Lecture 5: Regular Expressions

Talen en Compilers 2024-2025, period 2

Lawrence Chonavel

Department of Information and Computing Sciences, Utrecht University

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Recap: parser combinators

```
type Parser s a = [t] → [(a,[t])]
```

Recap: parser combinators

```
type Parser s a = [t] → [(a,[t])]
```

```
parseDate :: Parser Char Date
```

Recap: parser combinators

```
type Parser s a = [t] → [(a,[t])]
```

```
parseDate :: Parser Char Date
```

```
<$> ::            (a → b) → Parser s a → Parser s b
<*> :: Parser s (a → b) → Parser s a → Parser s b
```

Recap: parser combinators

```
type Parser s a = [t] → [(a,[t])]
```

```
parseDate :: Parser Char Date
```

```
<$> ::           (a → b) → Parser s a → Parser s b
<*> :: Parser s (a → b) → Parser s a → Parser s b
```

```
<$  ::           a  → Parser s b → Parser s a
<*  :: Parser s a  → Parser s b → Parser s a
<|> :: Parser s a  → Parser s a → Parser s a
```

Recap: parser combinators

```
type Parser s a = [t] → [(a,[t])]
```

```
parseDate :: Parser Char Date
```

```
<$> ::            (a → b) → Parser s a → Parser s b
<*> :: Parser s (a → b) → Parser s a → Parser s b
```

```
<$  ::            a  → Parser s b → Parser s a
<*  :: Parser s a  → Parser s b → Parser s a
<|> :: Parser s a  → Parser s a → Parser s a
```

```
symbol  :: (Eq s) =>  s  → Parser s  s
token   :: (Eq s) => [s] → Parser s [s]
```

Recap: parser combinators

```
type Parser s a = [t] → [(a,[t])]
```

```
parseDate :: Parser Char Date
```

```
<$> ::            (a → b) → Parser s a → Parser s b
<*> :: Parser s (a → b) → Parser s a → Parser s b
```

```
<$  ::            a  → Parser s b → Parser s a
<*  :: Parser s a  → Parser s b → Parser s a
<|> :: Parser s a  → Parser s a → Parser s a
```

```
symbol  :: (Eq s) =>  s  → Parser s  s
token   :: (Eq s) => [s] → Parser s [s]
```

```
satisfy :: (s → Bool) → Parser s s
```

Recap: parser combinators

```
type Parser s a = [t] → [(a,[t])]
```

```
parseDate :: Parser Char Date
```

```
<$> ::            (a → b) → Parser s a → Parser s b
<*> :: Parser s (a → b) → Parser s a → Parser s b
```

```
<$  ::            a  → Parser s b → Parser s a
<*  :: Parser s a  → Parser s b → Parser s a
<|> :: Parser s a  → Parser s a → Parser s a
```

```
symbol  :: (Eq s) =>  s  → Parser s  s
token   :: (Eq s) => [s] → Parser s [s]
```

```
satisfy :: (s → Bool) → Parser s s
```

```
guard   :: (a → Bool) → Parser s a → Parser s a
```

Recap: parser combinators

```
>>= :: Parser s a → (a → Parser s b) → Parser s b
fail   ::      Parser s a
```

Recap: parser combinators

```
>>= :: Parser s a → (a → Parser s b) → Parser s b
fail   ::      Parser s a
```

```
epsilon ::     Parser s ()
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Recap: parser combinators

```
>>= :: Parser s a → (a → Parser s b) → Parser s b
fail    ::      Parser s a
```

```
epsilon ::      Parser s ()
```

```
return  :: a → Parser s a
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Recap: parser combinators

```
>>= :: Parser s a → (a → Parser s b) → Parser s b
fail    ::      Parser s a

epsilon ::      Parser s ()

return  :: a → Parser s a

option  :: a → Parser s a → Parser s a

many    :: Parser s a → Parser s [a]
many1   :: Parser s a → Parser s [a]
```

Recap: parser combinators

```
>>= :: Parser s a → (a → Parser s b) → Parser s b
fail    ::      Parser s a

epsilon ::      Parser s ()

return  :: a → Parser s a

option  :: a → Parser s a → Parser s a

many    :: Parser s a → Parser s [a]
many1   :: Parser s a → Parser s [a]

greedy  :: Parser s a → Parser s [a]
greedy1 :: Parser s a → Parser s [a]
```

Recap: parser combinators

Recap: parser combinators

```
chainl :: Parser s a
       → Parser s (a → a → a)
       → Parser s a
chainr :: Parser s a
       → Parser s (a → a → a)
       → Parser s a
```

