Embedding Generalized Parsing in Haskell

Jaro Reinders

Why parser combinators?

For the parser generator crowd

Why parser combinators?

For the parser generator crowd

- Language integration

Why parser combinators?

For the parser generator crowd

- Language integration
- Directly parse into a semantic value

```
numbers = (+) <$> number <* char ‘ ‘ <*> numbers <|> pure 0

> parse numbers “2 31 9”
Just 42
```

Why parser combinators?

For the parser generator crowd

- Language integration
- Directly parse into a semantic value

```
numbers = (+) <$> number <* char ‘ ‘ <*> numbers <|> pure 0

> parse numbers “2 31 9”
Just 42
```

- Abstract over common patterns in your grammar

```
maybe p = Just <$> p <|> pure Nothing
```

Why parser combinators?

For the parser generator crowd

- Language integration
- Directly parse into a semantic value

```
numbers = (+) <$> number <* char ‘ ‘ <*> numbers <|> pure 0

> parse numbers “2 31 9”
Just 42
```

- Abstract over common patterns in your grammar

```
maybe p = Just <$> p <|> pure Nothing
```

- Monadic parsers enable data-dependent disambiguation (1)

```
ndots 0 = pure ()
ndots n = char ‘.’ *> ndots (n - 1)

> parse (number >>= ndots) “5....."
Just ()
```

Why generalized parsing?

For the parser combinator crowd

Why generalized parsing?

For the parser combinator crowd

- Left-recursion

$$\begin{aligned}\mathrm{D} &::= 0 \mid 1 \mid \ldots \mid 9 \\ \mathrm{N} &::= \mathrm{N}\ \mathrm{D} \mid \mathrm{D}\end{aligned}$$

Why generalized parsing?

For the parser combinator crowd

- Left-recursion

$$\begin{aligned}&\mathrm{D} ::= 0 \mid 1 \mid \ldots \mid 9\\ &\mathrm{N} ::= \mathrm{N}\ \mathrm{D} \mid \mathrm{D}\end{aligned}$$

- Compositionality

“… it can be quite difficult to determine what language is defined by a TDPL program.” ~ Aho and Ullman (2, p466)

Why generalized parsing?

For the parser combinator crowd

- Left-recursion

$$\begin{array}{l} \mathrm{D} ::= 0 \mid 1 \mid \ldots \mid 9 \\ \mathrm{N} ::= \mathrm{N}\ \mathrm{D} \mid \mathrm{D} \end{array}$$

- Compositionality

  "… it can be quite difficult to determine what language is defined by a TDPL program." ~ Aho and Ullman (2, p466)

- Disambiguation through annotation rather than deformation

Aside: eliminating left-recursion

“Can’t we just…”

- $\mathrm{N} ::= \mathrm{N}\ \mathrm{D} \mid \mathrm{D}$
  becomes
  $\mathrm{N} ::= \mathrm{D}\ \mathrm{N'}$
  $\mathrm{N'} ::= \mathrm{D}\ \mathrm{N'} \mid \varepsilon$
- Complicated for hidden left recursion and semantic values
- Grammar size can grow exponentially

GLL (3)

- Slots, Extended Packed Nodes, Descriptors, Commencements, Continuations
- Essentially building up big set of intermediate results
- $O(n^3)$ time and space

Generalized Parser Combinators

Generalized Parser Combinators

- Partial normalization up front (Free MonadPlus)

Generalized Parser Combinators

- Partial normalization up front (Free MonadPlus)
- Then simple driver

Generalized Parser Combinators

- Partial normalization up front (Free MonadPlus)
- Then simple driver
- Stack of continuations

Generalized Parser Combinators

- Partial normalization up front (Free MonadPlus)
- Then simple driver
- Stack of continuations
- Actions
  - Descend (Push)
  - Loop (Append)
  - Continue (Read)
  - Ascend (Pop)

Stack and continuations

Stack and continuations

- Stack entries consist of

Stack and continuations

- Stack entries consist of
    - nonterminal name and pivot

Stack and continuations

- Stack entries consist of
  - nonterminal name and pivot
  - list of loop continuations

