Shift Resolve Parsing

Simple, Linear Time, Unbounded Lookahead

José Fortes Gálvez[1]
Sylvain Schmitz[2]   Jacques Farré[2]

[1]Universidad de Las Palmas de Gran Canaria

[2]Laboratoire I3S, Université de Nice - Sophia Antipolis

August 23, 2006

Standard ML

Milner et al. [1997]

Example

```
datatype 'a option = NONE | SOME of 'a

fun filter pred l =
  let
    fun filterP (x::r, l) =
          case (pred x)
            of SOME y => filterP(r, y::l)
             | NONE   => filterP(r, l)
      | filterP ([], l) = rev l
  in
    filterP (l, [])
  end
```

The Issue
SML Compilers Behaviour

```
| filterP ([], l) = rev l
```

- MLton
- Moscow ML
- Poly/ML
- SML/NJ

```
Error: match.sml 9.25.
  Syntax error: replacing  EQUALOP with  DARROW.
```

The Issue
SML Compilers Behaviour

```
  |   filterP ([], l) = rev l
```

- MLton
- Moscow ML
- Poly/ML
- SML/NJ

```
! Toplevel input:
!      |  filterP ([], l) = rev l
!                         ^
! Syntax error.
```

The Issue
SML Compilers Behaviour

```
| filterP ([], l) = rev l
```

- MLton
- Moscow ML
- Poly/ML
- SML/NJ

```
Error: => expected but = was found
```

The Issue
SML Compilers Behaviour

```
  | filterP ([], l) = rev l
```

- MLton
- Moscow ML
- Poly/ML
- SML/NJ

```
stdIn:7.24-7.29 Error: syntax error:
  deleting  EQUALOP ID
```

The Issue
Partial SML Grammar

$$\begin{aligned}
<\!dec\!> &\rightarrow \textbf{fun}\ \ <\!fvalbind\!> \\
<\!fvalbind\!> &\rightarrow <\!sfvb\!>|<\!fvalbind\!>\ \ '|'\ \ <\!sfvb\!> \\
<\!sfvb\!> &\rightarrow vid\ \ <\!atpats\!> = <\!exp\!> \\
<\!atpats\!> &\rightarrow <\!atpat\!>|<\!atpats\!>\ <\!atpat\!>
\end{aligned}$$

$$\begin{aligned}
<\!exp\!> &\rightarrow \textbf{case}\ \ <\!exp\!>\ \ \textbf{of}\ \ <\!match\!> \\
<\!match\!> &\rightarrow <\!mrule\!>|<\!match\!>\ \ '|'\ \ <\!mrule\!> \\
<\!mrule\!> &\rightarrow <\!pat\!>\ \ =>\ \ <\!exp\!>
\end{aligned}$$

$$<\!pat\!> \rightarrow vid\ \ <\!atpat\!>$$

The Issue

Shift/Reduce Conflict

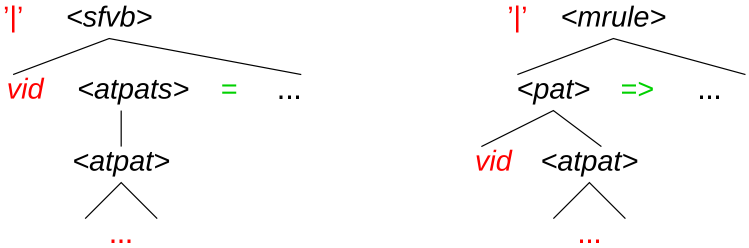
- GNU/bison
- Unbounded lookahead needed

```
 9 exp: "case" exp "of" match .
12 match: match . '|' mrule

  '|'  reduce using rule 9 (exp)
  '|'  shift, and go to state 38
```

The Issue
Shift/Reduce Conflict

- GNU/bison
- Unbounded lookahead needed



Solutions

1. No human intervention
2. Unambiguity
3. Linear parsing time

- Syntactic predicates
- LR-Regular parser
- Noncanonical LR(1) parser
- Noncanonical DR parser
- Generalized LR parser

Solutions

1. No human intervention
2. Unambiguity
3. Linear parsing time

- Syntactic predicates
- LR-Regular parser
- Noncanonical LR(1) parser
- Noncanonical DR parser
- Generalized LR parser

Solutions

1. No human intervention
2. Unambiguity
3. Linear parsing time

- Syntactic predicates
- LR-Regular parser
- Noncanonical LR(1) parser
- Noncanonical DR parser
- Generalized LR parser

Solutions

1. No human intervention
2. Unambiguity
3. Linear parsing time

- Syntactic predicates
- LR-Regular parser
- Noncanonical LR(1) parser
- Noncanonical DR parser
- Generalized LR parser

Solutions?

1. No human intervention
2. Unambiguity
3. Linear parsing time

- Syntactic predicates [Parr and Quong, 1995]
- LR-Regular parser
- Noncanonical LR(1) parser
- Noncanonical DR parser
- Generalized LR parser

Solutions?

1. No human intervention
2. Unambiguity
3. Linear parsing time

- Syntactic predicates
- LR-Regular parser [Bermudez and Schimpf, 1990, Farré and Fortes Gálvez, 2001]
- Noncanonical LR(1) parser
- Noncanonical DR parser
- Generalized LR parser

Solutions?

1. No human intervention
2. Unambiguity
3. Linear parsing time

- Syntactic predicates
- LR-Regular parser
- Noncanonical LR(1) parser [Tai, 1979, Schmitz, 2006]
- Noncanonical DR parser
- Generalized LR parser

Solutions?

1. No human intervention
2. Unambiguity
3. Linear parsing time

- Syntactic predicates
- LR-Regular parser
- Noncanonical LR(1) parser
- Noncanonical DR parser [Farré and Fortes Gálvez, 2004]
- Generalized LR parser

Solutions?

1. No human intervention
2. Unambiguity
3. Linear parsing time

- Syntactic predicates
- LR-Regular parser
- Noncanonical LR(1) parser
- Noncanonical DR parser
- Generalized LR parser [Tomita, 1986]

Solutions?

No solution with

1. No human intervention
2. Unambiguity
3. Linear parsing time

- Syntactic predicates
- LR-Regular parser
- Noncanonical LR(1) parser
- Noncanonical DR parser
- Generalized LR parser

Solutions

A solution with

1. No human intervention
2. Unambiguity
3. Linear parsing time

- Syntactic predicates
- LR-Regular parser
- Noncanonical LR(1) parser
- Noncanonical DR parser
- Generalized LR parser

} Shift Resolve parser

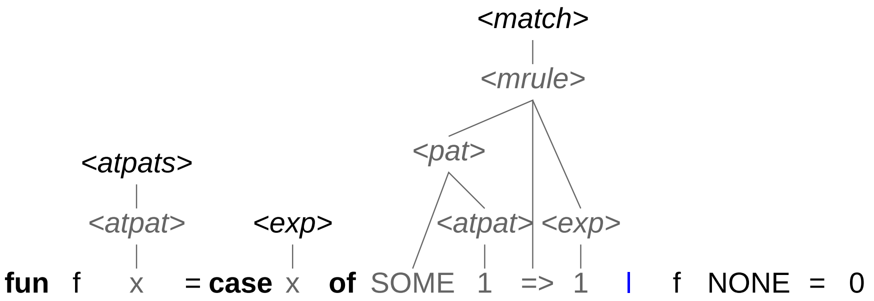
Shift-Resolve Parsing

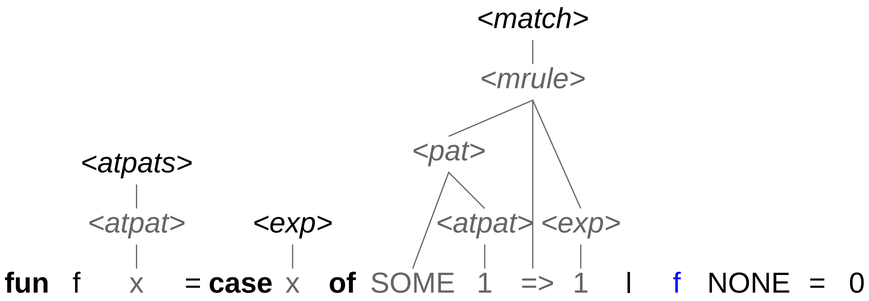
- noncanonical
- $k = 1$ reduced lookahead symbol
- resolve $=$ reduce $+$ pushback: emulates a bounded reduced lookahead without any preset bound