Recap: parser combinators

```
chainl :: Parser s a
       → Parser s (a → a → a)
       → Parser s a
chainr :: Parser s a
       → Parser s (a → a → a)
       → Parser s a
```

```
type Op a = (Char, a → a → a)
gen :: [Op a] → Parser Char a → Parser Char a
```

Recap: parser combinators

```
chainl :: Parser s a
       → Parser s (a → a → a)
       → Parser s a
chainr :: Parser s a
       → Parser s (a → a → a)
       → Parser s a
```

```
type Op a = (Char, a → a → a)
gen :: [Op a] → Parser Char a → Parser Char a
```

Objections to Parser Combinators

Objections to Parser Combinators

 "I hate types"

Objections to Parser Combinators


"I hate types"

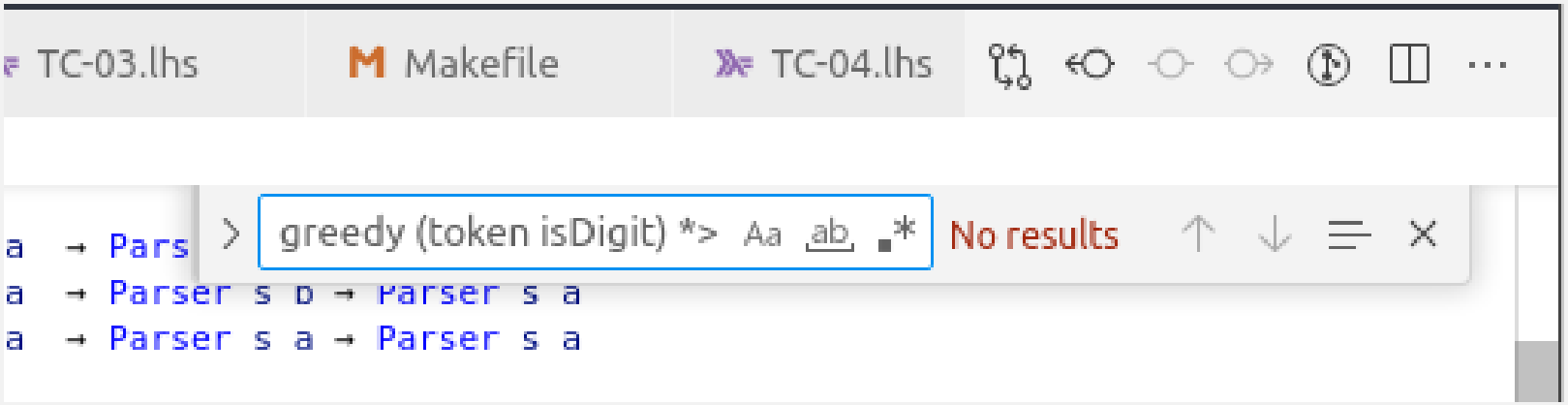

"My other code is C"

Objections to Parser Combinators


"I hate types"


"My other code is C"


"Didn't work in search bar"



A simple subset

```
<|> :: Parser s a          → Parser s a → Parser s a
<*> :: Parser s (a → b) → Parser s a → Parser s b
many  :: Parser s a → Parser s [a]
many1 :: Parser s a → Parser s [a]
option  :: a → Parser s a → Parser s a
symbol  :: (Eq s) => s → Parser s s
satisfy :: (s → Bool) → Parser s s
```

A simpler subset

```
<|> :: Parser Char a          → Parser Char a → Parser Ch
<*> :: Parser Char (a → b) → Parser Char a → Parser Ch
many  :: Parser Char a → Parser Char [a]
many1 :: Parser Char a → Parser Char [a]
option  :: a → Parser Char a → Parser Char a
symbol  :: (Eq Char) => Char → Parser Char Char
satisfy :: (Char → Bool) → Parser Char Char
```

A simpler subset

```
<|> :: Parser Char a          → Parser Char a → Parser Ch
<*> :: Parser Char (a → b) → Parser Char a → Parser Ch
many  :: Parser Char a → Parser Char [a]
many1 :: Parser Char a → Parser Char [a]
option  :: a → Parser Char a → Parser Char a
symbol  :: Char → Parser Char Char
satisfy :: (Char → Bool) → Parser Char Char
```