Stack and continuations

- Stack entries consist of
  - nonterminal name and pivot
  - list of loop continuations
  - continuation parser

Stack and continuations

- Stack entries consist of
  - nonterminal name and pivot
  - list of loop continuations
  - continuation parser
- Loop continuations consist of

Stack and continuations

- Stack entries consist of
    - nonterminal name and pivot
    - list of loop continuations
    - continuation parser
- Loop continuations consist of
    - stack slice

Stack and continuations

- Stack entries consist of
  - nonterminal name and pivot
  - list of loop continuations
  - continuation parser
- Loop continuations consist of
  - stack slice
  - parser

Descend & Loop

Descend & Loop

- When a nonterminal is encountered

Descend & Loop

- When a nonterminal is encountered
- If it is **not** on the stack at the current pivot: Descend

Descend & Loop

- When a nonterminal is encountered
- If it is **not** on the stack at the current pivot: Descend
  - Push it to the stack with an empty list of loop continuations and the current continuation

Descend & Loop

- When a nonterminal is encountered
- If it is **not** on the stack at the current pivot: Descend
    - Push it to the stack with an empty list of loop continuations and the current continuation
    - Enter the right hand side

Descend & Loop

- When a nonterminal is encountered
- If it is **not** on the stack at the current pivot: Descend
  - Push it to the stack with an empty list of loop continuations and the current continuation
  - Enter the right hand side
- If it is on the stack at the current pivot: Loop

Descend & Loop

- When a nonterminal is encountered
- If it is **not** on the stack at the current pivot: Descend
    - Push it to the stack with an empty list of loop continuations and the current continuation
    - Enter the right hand side
- If it is on the stack at the current pivot: Loop
    - Capture a slice up to that occurrence

Descend & Loop

- When a nonterminal is encountered
- If it is **not** on the stack at the current pivot: Descend
  - Push it to the stack with an empty list of loop continuations and the current continuation
  - Enter the right hand side
- If it is on the stack at the current pivot: Loop
  - Capture a slice up to that occurrence
  - Append that slice along and current continuation to the loop continuations

Descend & Loop

- When a nonterminal is encountered
- If it is **not** on the stack at the current pivot: Descend
  - Push it to the stack with an empty list of loop continuations and the current continuation
  - Enter the right hand side
- If it is on the stack at the current pivot: Loop
  - Capture a slice up to that occurrence
  - Append that slice along and current continuation to the loop continuations
  - Bail out

Continue & Ascend

Continue & Ascend

- When a nonterminal is fully parsed, both

Continue & Ascend

- When a nonterminal is fully parsed, both
- Continue

Continue & Ascend

- When a nonterminal is fully parsed, both
- Continue
  - Peek at the list of loop continuations

Continue & Ascend

- When a nonterminal is fully parsed, both
- Continue
  - Peek at the list of loop continuations
  - Choose one

Continue & Ascend

- When a nonterminal is fully parsed, both
- Continue
  - Peek at the list of loop continuations
  - Choose one
  - Append its slice to the current stack

Continue & Ascend

- When a nonterminal is fully parsed, both
- Continue
    - Peek at the list of loop continuations
    - Choose one
    - Append its slice to the current stack
    - Continue with its parser

Continue & Ascend

- When a nonterminal is fully parsed, both
- Continue
  - Peek at the list of loop continuations
  - Choose one
  - Append its slice to the current stack
  - Continue with its parser
- Ascend

Continue & Ascend

- When a nonterminal is fully parsed, both
- Continue
  - Peek at the list of loop continuations
  - Choose one
  - Append its slice to the current stack
  - Continue with its parser
- Ascend
  - Pop the stack

Continue & Ascend

- When a nonterminal is fully parsed, both
- Continue
    - Peek at the list of loop continuations
    - Choose one
    - Append its slice to the current stack
    - Continue with its parser
- Ascend
    - Pop the stack
    - Continue with the final continuation

