An Augmented Backus-Naur Format, (ABNF),
Parser Generator for Erlang

Anders Nygren
anygren@txm.com.mx

Contents

- ABNF
- Using abnfc
- Implementation
- Todo
Why abnfc?

- ABNF used for specifying many important protocols, e.g. HTTP, SIP, SDP
- Handwritten parsers are
  - A lot of work
  - Error prone
- Not practical, (impossible?), to use yecc

What is ABNF?

- Augmented Backus-Naur Form
- Used by IETF for specifying protocols
- Initially informally defined in the RFCs where it was used
- later defined in a series of RFCs, currently in RFC 5234
Rule

Name = elements CRLF

Ex,
Request-Line = Method SP Request-URI SP SIP-Version CRLF

Terminal Values

Binary, Decimal, Hex
CR = %d13
CR = %x0D
Sequence
CRLF = %d13.10
String, (case insensitive)
rule = “abc”
Concatenation

Rule = Rule1 Rule2 … RuleN

foo = %x61 ; a
bar = %x62 ; b
mumble = foo bar foo
Accepts “aba”

Alternation

Rule = Rule1 / Rule2 / … / RuleN

foo = %x61 ; a
bar = %x62 ; b
A-or-b = foo / bar
Accepts “a” and “b”
Value Range

DIGIT = %x30-39

Is equivalent to

DIGIT = “0” / “1” / “2” / “3” / “4” / “5” / “6” / “7”

/ “8” / “9”

Sequence Group

Rule = (Rule1 Rule2 ... RuleN)
Repetition

*rule ; 0 or many occurrences
<n>*rule ; n or more occurrences
*<m>rule ; 0 to m occurrences
<n>*<m>rule ; n to m occurrences
<n>rule ; exactly n occurrences, equivalent to <n>*<n>rule

Optional

[foo bar]
Equivalent to
0*1(foo bar)
“Imported” Rules

Rules defined in other RFCs are frequently reused.

But there is no formal way of specifying the imports. Normally it is done with

- A comment in the ABNF specification
- A note in the RFC text

Using abnfc

Grammar definition file: my_mod.abnf
Name = elements : Erlang_Code.

Ex.
callid  = word [ "@" word ] : lists:flatten(_YY).
Variable Bindings

In the Erlang code the following variables are available

- \(_{YY}\) : Bound to the complete match for the rule
- \(_{YY1}\) to \(_{YYn}\) : Bound to each part of the match

ex.

\[
\text{opt-name} = "\(\text{" name ")"} : \{\text{opt_name}, _{YY2}\}.
\]

Variable Bindings

\[
\text{Allow} = \"Allow\" \text{ HCOLON } [\text{Method } * (\text{COMMA Method})] : \\
\text{Allowed} = \text{case } _{YY3} \text{ of} \\
[] \to []; \\
[[M1,Ms]] \to [M1][M][\text{"COMMA",M}<-Ms]] \\
\text{end,} \\
\{\text{\'Allow\', Allowed}\}.
\]
Implementation

Three main parts

- Syntax specification parser
- Transformations/optimizations
- Code generation

Syntax Spec. Parser

- Originally hand written using parser combinators
- Later generated by abnfc, (ABNF is specified in ABNF)
- Generates a simple AST
Transformations

- Convert AST to an internal representation
- A number of passes that performs transformations on the internal format
  - Remove \{repeat, 1, 1\}
  - Remove alternation with only one element
  - Remove concatenation with only one element
  - Merge num_val elements, (pending)
  - Inline imported rules when possible to improve optimizations, (pending)

Code generation

- The generated code currently uses parser combinators
  - Simple
  - Not very efficient
- Parser functions for all rules are exported
- A my_mod.hrl is always included, this makes it possible to import rules from other modules
Request = Request-Line *( message-header )
     CRLF [ message-body ] :
     {'Request', _YY1, _YY2, _YY4}.

'Request'() ->
fun (T) ->
  __P=abnf_rt:seq([ 'Request-Line'(),
                   abnf_rt:repeat(0, infinity, 'message-header'()),
                   'CRLF'(),
                   abnf_rt:repeat(0, 1, 'message-body'())]),
  case __P(T) of
    {ok, [_YY1, _YY2, _YY3, _YY4]=_YY, __Rest} ->
      __Ret = begin
        { 'Request' , _YY1 , _YY2 , _YY4 }
        end,
        {ok, __Ret, __Rest};
    fail ->
      fail
      end
end.
Limitations

- Limited backtracking
  - Some workarounds
    - Reorder alternatives
    - Handwritten parsers for difficult cases
    - Modify rules and do validation in Erlang

- No error messages

Todo

- More transformations/optimizations
- Better code generation
- Support older ABNF syntax
- Generate encoding functions (?)
Expertise Makes it simple

Anders Nygren
SL, product development and solutions
Telexpertise de México S.A. de C.V.
Phone: +52 (844)438-4800
anygren@txm.com.mx
www.txm.com.mx