A simpler subset

```
<|> :: Parser Char a          → Parser Char a → Parser Ch
<*> :: Parser Char (a → b) → Parser Char a → Parser Ch
many  :: Parser Char a → Parser Char [a]
many1 :: Parser Char a → Parser Char [a]
option  :: a → Parser Char a → Parser Char a
symbol  :: Char → Parser Char Char
satisfy :: (Char → Bool) → Parser Char Char
```

```
type P a = Parser Char a
```

A simpler subset

```
<|>  :: P a          → P a → P a
<*>  :: P (a → b) → P a → P b
many  :: P a → P [a]
many1 :: P a → P [a]
option  :: a → P a → P a
symbol  :: Char → P Char
satisfy :: (Char → Bool) → P Char
```

```
type P a = Parser Char a
```

A simpler subset

```
<|> :: P a       → P a → P a
<,> :: P a       → P b → P (a,b)
many  :: P a → P [a]
many1 :: P a → P [a]
option  :: a → P a → P a
symbol  :: Char → P Char
satisfy :: (Char → Bool) → P Char
```

```
type P a = Parser Char a
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

A simpler subset

```
<|> :: P String  → P String → P String
<,> :: P String  → P String → P (String,String)
many  :: P String → P [String]
many1 :: P String → P [String]
option  :: String → P String → P String
symbol  :: Char → P String
satisfy :: (Char → Bool) → P String
```

```
type P a = Parser Char a
```

A simpler subset

```
<|> :: P String  → P String → P String
<+> :: P String  → P String → P String
many  :: P String → P [String]
many1 :: P String → P [String]
option  :: String → P String → P String
symbol  :: Char → P String
satisfy :: (Char → Bool) → P String
```

```
type P a = Parser Char a
```

A simpler subset

```
<|> :: P String  → P String → P String
<+> :: P String  → P String → P String
many  :: P String → P String
many1 :: P String → P String
option  :: String → P String → P String
symbol  :: Char → P String
satisfy :: (Char → Bool) → P String
```

```
type P a = Parser Char a
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

A simpler subset

```
<|> :: P String  → P String → P String
<+> :: P String  → P String → P String
many  :: P String → P String
many1 :: P String → P String
option  ::          P String → P String
symbol  :: Char → P String
satisfy :: (Char → Bool) → P String
```

```
type P a = Parser Char a
```

A simpler subset

```
<|> :: P String  → P String → P String
<+> :: P String  → P String → P String
many  :: P String → P String
many1 :: P String → P String
option  ::          P String → P String
symbol  :: Char → P String
satisfy :: (Char → Bool) → P String
```

```
type R = Parser Char String
```

A simpler subset

```
<|> :: R → R → R
<+> :: R → R → R
many  :: R → R
many1 :: R → R
option  :: R → R
symbol  :: Char → R
satisfy :: (Char → Bool) → R
```

```
type R = Parser Char String
```

A simpler subset

```
<|> :: R → R → R
<+> :: R → R → R
many  :: R → R
many1 :: R → R
option  :: R → R
symbol  :: Char → R
satisfy :: (Char → Bool) → R
```

```
type R = Parser Char String
```

A simpler subset

```
<|> :: R → R → R
<+> :: R → R → R
many  :: R → R
many1 :: R → R
option  :: R → R
symbol  :: Char → R
satisfy :: (Char → Bool) → R
```

```
type R = Parser Char String
```

Challenge!

Universiteit Utrecht

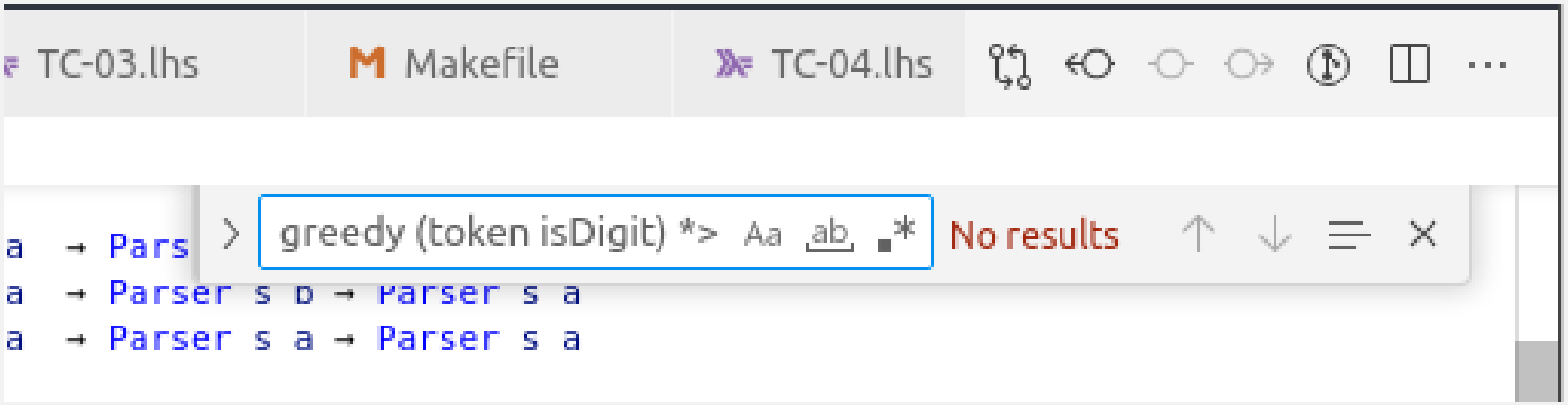
[Faculty of Science
Information and Computing
Sciences]