Shift-Resolve Parsing

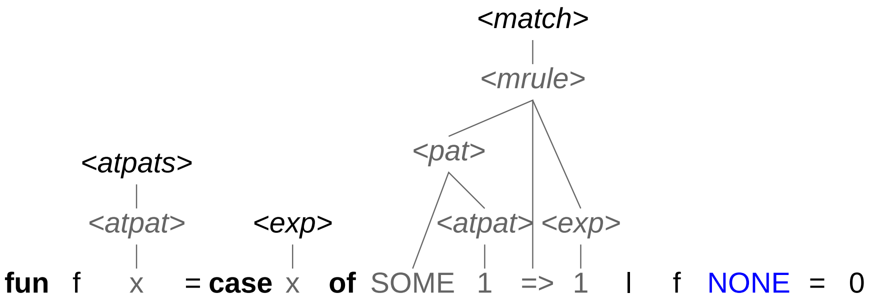
- noncanonical
- $k = 1$ reduced lookahead symbol
- resolve = reduce + pushback: emulates a bounded reduced lookahead without any preset bound

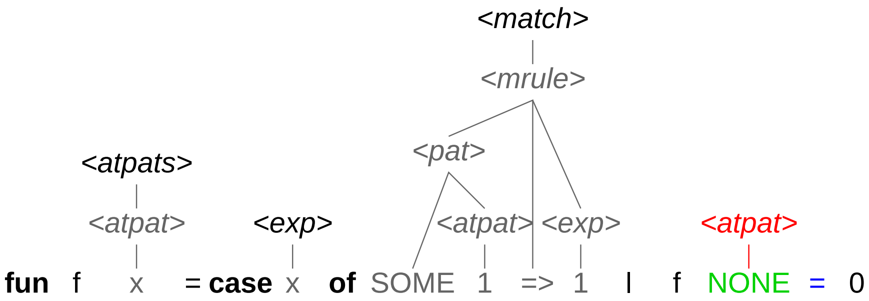
fun f x = **case** x **of** SOME 1 => 1 | f NONE = 0

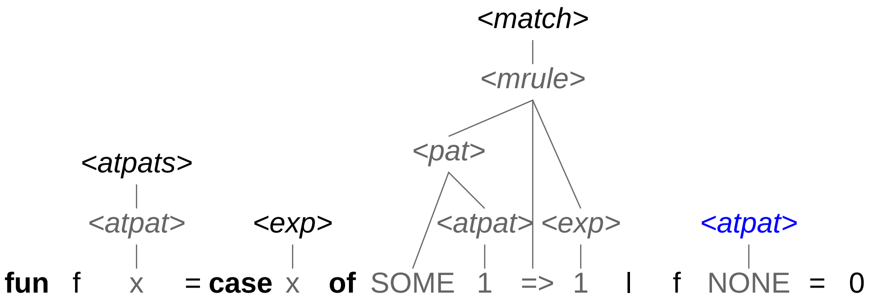

<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0



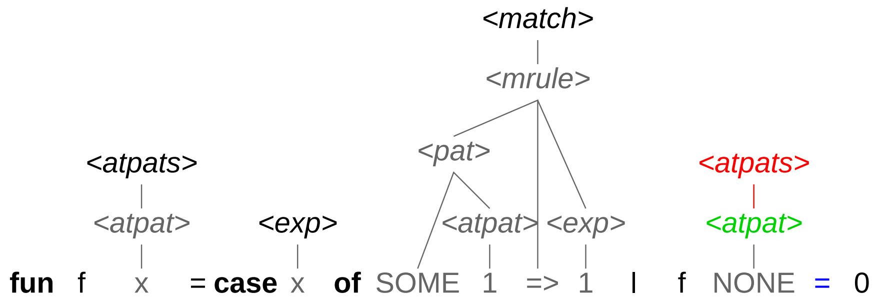
<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0

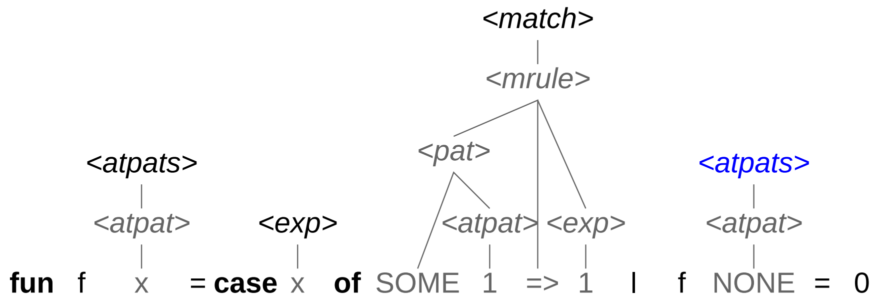

<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0

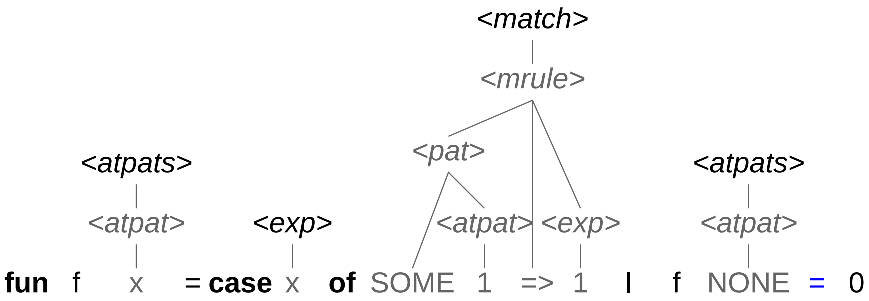

<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
fun f x = case x of SOME 1 => 1 | f NONE = 0

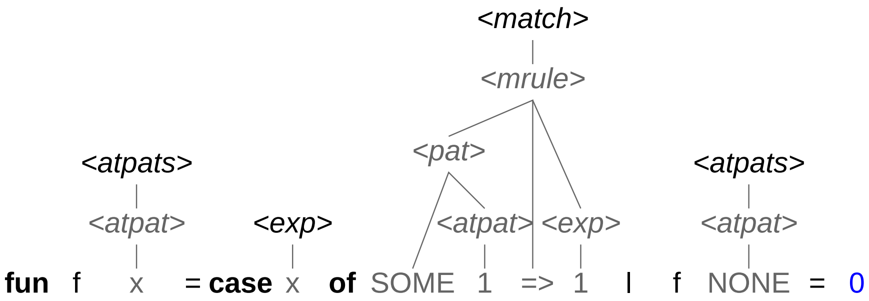


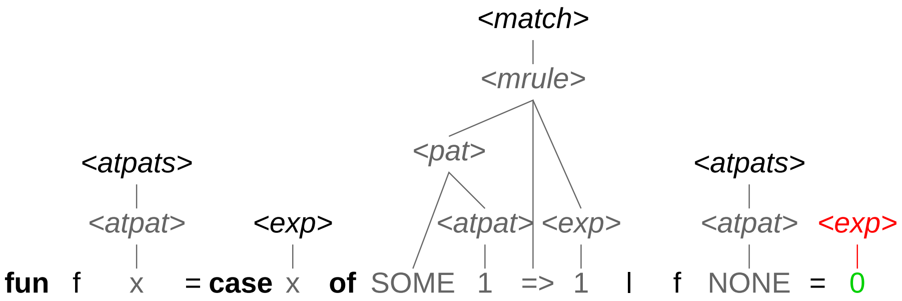
<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
fun f x = case x of SOME 1 => 1 | f NONE = 0