Parsing $5 + 3 + 7$ with $\mathrm{X} ::= \mathrm{X} + \mathrm{X} \mid \mathbb{N}$

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$X + X \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+\ X]$

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$X + X \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+ X]$
$X_0[+ X]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$X + X \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+ X]$
$X_0[+ X]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
$X_0[+ X]$	ε	$+ 3 + 7$	continue $[+ X]$

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$X + X \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+\ X]$
$X_0[+\ X]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
$X_0[+\ X]$	ε	$+ 3 + 7$	continue $[+\ X]$
$X_0[+\ X]$	$+\ X$	$+ 3 + 7$	parse $+$

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$X + X \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+ X]$
$X_0[+ X]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
$X_0[+ X]$	ε	$+ 3 + 7$	continue $[+ X]$
$X_0[+ X]$	$+ X$	$+ 3 + 7$	parse $+$
$X_0[+ X]$	X	$3 + 7$	descend X

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$X + X \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+ X]$
$X_0[+ X]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
$X_0[+ X]$	ε	$+ 3 + 7$	continue $[+ X]$
$X_0[+ X]$	$+ X$	$+ 3 + 7$	parse $+$
$X_0[+ X]$	X	$3 + 7$	descend X
$X_0[+ X];X_2$	$X + X \mid \mathbb{N}$	$3 + 7$	loop X, $[+ X]$

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$X + X \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+ X]$
$X_0[+ X]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
$X_0[+ X]$	ε	$+ 3 + 7$	continue $[+ X]$
$X_0[+ X]$	$+ X$	$+ 3 + 7$	parse $+$
$X_0[+ X]$	X	$3 + 7$	descend X
$X_0[+ X];X_2$	$X + X \mid \mathbb{N}$	$3 + 7$	loop X, $[+ X]$
$X_0[+ X];X_2[+ X]$	$\mathbb{N}$	$3 + 7$	parse $\mathbb{N}$

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$X + X \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+ X]$
$X_0[+ X]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
$X_0[+ X]$	ε	$+ 3 + 7$	continue $[+ X]$
$X_0[+ X]$	$+ X$	$+ 3 + 7$	parse $+$
$X_0[+ X]$	X	$3 + 7$	descend X
$X_0[+ X];X_2$	$X + X \mid \mathbb{N}$	$3 + 7$	loop X, $[+ X]$
$X_0[+ X];X_2[+ X]$	$\mathbb{N}$	$3 + 7$	parse $\mathbb{N}$
$X_0[+ X];X_2[+ X]$	ε	$+ 7$	continue $[+ X]$

Parsing $5 + 3 + 7$ with $\mathrm{X} ::= \mathrm{X} + \mathrm{X} \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$\mathrm{X} + \mathrm{X} \mid \mathbb{N}$	$5 + 3 + 7$	loop X, [+ X]
$\mathrm{X}_0[+\ \mathrm{X}]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
$\mathrm{X}_0[+\ \mathrm{X}]$	ε	$+\ 3 + 7$	continue [+ X]
$\mathrm{X}_0[+\ \mathrm{X}]$	$+\ \mathrm{X}$	$+\ 3 + 7$	parse +
$\mathrm{X}_0[+\ \mathrm{X}]$	X	$3 + 7$	descend X
$\mathrm{X}_0[+\ \mathrm{X}];\mathrm{X}_2$	$\mathrm{X} + \mathrm{X} \mid \mathbb{N}$	$3 + 7$	loop X, [+ X]
$\mathrm{X}_0[+\ \mathrm{X}];\mathrm{X}_2[+\ \mathrm{X}]$	$\mathbb{N}$	$3 + 7$	parse $\mathbb{N}$
$\mathrm{X}_0[+\ \mathrm{X}];\mathrm{X}_2[+\ \mathrm{X}]$	ε	$+\ 7$	continue [+ X]
$\mathrm{X}_0[+\ \mathrm{X}];\mathrm{X}_2[+\ \mathrm{X}]$	$+\ \mathrm{X}$	$+\ 7$	parse +