Progress on Objections


"I hate types"

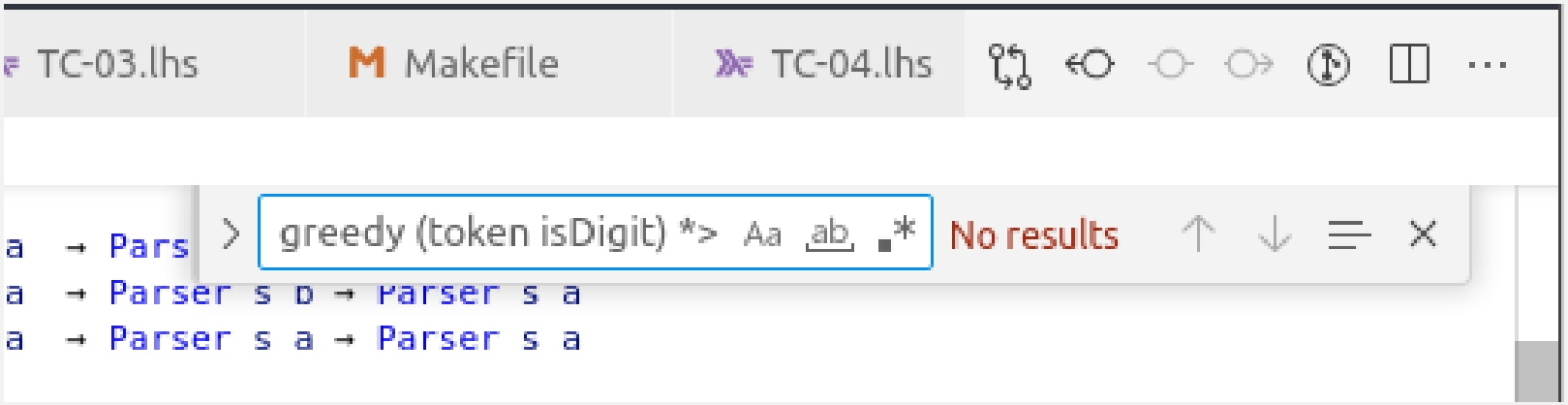

"My other code is C"


"Didn't work in search bar"