<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpats>
<atpat>
fun f x = case x of SOME 1 => 1 | f NONE = 0

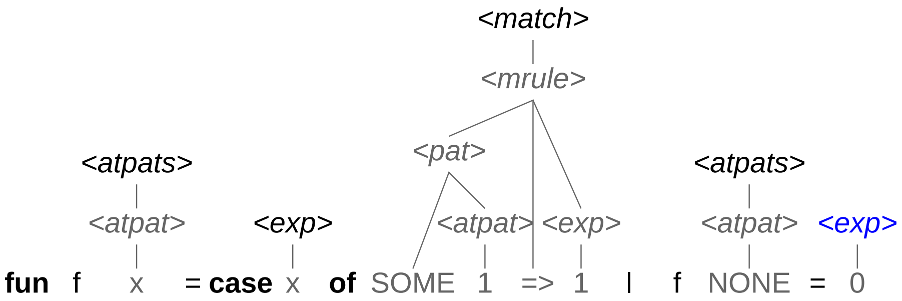

<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpats>
<atpat>
fun f x = case x of SOME 1 => 1 | f NONE = 0

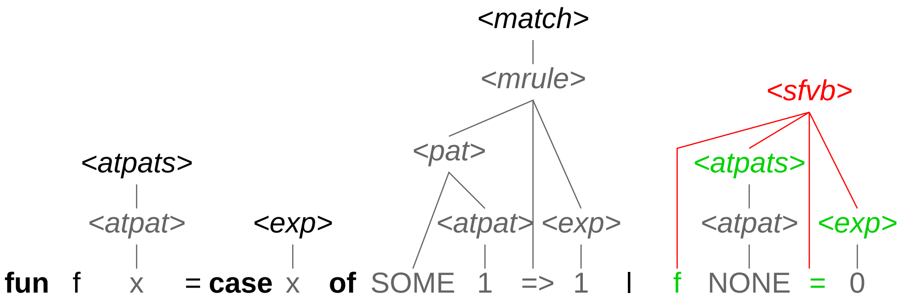

<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpats>
<atpat>
fun f x = case x of SOME 1 => 1 | f NONE = 0

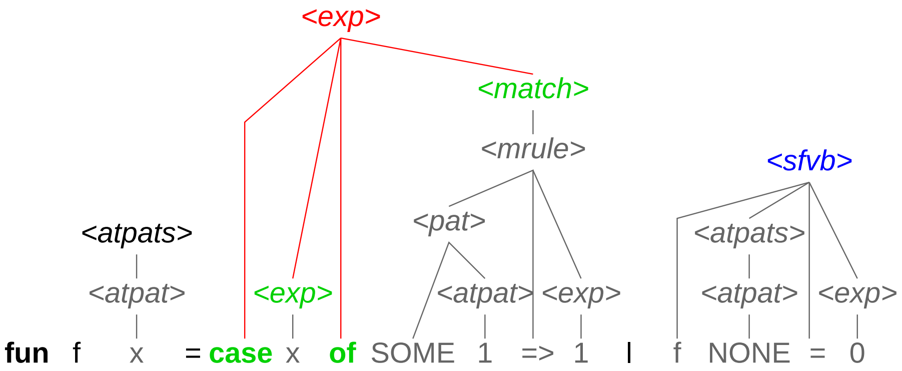

<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpats>
<atpat>
fun f x = case x of SOME 1 => 1 | f NONE = 0

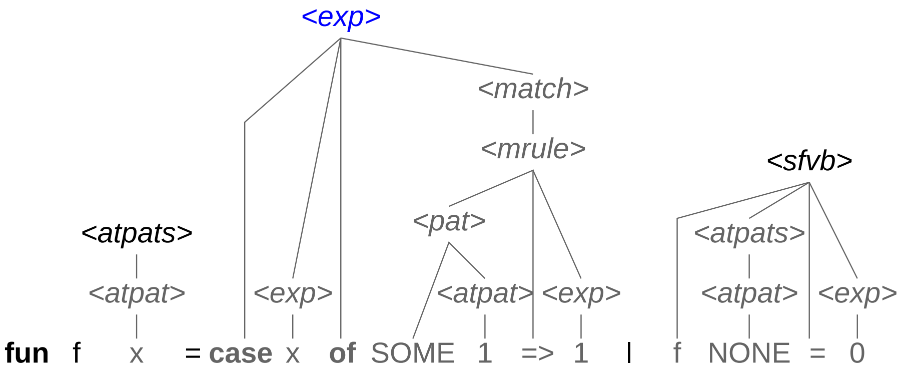


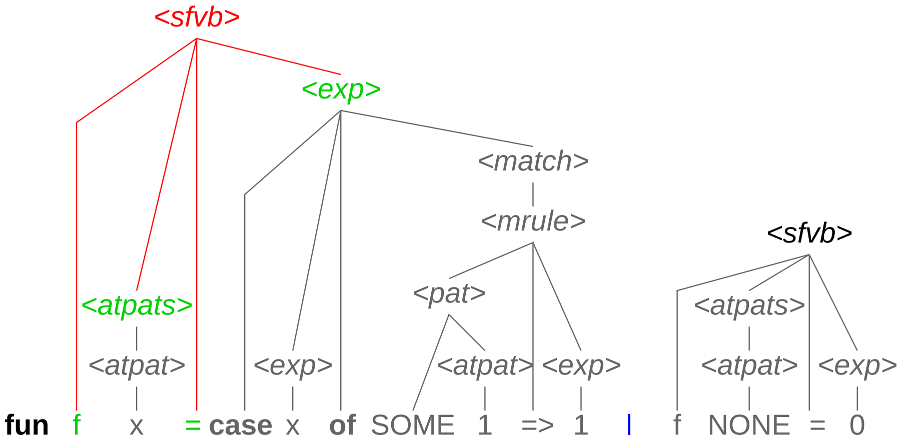
<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpats>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0


<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpats>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0

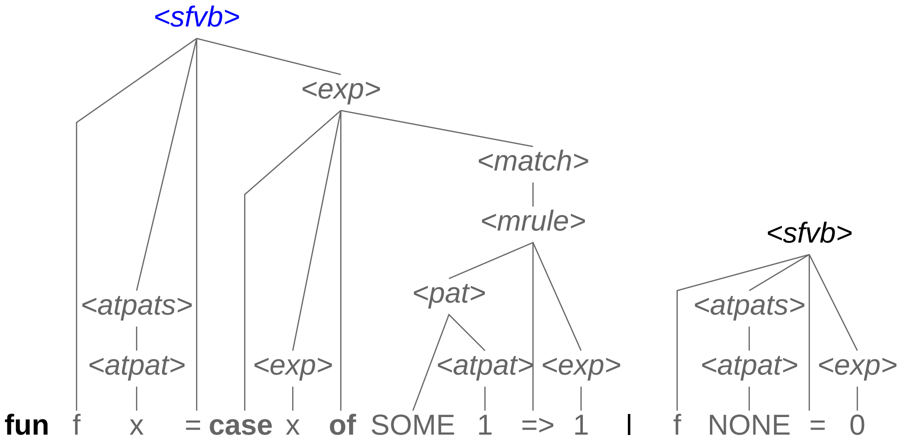

<match>
<mrule>
<sfvb>
<pat>
<atpats>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0