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$X + X \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+\ X]$
$X_0[+\ X]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
$X_0[+\ X]$	ε	$+ 3 + 7$	continue $[+\ X]$
$X_0[+\ X]$	$+\ X$	$+ 3 + 7$	parse $+$
$X_0[+\ X]$	X	$3 + 7$	descend X
$X_0[+\ X];X_2$	$X + X \mid \mathbb{N}$	$3 + 7$	loop X, $[+\ X]$
$X_0[+\ X];X_2[+\ X]$	$\mathbb{N}$	$3 + 7$	parse $\mathbb{N}$
$X_0[+\ X];X_2[+\ X]$	ε	$+ 7$	continue $[+\ X]$
$X_0[+\ X];X_2[+\ X]$	$+\ X$	$+ 7$	parse $+$
$X_0[+\ X];X_2[+\ X]$	X	7	descend X

Parsing $5 + 3 + 7$ with $\mathrm{X} ::= \mathrm{X} + \mathrm{X} \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$\mathrm{X} + \mathrm{X} \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+\ \mathrm{X}]$
$\mathrm{X}_0[+\ \mathrm{X}]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
$\mathrm{X}_0[+\ \mathrm{X}]$	ε	$+ 3 + 7$	continue $[+\ \mathrm{X}]$
$\mathrm{X}_0[+\ \mathrm{X}]$	$+\ \mathrm{X}$	$+ 3 + 7$	parse $+$
$\mathrm{X}_0[+\ \mathrm{X}]$	X	$3 + 7$	descend X
$\mathrm{X}_0[+\ \mathrm{X}];\mathrm{X}_2$	$\mathrm{X} + \mathrm{X} \mid \mathbb{N}$	$3 + 7$	loop X, $[+\ \mathrm{X}]$
$\mathrm{X}_0[+\ \mathrm{X}];\mathrm{X}_2[+\ \mathrm{X}]$	$\mathbb{N}$	$3 + 7$	parse $\mathbb{N}$
$\mathrm{X}_0[+\ \mathrm{X}];\mathrm{X}_2[+\ \mathrm{X}]$	ε	$+ 7$	continue $[+\ \mathrm{X}]$
$\mathrm{X}_0[+\ \mathrm{X}];\mathrm{X}_2[+\ \mathrm{X}]$	$+\ \mathrm{X}$	$+ 7$	parse $+$
$\mathrm{X}_0[+\ \mathrm{X}];\mathrm{X}_2[+\ \mathrm{X}]$	X	7	descend X
$\mathrm{X}_0[+\ \mathrm{X}];\mathrm{X}_2[+\ \mathrm{X}];\mathrm{X}_4$	$\mathrm{X} + \mathrm{X} \mid \mathbb{N}$	7	fail*

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$X + X \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+\ X]$
$X_0[+\ X]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
$X_0[+\ X]$	ε	$+ 3 + 7$	continue $[+\ X]$
$X_0[+\ X]$	$+\ X$	$+ 3 + 7$	parse $+$
$X_0[+\ X]$	X	$3 + 7$	descend X
$X_0[+\ X];X_2$	$X + X \mid \mathbb{N}$	$3 + 7$	loop X, $[+\ X]$
$X_0[+\ X];X_2[+\ X]$	$\mathbb{N}$	$3 + 7$	parse $\mathbb{N}$
$X_0[+\ X];X_2[+\ X]$	ε	$+ 7$	continue $[+\ X]$
$X_0[+\ X];X_2[+\ X]$	$+\ X$	$+ 7$	parse $+$
$X_0[+\ X];X_2[+\ X]$	X	7	descend X
$X_0[+\ X];X_2[+\ X];X_4$	$X + X \mid \mathbb{N}$	7	fail*
$X_0[+\ X];X_2[+\ X];X_4$	$\mathbb{N}$	7	parse $\mathbb{N}$