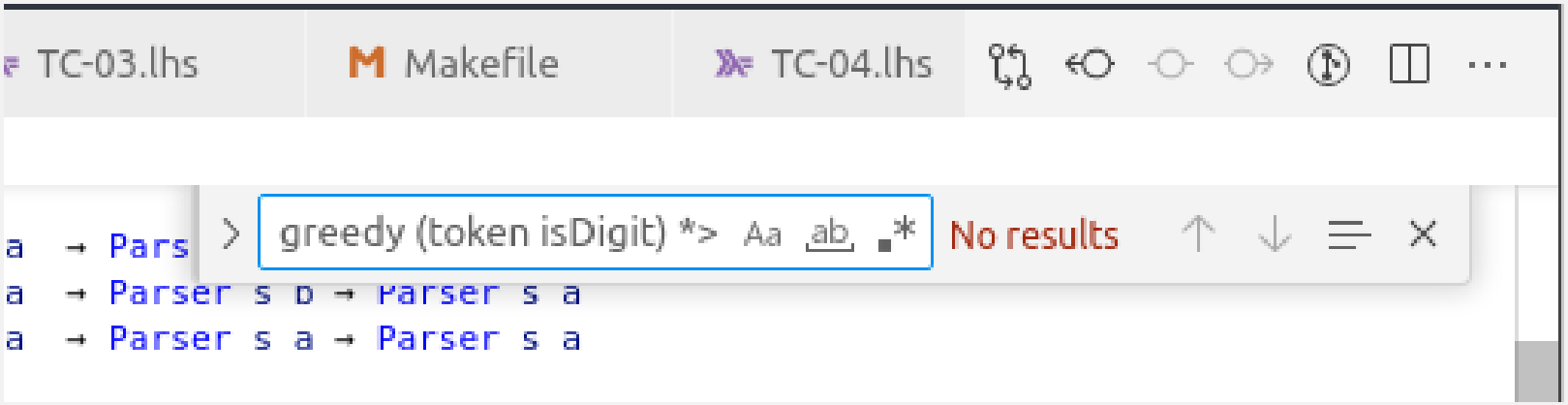
Progress on Objections

- “I hate types” ✅
- “My other code is C” ✅
- “Didn’t work in search bar”



Progress on Objections

- “I hate types” ✅
- “My other code is C” ✅
- “Didn’t work in search bar” 👈



A language for the search bar

```
-- Haskell types

<|> :: R → R → R
<+> :: R → R → R
many  :: R → R
many1 :: R → R
option  :: R → R
symbol  :: Char → R
satisfy :: (Char → Bool) → R
```

A language for the search bar

```
-- Haskell
```

```
p1 <|> p2
p1 <+> p2
many p
many1 p
option p
symbol c
satisfy q
```

A language for the search bar

```
-- Haskell
p1 <|> p2
p1 <+> p2
many p
many1 p
option p
symbol c
satisfy q
```

```
-- New Language
r1|r2
r1r2
r*
r+
r?
c
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

A language for the search bar

```
-- Haskell
p1 <|> p2
p1 <+> p2
many p
many1 p
option p
symbol c
satisfy q
```

```
-- Regular Expression
r1|r2
r1r2
r*
r+
r?
c
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

A language for the search bar

```haskell
-- Haskell
p1 <|> p2
p1 <+> p2
many p
many1 p
option p
symbol c
satisfy isDigit
```

```regex
-- Regular Expression
r1|r2
r1r2
r*
r+
r?
c
\d
```

A language for the search bar

```haskell
-- Haskell
p1 <|> p2
p1 <+> p2
many p
many1 p
option p
symbol c
satisfy isDigit
satisfy isWhitespace
```

```regex
-- Regular Expression
r1|r2
r1r2
r*
r+
r?
c
\d
\s
```

A language for the search bar

```haskell
-- Haskell
p1 <|> p2
p1 <+> p2
many p
many1 p
option p
symbol c
satisfy isDigit
satisfy isWhitespace
satisfy (not . isWhitespace)
```

```regex
-- Regular Expression
r1|r2
r1r2
r*
r+
r?
c
\d
\s
\S
```

A language for the search bar

```haskell
-- Haskell
p1 <|> p2
p1 <+> p2
many p
many1 p
option p
symbol c
satisfy isDigit
satisfy isWhitespace
satisfy (not . isWhitespace)
satisfy (`elem` ['a'..'z'])
```

```
-- Regular Expression
r1|r2
r1r2
r*
r+
r?
c
\d
\s
\S
[a-z]
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

A language for the search bar

```haskell
-- Haskell
p1 <|> p2
p1 <+> p2
many p
many1 p
option p
symbol c
satisfy isDigit
satisfy isWhitespace
satisfy (not . isWhitespace)
satisfy (`elem` ['a'..'z'])
satisfy (const True)
```

```
-- Regular Expression
r1|r2
r1r2
r*
r+
r?
c
\d
\s
\S
[a-z]
.
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

A language for the search bar

```haskell
-- Haskell
p1 <|> p2
p1 <+> p2
many p
many1 p
option p
symbol c
satisfy isDigit
satisfy isWhitespace
satisfy (not . isWhitespace)
satisfy (`elem` ['a'..'z'])
satisfy (const True)

...
```

```regex
-- Regular Expression
r1|r2
r1r2
r*
r+
r?
c
\d
\s
\S
[a-z]
.

...
```

Example

```
matchNums "{30,60,155,1024}" ✅
```

Example

```
matchNums "{30,60,155,1024}" ✅
```

```
matchNums "{}" ✅
```

Example

```
matchNums "{30,60,155,1024}" ✅
```

```
matchNums "{}" ✅
```

```
matchNums "{3.1}" ✅
```

Example

```
matchNums "{30,60,155,1024}" ✅
```

```
matchNums "{}" ✅
```

```
matchNums "{3.1}" ✅
```

```
matchNums "{3.1,-2}" ✅
```