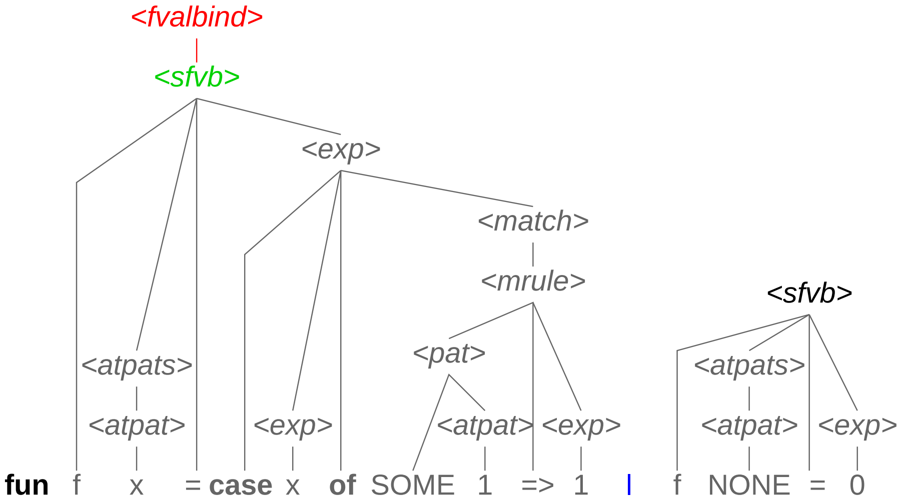

<exp>
<match>
<mrule>
<sfvb>
<atpats>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0


<exp>
<match>
<mrule>
<sfvb>
<pat>
<atpats>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0

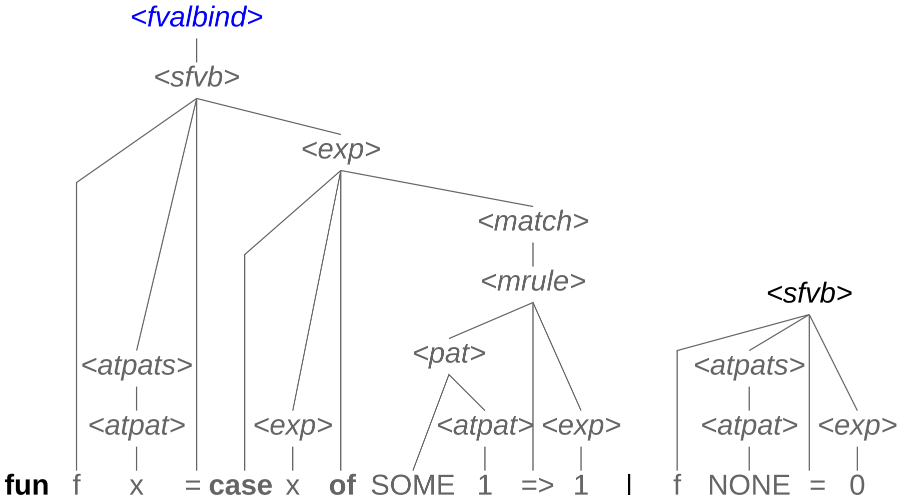


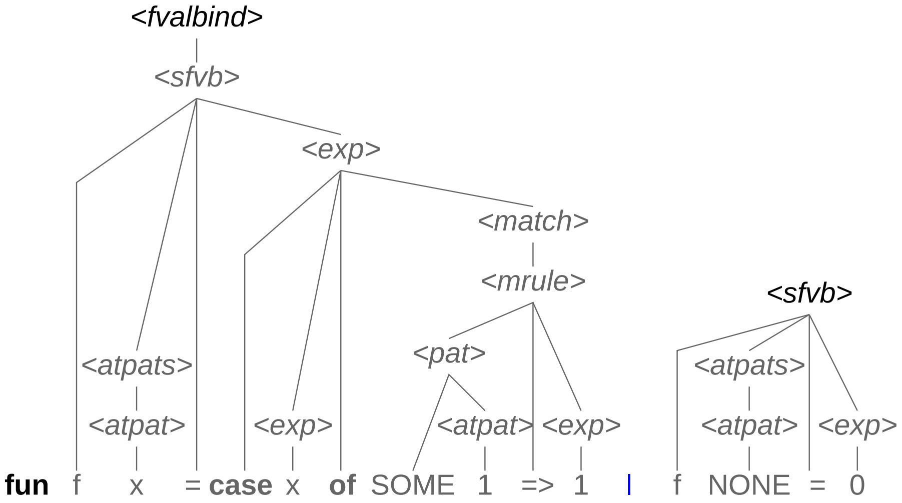
<sfvb>
<exp>
<match>
<mrule>
<sfvb>
<pat>
<atpats>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0


<sfvb>
<exp>
<match>
<mrule>
<sfvb>
<pat>
<atpats>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0

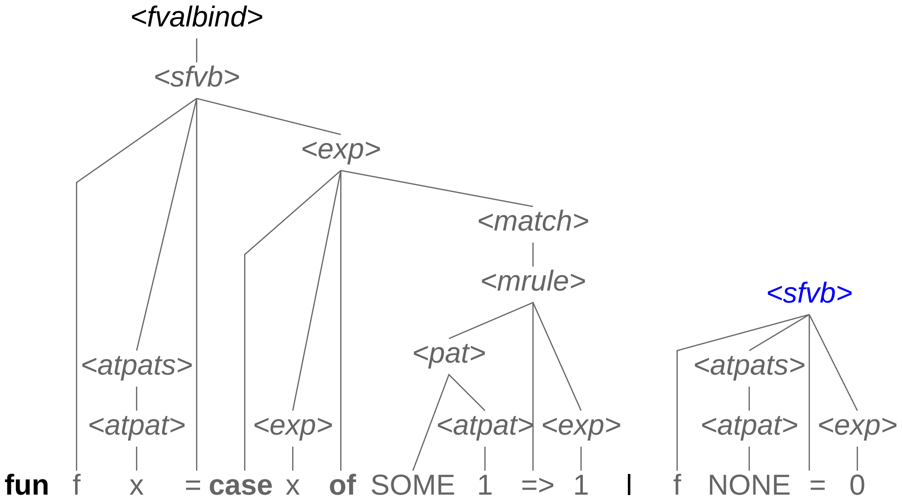


<fvalbind>
<sfvb>
<exp>
<match>
<mrule>
<sfvb>
<pat>
<atpats>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0

