerequisites

Java Script Object Notation (JSON)

```
{
      "object" : {
        "key" : "valuePairs"
      }
    , "string" : "hello"
    , "boolean" : true
    , "null"    : null
    , "integer" : 100
    , "float"   : 10.3
    , "array"   : [
          "comma"
        , "separated"
        , "values"
    ]
}
```

1. Prerequisites

Java Script Object Notation (JSON)

```
{
      "object" : {
        "key" : "valuePairs"
      }
    , "string" : "hello"
    , "boolean" : true
    , "null"    : null
    , "integer": 100
    , "float"   : 10.3
    , "array"   : [
        "comma"
      , "separated"
      , "values"
    ]
}
```

Java Script Object Notation (JSON)

```
{
      "object" : {
        "key" : "valuePairs"
      }
    , "string" : "hello"
    , "boolean" : true
    , "null"    : null
    , "integer" : 100
    , "float"   : 10.3
    , "array"   : [
          "comma"
        , "separated"
        , "values"
    ]
}
```

2. Goals

Improve the error handling of **PetitParser** and provide the **toolset** to exploit the benefits of permissive parsing and **syntax-error recovery**.

PEGs, PetitParser and Caveats

3. Problems

Main Problems

1. Failure Location
2. Error Detection
3. Error Classification
4. Recovery

3. Problems

Failure Location

```
{
      "object" : {
        "key" : "valuePairs"
      }
    , "string" : "hello"
    , "boolean" : true
    , "null"    : null
    , "integer": 100
    , "float"   : 10.3
    , "array"   : [
          "comma"
        , "separated"
        , "values"
    ]
}
```

3. Problems

Failure Location

```
{
      "object" : {
        "key" : "valuePairs"
      }
    , "string" : "hello"
    , "boolean" : true
    , "null"    : null
    , "integer": 100
    , "float"   : 10.3
    , "array"   : [
          "comma"
        , "separated"
          "values"
    ]
}
```

3. Problems

Failure Location

```
{  ← failure: " '[' expected at position 1 "
     "object" : {
       "key" : "valuePairs"
     }
   , "string" : "hello"
   , "boolean" : true
   , "null"    : null
   , "integer" : 100
   , "float"   : 10.3
   , "array"   : [
         "comma"
       , "separated"
         "values" ← source
   ]
}
```

3. Problems

Failure Location

```
{  ← failure: " '[' expected at position 1 "
     "object" : {
       "key" : "valuePairs"
     }
   , "string" : "hello"
   , "boolean" : true
   , "null"    : null
   , "integer" : 100
   , "float"   : 10.3
   , "array"   : [
         "comma"
       , "separated"
         "values" ← source
   ]
}
```

value ← …

/ object

/ array

3. Problems

Failure Location

```
{  ← failure: " '[' expected at position 1 "
     "object" : {
       "key" : "valuePairs"
     }
   , "string" : "hello"
   , "boolean" : true
   , "null"    : null
   , "integer" : 100
   , "float"   : 10.3
   , "array"   : [
         "comma"
       , "separated"
         "values" ← source
   ]
}
```

3. Problems

Error Detection

1. **Failure**
   - current parser is **not able to recognize** the input
   - triggers **backtracking**
   - usually **not equal to a syntax-error**
2. **Success**
   - the **parsing result**

Error Detection

`null`

value ← **string**

/ integer

/ float

/ boolean

/ null

...

3. Problems

Error Detection

```
null
```

value ← **string**

/ ...

Returns a **Failure** with message **‘ “ expected ‘**

Error Detection

```
null
```

```
value ← string → " expected
      / integer → digit exp ...
      / float   → digit exp ...
      / boolean → false exp...
      / null → success
      ...
```

3. Problems

Error Classification

1. **Omission** - *missing token*
2. **Submission** - *additional token*
3. **Substitution** - *wrong token*

3. Problems

Error Classification - Omission

```
{
      "float" : 10.3
    , "array"     : [
        "comma"
       , "separated"
        "values"
    ]
}
```

Error Classification - Submission

```
{
      "float" : 10.3
    , "array"     : [
        "comma"
      , "separated"
      , "values"
    ]
    ]
}
```

Error Classification - Substitution

```
{
      "float"  =  10.3
    , "array"     : [
        "comma"
      , "separated"
      , "values"
    ]
}
```

3. Problems

Error Recovery

1. **Repairing**
2. **Skipping**

Repairing

```
{
      "float"  =  10.3
    , "array"     : [
        "comma"
      , "separated"
      , "values"
    ]
}
```

3. Problems

Repairing

```
{
      "float"   :   10.3
    , "array"     : [
        "comma"
      , "separated"
      , "values"
    ]
}
```

3. Problems

Skipping

```
{
     "float"   :  Nonsense
   , "array"     : [
       "comma"
     , "separated"
     , "values"
   ]
}
```

3. Problems

Skipping

```
{
      "float"   :  Nonsense
    , "array"     : [
        "comma"
      , "separated"
      , "values"
    ]
}
```

The concept

1. improve **location** and **detection of syntax errors**
2. implement **local error classification**
3. choose according recovery (or just try the possibilites)
4. **deliver feedback**