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	5 + 3 + 7	descend X
X_0	X + X | ℕ	5 + 3 + 7	loop X, [+ X]
X_0[+ X]	ℕ	5 + 3 + 7	parse ℕ
X_0[+ X]	ε	+ 3 + 7	continue [+ X]
X_0[+ X]	+ X	+ 3 + 7	parse +
X_0[+ X]	X	3 + 7	descend X
X_0[+ X];X_2	X + X | ℕ	3 + 7	loop X, [+ X]
X_0[+ X];X_2[+ X]	ℕ	3 + 7	parse ℕ
X_0[+ X];X_2[+ X]	ε	+ 7	continue [+ X]
X_0[+ X];X_2[+ X]	+ X	+ 7	parse +
X_0[+ X];X_2[+ X]	X	7	descend X
X_0[+ X];X_2[+ X];X_4	X + X | ℕ	7	fail*
X_0[+ X];X_2[+ X];X_4	ℕ	7	parse ℕ
X_0[+ X];X_2[+ X];X_4	ε	ε	ascend

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$X + X \mid \mathbb{N}$	$5 + 3 + 7$	loop X, $[+ X]$
$X_0[+ X]$	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
$X_0[+ X]$	ε	$+ 3 + 7$	continue $[+ X]$
$X_0[+ X]$	$+ X$	$+ 3 + 7$	parse $+$
$X_0[+ X]$	X	$3 + 7$	descend X
$X_0[+ X];X_2$	$X + X \mid \mathbb{N}$	$3 + 7$	loop X, $[+ X]$
$X_0[+ X];X_2[+ X]$	$\mathbb{N}$	$3 + 7$	parse $\mathbb{N}$
$X_0[+ X];X_2[+ X]$	ε	$+ 7$	continue $[+ X]$
$X_0[+ X];X_2[+ X]$	$+ X$	$+ 7$	parse $+$
$X_0[+ X];X_2[+ X]$	X	7	descend X
$X_0[+ X];X_2[+ X];X_4$	$X + X \mid \mathbb{N}$	7	fail*
$X_0[+ X];X_2[+ X];X_4$	$\mathbb{N}$	7	parse $\mathbb{N}$
$X_0[+ X];X_2[+ X];X_4$	ε	ε	ascend
$X_0[+ X];X_2[+ X]$	ε	ε	ascend

Parsing $5 + 3 + 7$ with $X ::= X + X \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	5 + 3 + 7	descend X
X_0	X + X | ℕ	5 + 3 + 7	loop X, [+ X]
X_0[+ X]	ℕ	5 + 3 + 7	parse ℕ
X_0[+ X]	ε	+ 3 + 7	continue [+ X]
X_0[+ X]	+ X	+ 3 + 7	parse +
X_0[+ X]	X	3 + 7	descend X
X_0[+ X];X_2	X + X | ℕ	3 + 7	loop X, [+ X]
X_0[+ X];X_2[+ X]	ℕ	3 + 7	parse ℕ
X_0[+ X];X_2[+ X]	ε	+ 7	continue [+ X]
X_0[+ X];X_2[+ X]	+ X	+ 7	parse +
X_0[+ X];X_2[+ X]	X	7	descend X
X_0[+ X];X_2[+ X];X_4	X + X | ℕ	7	fail*
X_0[+ X];X_2[+ X];X_4	ℕ	7	parse ℕ
X_0[+ X];X_2[+ X];X_4	ε	ε	ascend
X_0[+ X];X_2[+ X]	ε	ε	ascend
X_0[+ X]	ε	ε	ascend