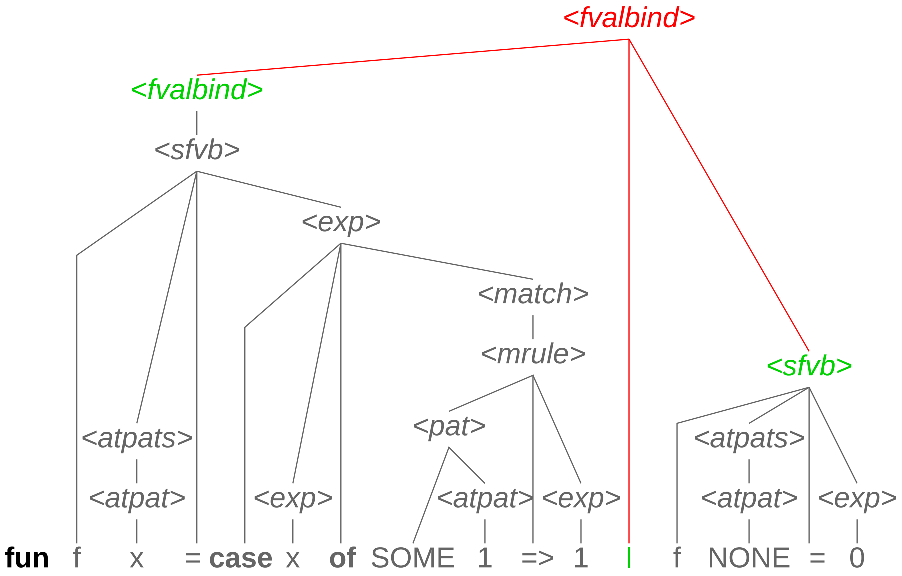

<fvalbind>
<sfvb>
<exp>
<match>
<mrule>
<sfvb>
<pat>
<atpats>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0

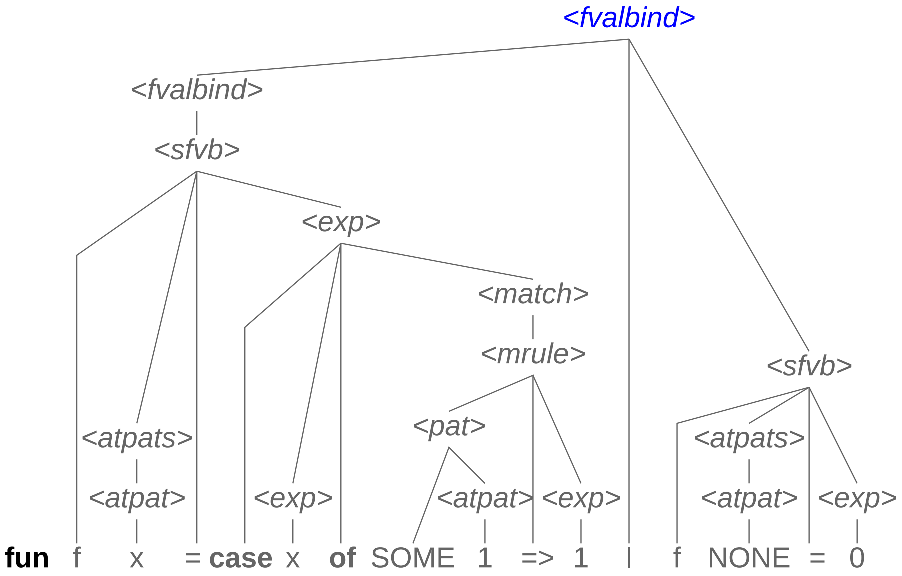


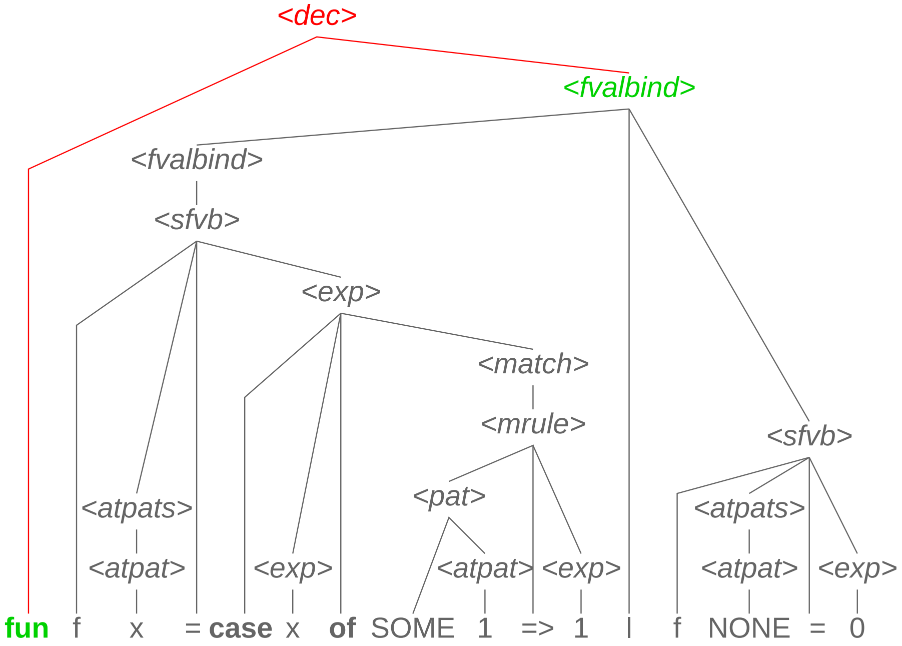
<fvalbind>
<sfvb>
<exp>
<match>
<mrule>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<sfvb>
<atpats>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0

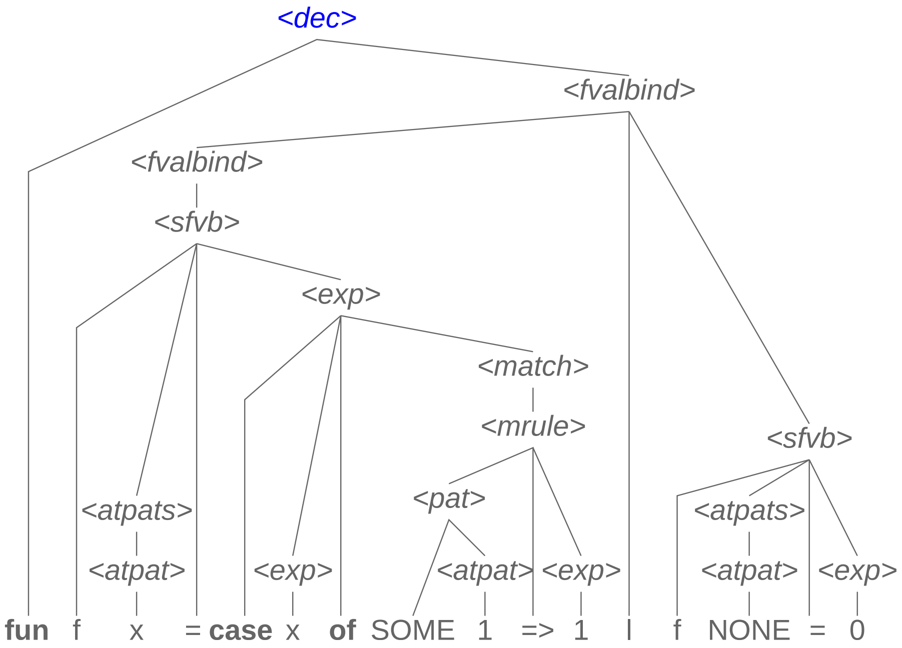


<fvalbind>
<sfvb>
<exp>
<match>
<mrule>
<sfvb>
<pat>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpats>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0


<fvalbind>
<fvalbind>
<sfvb>
<exp>
<match>
<mrule>
<sfvb>
<pat>
<atpats>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0


<fvalbind>
<fvalbind>
<sfvb>
<exp>
<match>
<mrule>
<sfvb>
<pat>
<atpats>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0


<dec>
<fvalbind>
<fvalbind>
<sfvb>
<exp>
<match>
<mrule>
<sfvb>
<pat>
<atpats>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0