Example

```
matchNums "{30,60,155,1024}" ✅
```

```
matchNums "{}" ✅
```

```
matchNums "{3.1}" ✅
```

```
matchNums "{3.1,-2}" ✅
```

```
matchNums "{π,-2}" ❌
```

Example

```
matchNums "{30,60,155,1024}" ✅
```

```
matchNums "{}" ✅
```

```
matchNums "{3.1}" ✅
```

```
matchNums "{3.1,-2}" ✅
```

```
matchNums "{π,-2}" ❌
```

```
matchNums "{3.1,-2,}" ❌
```

Example

```
matchNums "{30,60,155,1024}" ✅
```

```
matchNums "{}" ✅
```

```
matchNums "{3.1}" ✅
```

```
matchNums "{3.1,-2}" ✅
```

```
matchNums "{π,-2}" ❌
```

```
matchNums "{3.1,-2,}" ❌
```

```
matchNums " {3.1,  -2 }" ❌
```

Implementing Example

```
matchNums :: P
matchNums = ???
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Implementing Example

```
matchNums :: P
```

```
matchNums
  =    symbol '{' <+> symbol '}'
  <|> symbol '{' <+> matchNum <+> symbol '}'
  <|> symbol '{' many1 (matchNum <+> symbol ',')
  <+> matchNum <+> symbol '}'
```

Implementing Example

```
matchNums,matchNum :: P
```

```
matchNums
  =    symbol '{' <+> symbol '}'
  <|> symbol '{' <+> matchNum <+> symbol '}'
  <|> symbol '{' many1 (matchNum <+> symbol ',')
  <+> matchNum <+> symbol '}'
```

```
matchNum
  =    option (symbol '-') <+> matchNat
  <+> option (symbol '.' <+> matchNat)
```

Implementing Example

```
matchNums,matchNum,matchNat :: P
```

```
matchNums
  =    symbol '{' <+> symbol '}'
  <|> symbol '{' <+> matchNum <+> symbol '}'
  <|> symbol '{' many1 (matchNum <+> symbol ',')
  <+> matchNum <+> symbol '}'
```

```
matchNum
  =    option (symbol '-') <+> matchNat
  <+> option (symbol '.' <+> matchNat)
```

```
matchNat = many1 (satisfy isDigit)
```

Implementing Example

```
matchNums,matchNum,matchNat :: P
```

```
matchNums
  =    symbol '{' <+> symbol '}'
  <|> symbol '{' <+> matchNum <+> symbol '}'
  <|> symbol '{' many1 (matchNum <+> symbol ',')
  <+> matchNum <+> symbol '}'
```

```
matchNum
  =    option (symbol '-') <+> matchNat
  <+> option (symbol '.' <+> matchNat)
```

```
matchNat = many1 (satisfy isDigit)
```

Implementing Example

```
matchNums,matchNum,matchNat :: P
```

```
matchNums
  =   symbol '{' <+> symbol '}'
  <|> symbol '{' <+> matchNum <+> symbol '}'
  <|> symbol '{' many1 (matchNum <+> symbol ',')
  <+> matchNum <+> symbol '}'
```

```
matchNum
  =   option (symbol '-') <+> matchNat
  <+> option (symbol '.' <+> matchNat)
```

```
-?\d+(\.\d+)?
```

```
matchNat = many1 (satisfy isDigit)
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Implementing Example

```
matchNums,matchNum,matchNat :: P
```

```
matchNums
  =   symbol '{' <+> symbol '}'
  <|> symbol '{' <+> matchNum <+> symbol '}'
  <|> symbol '{' many1 (matchNum <+> symbol ',')
  <+> matchNum <+> symbol '}'
```

```
{}
```

```
matchNum
  =   option (symbol '-') <+> matchNat
  <+> option (symbol '.' <+> matchNat)
```

```
-?\d+(\.\d+)?
```

```
matchNat = many1 (satisfy isDigit)
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Implementing Example

```
matchNums,matchNum,matchNat :: P
```

```
matchNums
  =   symbol '{' <+> symbol '}'
  <|> symbol '{' <+> matchNum <+> symbol '}'
  <|> symbol '{' many1 (matchNum <+> symbol ',')
  <+> matchNum <+> symbol '}'
```

```
{} |{-?\d+(\.\d+)?}
```

```
matchNum
  =   option (symbol '-') <+> matchNat
  <+> option (symbol '.' <+> matchNat)
```