Parsing $5 + 3 + 7$ with $\text{X} ::= \text{X} + \text{X} \mid \mathbb{N}$

Stack	Grammar	Input	Action
ε	X	$5 + 3 + 7$	descend X
X_0	$\text{X} + \text{X} \mid \mathbb{N}$	$5 + 3 + 7$	loop X, [+ X]
X_0[+ X]	$\mathbb{N}$	$5 + 3 + 7$	parse $\mathbb{N}$
X_0[+ X]	ε	$+ 3 + 7$	continue [+ X]
X_0[+ X]	$+ \text{X}$	$+ 3 + 7$	parse +
X_0[+ X]	X	$3 + 7$	descend X
X_0[+ X];X_2	$\text{X} + \text{X} \mid \mathbb{N}$	$3 + 7$	loop X, [+ X]
X_0[+ X];X_2[+ X]	$\mathbb{N}$	$3 + 7$	parse $\mathbb{N}$
X_0[+ X];X_2[+ X]	ε	$+ 7$	continue [+ X]
X_0[+ X];X_2[+ X]	$+ \text{X}$	$+ 7$	parse +
X_0[+ X];X_2[+ X]	X	7	descend X
X_0[+ X];X_2[+ X];X_4	$\text{X} + \text{X} \mid \mathbb{N}$	7	fail*
X_0[+ X];X_2[+ X];X_4	$\mathbb{N}$	7	parse $\mathbb{N}$
X_0[+ X];X_2[+ X];X_4	ε	ε	ascend
X_0[+ X];X_2[+ X]	ε	ε	ascend
X_0[+ X]	ε	ε	ascend
ε	ε	ε	done

Conflict-free nonterminal names

Conflict-free nonterminal names

- Template Haskell (4) quotes

Conflict-free nonterminal names

- Template Haskell (4) quotes

```
number :: Parser Int
number = 'number
    ::= (\x y → 10 * x + y) <$> number <*> digit
    <|> digit
```

Conflict-free nonterminal names

- Template Haskell (4) quotes

```
number :: Parser Int
number = 'number
    ::= (\x y → 10 * x + y) <$> number <*> digit
    <|> digit
```

- Alternative: GADTs (5)

Conflict-free nonterminal names

- Template Haskell (4) quotes

```
number :: Parser Int
number = 'number
    ::= (\x y → 10 * x + y) <$> number <*> digit
    <|> digit
```

- Alternative: GADTs (5)

```
data Number a where
  Number :: Number Int
```

Conflict-free nonterminal names

- Template Haskell (4) quotes

```
number :: Parser Int
number = 'number
    ::= (\x y → 10 * x + y) <$> number <*> digit
    <|> digit
```

- Alternative: GADTs (5)

```
data Number a where
  Number :: Number Int
```

Combined with Data Types à la Carte (6)

Conclusion

- We can combine
  - lightweight
  - embedded
  - generalized
  - monadic
- Parser combinators

Future work

Future work

- Disambiguation (Layout, Precedence, Fixity)

Future work

- Disambiguation (Layout, Precedence, Fixity)
- Memoization

Future work

- Disambiguation (Layout, Precedence, Fixity)
- Memoization
- Higher-order combinators

Future work

- Disambiguation (Layout, Precedence, Fixity)
- Memoization
- Higher-order combinators
- Actually start writing parsers

Thank you!

https://github.com/noughtmare/gigaparsec

References

(1) Trevor Jim, Yitzhak Mandelbaum, & David Walker. (2010, January). *Semantics and algorithms for data-dependent grammars.* In Proceedings of the 37th annual ACM SIGPLAN-SIGACT symposium on *Principles of programming languages* (pp. 417-430).

(2) Alfred V. Aho and Jeffrey D. Ullman. (1972). *The Theory of Parsing, Translation and Compiling*, volume 1 — Parsing of *Series in Automatic Computation*. Prentice-Hall.

(3) Thomas van Binsbergen. (2019). *Executable formal specification of programming languages with reusable components* (Doctoral dissertation, Royal Holloway, University of London).

(4) Tim Sheard and Simon Peyton Jones. (2002). *Template meta-programming for Haskell.* In Proceedings of the 2002 ACM SIGPLAN workshop on *Haskell (Haskell '02)*.

(5) Simon Peyton Jones, Geoffrey Washburn, Stephanie Weirich. (2004). *Wobbly types: type inference for generalised algebraic data types.* Technical Report MS-CIS-05-26, Univ. of Pennsylvania.

(6) Wouter Swierstra. (2008). *Data Types à la Carte.* In *Journal of functional programming, 18(4),* 423-436.