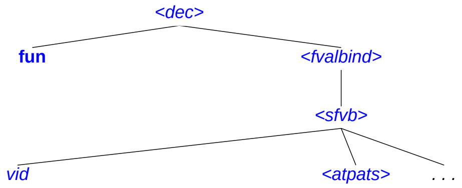


<dec>
<fvalbind>
<fvalbind>
<sfvb>
<exp>
<match>
<mrule>
<sfvb>
<pat>
<atpats>
<atpats>
<atpat>
<exp>
<atpat>
<exp>
<atpat>
<exp>
fun f x = case x of SOME 1 => 1 | f NONE = 0

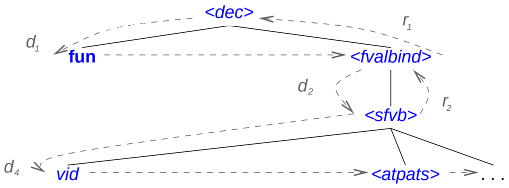
Generating the Parser

1. nondeterministic automaton
2. determinization by subset construction

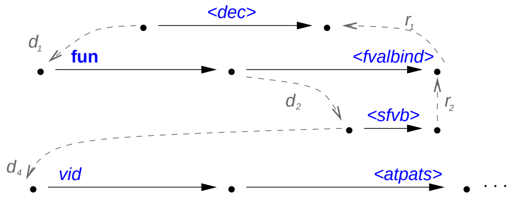
Position Graph



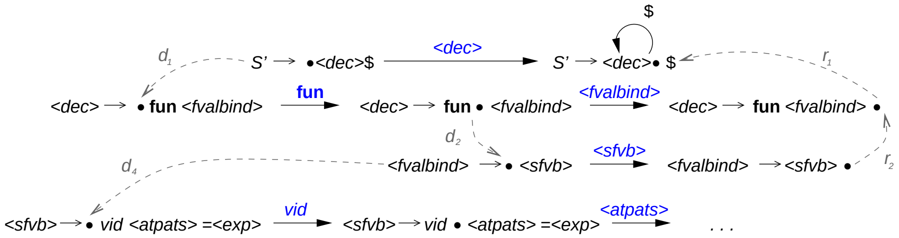
Position Graph



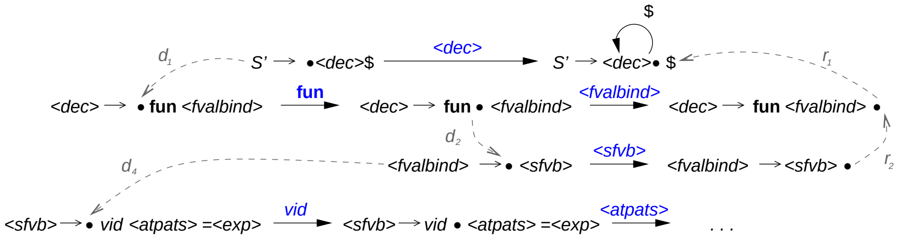
Position Graph



Position Graph



Position Graph



(Potentially) Infinite graph of all derivations trees.

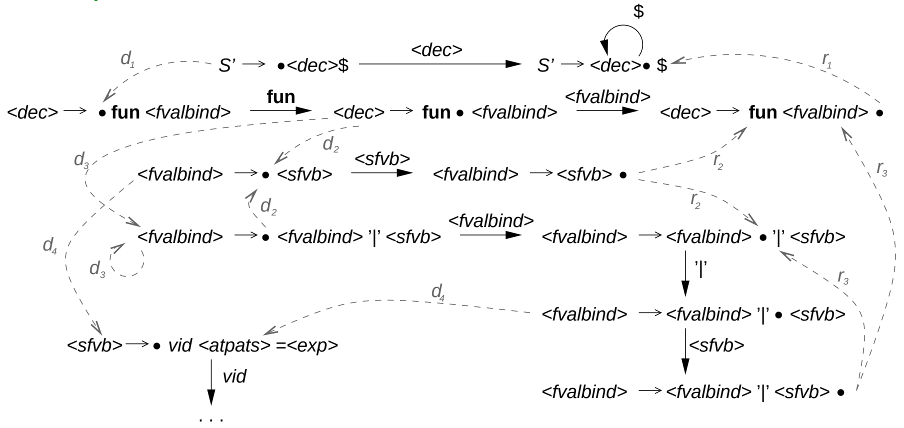
Nondeterministic Automaton

Apply an equivalence relation on the nodes of the position graph.

Nondeterministic Automaton

Apply an equivalence relation on the nodes of the position graph.

Example κ_0, same dotted label



Subset Construction

- d_i transitions denote traditional item closures
- r_i transitions denote a phrase that should be reduced
- other transitions denote shifts
- items in the construction hold
  1. a state of the nondeterministic automaton
  2. a parsing action
  3. a pushback length

Subset Construction

- d_i transitions denote traditional item closures
- r_i transitions denote a phrase that should be reduced
- other transitions denote shifts
- items in the construction hold
  1. a state of the nondeterministic automaton
  2. a parsing action
  3. a pushback length

Subset Construction

$$\begin{aligned} \textit{<exp>} &\longrightarrow \textbf{case}\ \textit{<exp>}\ \ \textbf{of}\ \ \textit{<match>} \bullet \\ \textit{<match>} &\longrightarrow \textit{<match>} \bullet \text{'|'}\ \textit{<mrule>} \end{aligned}$$

Subset Construction

$$\begin{tikzcd}
\langle exp\rangle \longrightarrow \textbf{case}\ \langle exp\rangle\ \textbf{of}\ \langle match\rangle \bullet \arrow[dd, dashed, bend right, "r_5"'] \\
\langle match\rangle \longrightarrow \langle match\rangle \bullet \text{'|'}\ \langle mrule\rangle \\
\langle sfvb\rangle \longrightarrow vid\ \langle atpats\rangle = \langle exp\rangle \bullet, 5, 0
\end{tikzcd}$$

Subset Construction

$$\begin{array}{rl}
\textit{<exp>} \longrightarrow & \textbf{case}\ \textit{<exp>}\ \ \textbf{of}\ \textit{<match>} \bullet \\
\textit{<match>} \longrightarrow & \textit{<match>} \bullet \text{'|'}\ \textit{<mrule>} \\
r \dashleftarrow \textit{<sfvb>} \longrightarrow & \textit{vid}\ \textit{<atpats>} = \textit{<exp>}\ \bullet, 5, 0 \\
\searrow \textit{<fvalbind>} \longrightarrow & \textit{<fvalbind>}\ \text{'|'}\ \textit{<sfvb>} \bullet, 5, 0
\end{array}$$

Subset Construction

$$\begin{tikzcd}[row sep=small]
\textit{<exp>} \longrightarrow \textbf{case}\ \textit{<exp>}\ \textbf{of}\ \textit{<match>}\ \bullet \\
\textit{<match>} \longrightarrow \textit{<match>}\ \bullet\ \text{'|'}\ \textit{<mrule>} \\
\textit{<sfvb>} \longrightarrow \textit{vid}\ \textit{<atpats>}\ =\textit{<exp>}\ \bullet, 5, 0 \arrow[dd, dashed, bend right, "r"'] \\
\textit{<fvalbind>} \longrightarrow \textit{<fvalbind>}\ \text{'|'}\ \textit{<sfvb>}\ \bullet, 5, 0 \\
\textit{<fvalbind>} \longrightarrow \textit{<sfvb>}\ \bullet, 5, 0
\end{tikzcd}$$