```
-?\d+(\.\d+)?
```

```
matchNat = many1 (satisfy isDigit)
```

\d+

Implementing Example

```
matchNums,matchNum,matchNat :: P
```

```
matchNums
  =   symbol '{' <+> symbol '}'
  <|> symbol '{' <+> matchNum <+> symbol '}'
  <|> symbol '{' many1 (matchNum <+> symbol ',')
  <+> matchNum <+> symbol '}'
```

```
{} |{-?\d+(\.\d+)?}
|{(-?\d+(\.\d+)?,)+-?\d+(\.\d+)?}
```

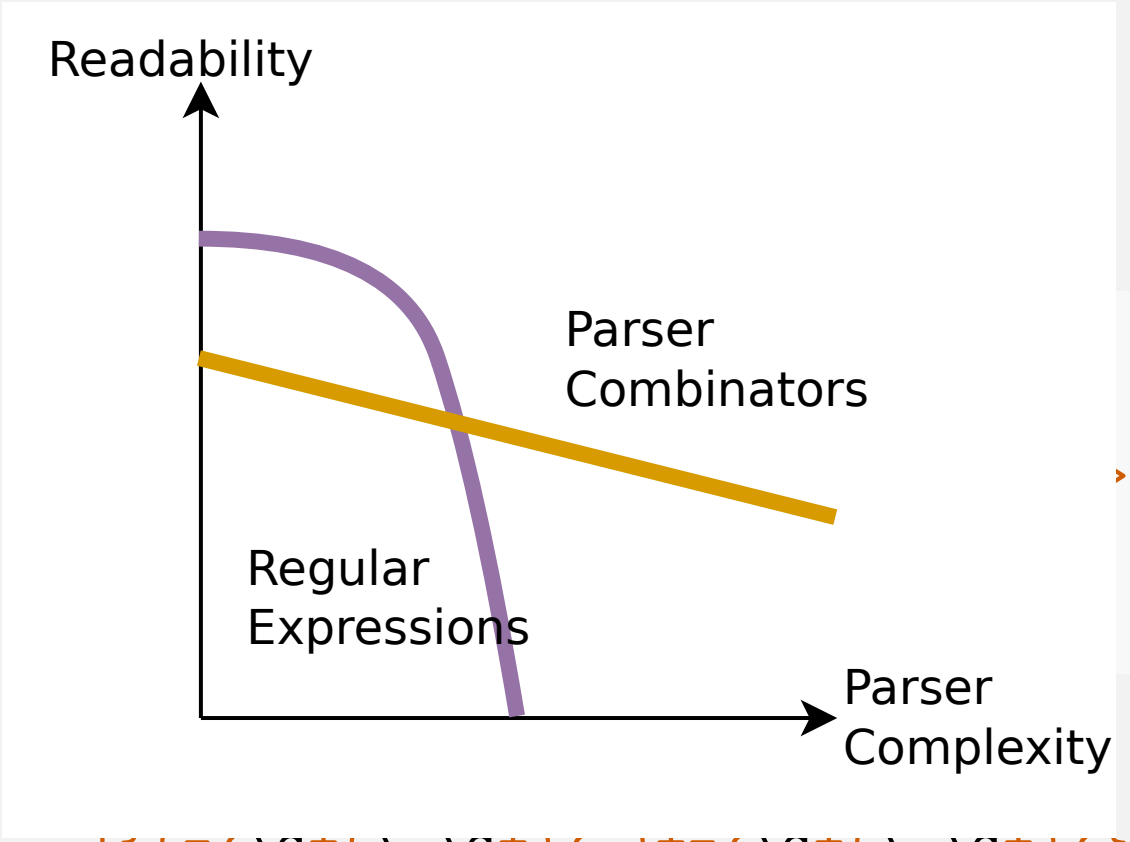
```
matchNum
  =   option (symbol '-') <+> matchNat
  <+> option (symbol '.' <+> matchNat)
```

```
-?\d+(\.\d+)?
```

```
matchNat = many1 (satisfy isDigit)
```

```
\d+
```

Impl




```
> symbol '}'
<+> symbol ',')
```

```
matchNum
  =    option (symbol '-') <+> matchNat
  <+> option (symbol '.' <+> matchNat)
```

```
-?\d+(\.\d+)?
```

```
matchNat = many1 (satisfy isDigit)
```

```
\d+
```

Break time

- Challenges:
  - regextutorials.com/excercise.html
    - Exercises `[1-3]|5|7|1[0236]`
  - What does `grep -RE "Parser\s+\w+\s+\w+"` do?
  - Use your editor's regex search
  - Debugger: regexr.com

Recap

