Rakudo
Perl 6 on Parrot
Written in Perl 6
Parser is in Perl 6 rules
rule unless_statement { 
    $<\text{sym}>=[\text{unless}]
    <\text{EXPR}> <\text{block}>
    {{*}}
}

rule unless_statement { 
    $<\text{sym}>=[\text{unless}] <\text{EXPR}> <\text{block}> {[*]} 
}
rule unless_statement {
    $<sym>$=\[unless\]
    <EXPR> <block> {*}
}
rule unless_statement {
    $<\text{sym}> = \text{[unless]} \\
    <\text{EXPR}> <\text{block}>
    {*}
}
rule unless_statement {
    <$sym>=[unless]
    <EXPR> <block>
    {*}
}

rule unless_statement {
  $<sym>=[unless]
  <EXPR> <block>
  {[*]}
}

Parse tree to AST transform in NQP
method unless_statement($/) {
    my $then := $( $<block> );
    $then.blocktype('immediate');
    my $past := PAST::Op.new( $( $<EXPR> ), $then, :pasttype('unless'), :node( $/ )
    )
    make $past;
}

method unless_statement($/) {
    my $then := $( $<block> );
    $then.blocktype('immediate');
    my $past := PAST::Op.new(
        $( $<EXPR> ), $then,
        :pasttype('unless'),
        :node( $/ )
    );
    make $past;
}

method unless_statement($/) {
    my $then := $( $<block> );
    $then.blocktype('immediate');
    my $past := PAST::Op.new(
        $( $<EXPR> ), $then,
        :pasttype('unless'),
        :node( $/ )
    );
    make $past;
}
method unless_statement($/)
{
    my $then := $( $<block> );
    $then.blocktype('immediate');
    my $past := PAST::Op.new(
        $( $<EXPR> ), $then,
        :pasttype('unless'),
        :node( $/ )
    );
    make $past;
}

method unless_statement($/) {
    my $then := $( $<block> );
    $then.blocktype('immediate');
    my $past := PAST::Op.new(
        $( $<EXPR> ), $then,
        :pasttype('unless'),
        :node( $/ )
    );
    make $past;
}

method unless_statement($/) {
    my $then := $( $<block> );
    $then.blocktype('immediate');
    my $past := PAST::Op.new( $( $<EXPR> ), $then, :pasttype('unless'), :node( $/ )
    );
    make $past;
}
method unless_statement($/=) { 
  my $then := $( $<block> );
  $then.blocktype('immediate');
  my $past := PAST::Op.new( $( $<EXPR> ), $then, :pasttype('unless'), :node( $/= ) );
  make $past;
}

method unless_statement($/){
  my $then := $( $<block> );
  $then.blocktype('immediate');
  my $past := PAST::Op.new( $( $<EXPR> ), $then, :pasttype('unless'), :node( $/ ) );
  make $past;
}

Features
Features
(Before GPW)
Variables
Conditionals
Loops
Partial Implementation of Junctions
mod_perl6
During your talks, I did some hacking...
Role
Composition
Now Works
Now Works
Started parsing grammars and rules too
regex Year {\d\d\d\d\d};
regex Location {German|French|Italian|London|Dutch|Ukrainian};
regex PerlConference {<Location>\sPerl\sWorkshop[\s<Year>]??};

if "German Perl Workshop 2008" ~~ PerlConference {
    say "GPW 2008 is a Perl conference.";
}

if "French Perl Workshop" ~~ PerlConference {
    say "FPW is a Perl conference.";
}

if "RailsConf" ~~ PerlConference {
    say "RailsConf is not a Perl conference.";
}
My approach: breadth first implementation
Coming soon
Support for writing built-ins in Perl 6
More work on grammars, match objects and so on
What people doing stuff in Perl 6 ask for