Subset Construction

$$\begin{tikzcd}[row sep=small]
\textit{<exp>} \longrightarrow \textbf{case}\ \textit{<exp>}\ \textbf{of}\ \textit{<match>} \bullet \\
\textit{<match>} \longrightarrow \textit{<match>} \bullet \text{'|'}\ \textit{<mrule>} \\
\textit{<sfvb>} \longrightarrow \textit{vid}\ \textit{<atpats>} = \textit{<exp>} \bullet, 5, 0 \\
\textit{<fvalbind>} \longrightarrow \textit{<fvalbind>}\ \text{'|'}\ \textit{<sfvb>} \bullet, 5, 0 \arrow[dd, dashed, bend right] \arrow[ddd, dashed, bend right=60, "r"'] \\
\textit{<fvalbind>} \longrightarrow \textit{<sfvb>} \bullet, 5, 0 \\
\textit{<fvalbind>} \longrightarrow \textit{<fvalbind>} \bullet \text{'|'}\ \textit{<sfvb>}, 5, 0 \\
\textit{<dec>} \longrightarrow \textbf{fun}\ \textit{<fvalbind>} \bullet, 5, 0
\end{tikzcd}$$

Subset Construction

<exp> ⟶ **case** *<exp>* **of** *<match>* •

<match> ⟶ *<match>* • '|' *<mrule>*

<sfvb> ⟶ *vid <atpats>* =*<exp>* •, 5, 0

<fvalbind> ⟶ *<fvalbind>* '|' *<sfvb>* •, 5, 0

<fvalbind> ⟶ *<sfvb>* •, 5, 0

<fvalbind> ⟶ *<fvalbind>* • '|' *<sfvb>*, 5, 0

<dec> ⟶ **fun** *<fvalbind>* •, 5, 0

r ⇢ *S'* ⟶ *<dec>* • $, 5, 0

Subset Construction

<exp> $\longrightarrow$ **case** *<exp>* **of** *<match>* •
<match> $\longrightarrow$ *<match>* • '|' *<mrule>*
<sfvb> $\longrightarrow$ *vid <atpats> =<exp>* •, 5, 0
<fvalbind> $\longrightarrow$ *<fvalbind>* '|' *<sfvb>* •, 5, 0
<fvalbind> $\longrightarrow$ *<sfvb>* •, 5, 0
<fvalbind> $\longrightarrow$ *<fvalbind>* • '|' *<sfvb>*, 5, 0
<dec> $\longrightarrow$ **fun** *<fvalbind>* •, 5, 0
S' $\longrightarrow$ *<dec>* • $, 5, 0

Subset Construction

<exp> ⟶ **case** *<exp>* **of** *<match>* •
<match> ⟶ *<match>* • '|' *<mrule>*
<sfvb> ⟶ *vid <atpats>* =*<exp>* •, 5, 0
<fvalbind> ⟶ *<fvalbind>* '|' *<sfvb>* •, 5, 0
<fvalbind> ⟶ *<sfvb>* •, 5, 0
<fvalbind> ⟶ *<fvalbind>* • '|' *<sfvb>*, 5, 0
<dec> ⟶ **fun** *<fvalbind>* •, 5, 0
S' ⟶ *<dec>* • $, 5, 0

Subset Construction

<exp> ⟶ **case** *<exp>* **of** *<match>* •

<match> ⟶ *<match>* • '|' *<mrule>*

<sfvb> ⟶ *vid <atpats>* =*<exp>* •, 5, 0

<fvalbind> ⟶ *<fvalbind>* '|' *<sfvb>* •, 5, 0

<fvalbind> ⟶ *<sfvb>* •, 5, 0

<fvalbind> ⟶ *<fvalbind>* • '|' *<sfvb>*, 5, 0

<dec> ⟶ **fun** *<fvalbind>* •, 5, 0

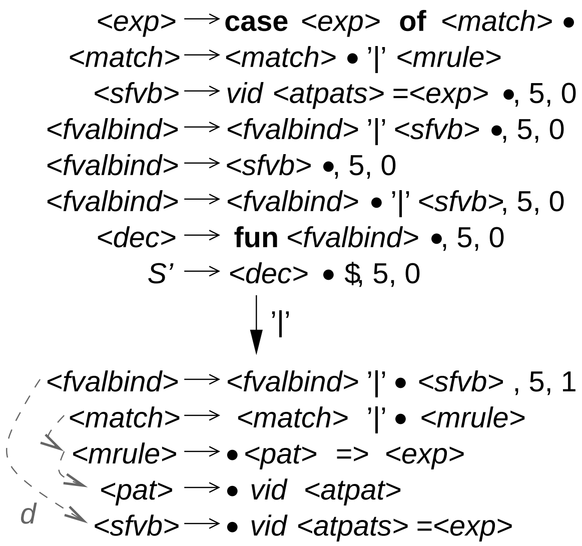
S' ⟶ *<dec>* • $, 5, 0

$$\Big\downarrow \text{'|'}$$

<fvalbind> ⟶ *<fvalbind>* '|' • *<sfvb>* , 5, 1

<match> ⟶ *<match>* '|' • *<mrule>*

Subset Construction



<exp> ⟶ **case** *<exp>* **of** *<match>* •
<match> ⟶ *<match>* • '|' *<mrule>*
<sfvb> ⟶ *vid <atpats>* =*<exp>* •, 5, 0
<fvalbind> ⟶ *<fvalbind>* '|' *<sfvb>* •, 5, 0
<fvalbind> ⟶ *<sfvb>* •, 5, 0
<fvalbind> ⟶ *<fvalbind>* • '|' *<sfvb>*, 5, 0
<dec> ⟶ **fun** *<fvalbind>* •, 5, 0
S' ⟶ *<dec>* • $, 5, 0

$\downarrow$ '|'

<fvalbind> ⟶ *<fvalbind>* '|' • *<sfvb>* , 5, 1
<match> ⟶ *<match>* '|' • *<mrule>*
<mrule> ⟶ • *<pat>* => *<exp>*
<pat> ⟶ • *vid <atpat>*
<sfvb> ⟶ • *vid <atpats>* =*<exp>*

Construction Failure

$<$*exp*$>$ $\longrightarrow$ **case** $<$*exp*$>$ **of** $<$*match*$>$ $\bullet$
$<$*match*$>$ $\longrightarrow$ $<$*match*$>$ $\bullet$ '|' $<$*mrule*$>$
$<$*sfvb*$>$ $\longrightarrow$ *vid* $<$*atpats*$>$ =$<$*exp*$>$ $\bullet$, 5, 0
$<$*fvalbind*$>$ $\longrightarrow$ $<$*fvalbind*$>$ '|' $<$*sfvb*$>$ $\bullet$, 5, 0
$<$*fvalbind*$>$ $\longrightarrow$ $<$*sfvb*$>$ $\bullet$, 5, 0
$<$*fvalbind*$>$ $\longrightarrow$ $<$*fvalbind*$>$ $\bullet$ '|' $<$*sfvb*$>$, 5, 0
$<$*dec*$>$ $\longrightarrow$ **fun** $<$*fvalbind*$>$ $\bullet$, 5, 0
S' $\longrightarrow$ $<$*dec*$>$ $\bullet$ \$, 5, 0

Construction Failure

<exp> ⟶ **case** *<exp>* **of** *<match>* •

<match> ⟶ *<match>* • '|' *<mrule>*

<sfvb> ⟶ *vid <atpats>* =*<exp>* •, 5, 0

<fvalbind> ⟶ *<fvalbind>* '|' *<sfvb>* •, 5, 0

<fvalbind> ⟶ *<sfvb>* •, 5, 0

<fvalbind> ⟶ *<fvalbind>* • '|' *<sfvb>*, 5, 0

<dec> ⟶ **fun** *<fvalbind>* •, 5, 0

r_5 *S'* ⟶ *<dec>* • $, 5, 0

<mrule> ⟶ *<pat>* => *<exp>* •, 5, 0

Construction Failure

<exp> ⟶ **case** *<exp>* **of** *<match>* •

<match> ⟶ *<match>* • '|' *<mrule>*

<sfvb> ⟶ *vid <atpats>* =*<exp>* •, 5, 0

<fvalbind> ⟶ *<fvalbind>* '|' *<sfvb>* •, 5, 0

<fvalbind> ⟶ *<sfvb>* •, 5, 0

<fvalbind> ⟶ *<fvalbind>* • '|' *<sfvb>*, 5, 0

<dec> ⟶ **fun** *<fvalbind>* •, 5, 0

S' ⟶ *<dec>* • $, 5, 0

r *<mrule>* ⟶ *<pat>* => *<exp>* •, 5, 0

<match> ⟶ *<mrule>* •, 5, 0

Construction Failure

<exp> → **case** *<exp>* **of** *<match>* •

<match> → *<match>* • '|' *<mrule>*

<sfvb> → *vid <atpats>* =*<exp>* •, 5, 0

<fvalbind> → *<fvalbind>* '|' *<sfvb>* •, 5, 0

<fvalbind> → *<sfvb>* •, 5, 0

<fvalbind> → *<fvalbind>* • '|' *<sfvb>*, 5, 0

<dec> → **fun** *<fvalbind>* •, 5, 0

S' → *<dec>* • $, 5, 0

<mrule> → *<pat>* => *<exp>* •, 5, 0

<match> → *<mrule>* •, 5, 0

r

<match> → *<match>* • '|' *<mrule>*, 5, 0

Handling Ambiguity

Longest match rule:

```
case (pred x)
  of SOME y => case y
    of pattern => filterP(r, y::l)
   | NONE    => filterP(r, l)
```

Differentiate two instances of the same shift/reduce conflict!

Handling Ambiguity

Longest match rule:

```
case (pred x)
  of SOME y => case y
    of pattern => filterP(r, y::l)
     | NONE    => filterP(r, l)
```

Differentiate two instances of the same shift/reduce conflict!

Handling Ambiguity

Longest match rule:

```
case (pred x)
  of SOME y => case y
    of pattern => filterP(r, y::l)
     | NONE    => filterP(r, l)
```

Differentiate two instances of the same shift/reduce conflict!

Handling Ambiguity

$$<exp> \longrightarrow \textbf{case } <exp> \textbf{ of } <match> \bullet$$
$$<match> \longrightarrow <match> \bullet \text{'|'} <mrule>$$
$$<sfvb> \longrightarrow vid <atpats> =<exp> \bullet, 5, 0$$
$$<fvalbind> \longrightarrow <fvalbind> \text{'|'} <sfvb> \bullet, 5, 0$$
$$<fvalbind> \longrightarrow <sfvb> \bullet, 5, 0$$
$$<fvalbind> \longrightarrow <fvalbind> \bullet \text{'|'} <sfvb>, 5, 0$$
$$<dec> \longrightarrow \textbf{fun } <fvalbind> \bullet, 5, 0$$
$$S' \longrightarrow <dec> \bullet \$, 5, 0$$
$$<mrule> \longrightarrow <pat> => <exp> \bullet, 5, 0$$
$$<match> \longrightarrow <mrule> \bullet, 5, 0$$
$$<match> \longrightarrow <match> \bullet \text{'|'} <mrule>, 5, 0$$

Complexity

- $|\Gamma|$: size of the nondeterministic automaton
- $|\mathcal{A}|$: size of the parser: $\mathcal{O}(2^{|\Gamma|\,|P|})$
- parsing time complexity for input w: $\mathcal{O}(|w|)$

Complexity

- $|\Gamma|$: size of the nondeterministic automaton
  $|\Gamma_{\kappa_0}| = \mathcal{O}(|\mathcal{G}|\ |P|)$
- $|\mathcal{A}|$: size of the parser: $\mathcal{O}(2^{|\Gamma|\ |P|})$
- parsing time complexity for input w: $\mathcal{O}(|w|)$

Complexity

- $|\Gamma|$: size of the nondeterministic automaton
- $|\mathcal{A}|$: size of the parser: $\mathcal{O}(2^{|\Gamma|\,|P|})$
- parsing time complexity for input w: $\mathcal{O}(|w|)$

Complexity

- $|\Gamma|$: size of the nondeterministic automaton
- $|\mathcal{A}|$: size of the parser: $\mathcal{O}(2^{|\Gamma|\,|P|})$
- parsing time complexity for input w: $\mathcal{O}(|w|)$

Limitations

– incomparable with classical parsing techniques

+ subset construction amendable

Limitations

− incomparable with classical parsing techniques

+ subset construction amendable

Ending Comments

- Shift Resolve parsers
  1. Large class of grammars accepted
  2. Unambiguity
  3. Linear time parsing
- 2-steps construction
  1. Simple
  2. Flexible

Ending Comments

- Shift Resolve parsers
  1. Large class of grammars accepted
  2. Unambiguity
  3. Linear time parsing
- 2-steps construction
  1. Simple
  2. Flexible

Ending Comments

- Shift Resolve parsers
  1. Large class of grammars accepted
  2. Unambiguity
  3. Linear time parsing
- 2-steps construction
  1. Simple
  2. Flexible

Ending Comments

- Shift Resolve parsers
  1. Large class of grammars accepted
  2. Unambiguity
  3. Linear time parsing
- 2-steps construction
  1. Simple
  2. Flexible

M. E. Bermudez and K. M. Schimpf. Practical arbitrary lookahead LR parsing. J. Comput. Syst. Sci., 41(2):230–250, 1990. ISSN 0022-0000. doi: 10.1016/0022-0000(90)90037-L.

J. Farré and J. Fortes Gálvez. A bounded-connect construction for LR-Regular parsers. In R. Wilhelm, editor, CC'01, volume 2027 of Lecture Notes in Computer Science, pages 244–258. Springer, 2001. URL `http://www.springerlink.com/link.asp?id=e3e8g77kxevkyjfd`.

J. Farré and J. Fortes Gálvez. Bounded-connect noncanonical discriminating-reverse parsers. Theoretical Comput. Sci., 313(1): 73–91, 2004. ISSN 0304-3975. doi: 10.1016/j.tcs.2003.10.006.

R. Milner, M. Tofte, R. Harper, and D. MacQueen. The definition of Standard ML. MIT Press, revised edition, 1997. ISBN 0-262-63181-4.

T. J. Parr and R. W. Quong. ANTLR: A predicated-LL(k) parser generator. Software: Practice & Experience, 25(7):789–810, 1995. ISSN 0038-0644. URL `http://citeseer.nj.nec.com/12770`.

S. Schmitz. Noncanonical LALR(1) parsing. In Z. Dang and O. H. Ibarra, editors, DLT'06, volume 4036 of Lecture Notes in Computer Science, pages 95–107. Springer, 2006. ISBN 3-540-35428-X. doi: 10.1007/11779148_10.

K.-C. Tai. Noncanonical SLR(1) grammars. ACM Trans. Prog. Lang. Syst., 1(2):295–320, 1979. ISSN 0164-0925. doi: 10.1145/357073.357083.

M. Tomita. Efficient Parsing for Natural Language. Kluwer Academic Publishers, 1986. ISBN 0-89838-202-5.