```
<|>  :: R → R → R
<+>  :: R → R → R
many  :: R → R
many1 :: R → R
option  :: R → R
symbol  :: Char → R
satisfy :: (Char → Bool) → R
```

Recap

```
<|> :: R → R → R          r1|r2
<+> :: R → R → R          r1r2
many  :: R → R            r*
many1 :: R → R            r+
option  :: R → R          r?
symbol  :: Char → R       c
satisfy :: (Char → Bool) → R  \d \s \S [a-z] ...
```

```
{}|{-?\d+(\.\d+)?}
|{(-?\d+(\.\d+)?,)+-?\d+(\.\d+)?}
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Recap

```
<|> :: R → R → R
<+> :: R → R → R
many  :: R → R
many1 :: R → R
option  :: R → R
symbol  :: Char → R
satisfy :: (Char → Bool) → R
```

$r_1|r_2$
r_1r_2
$r*$
$r+$
$r?$
c
\d \s \S [a-z] ...

```
{}|{-?\d+(\.\d+)?}
|{(-?\d+(\.\d+)?,)+-?\d+(\.\d+)?}
```

Ubiquity

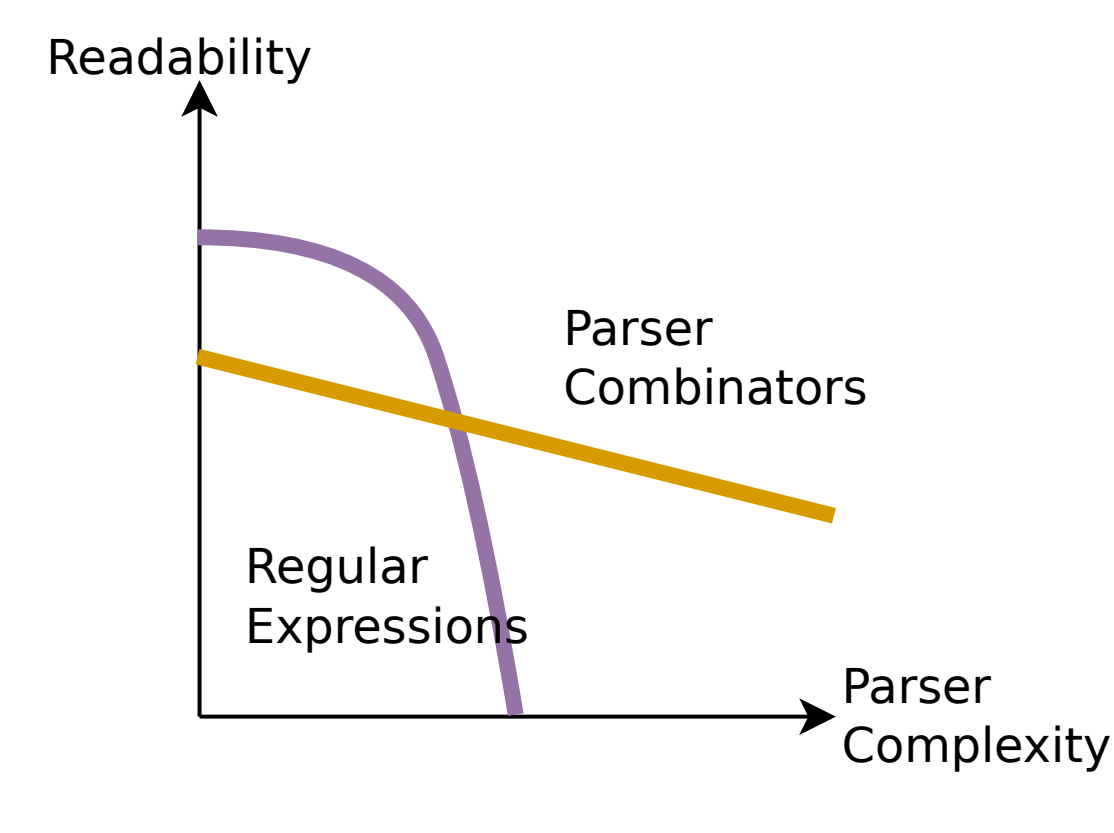
code eclipse sublime idea xcode notepad++ emacs vim vi ed grep awk sed find perl python javascript lua …

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

RegExp problems

- Write-only code



RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

- No recursion

RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

- No recursion

RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

- No recursion
  - ((((())(()))(()))(()))

RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

- No recursion
  - ((((()())(()))())())
  - ((((()()(()))())())

RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

- No recursion
  - `((((()())(()))())())`
  - `((((()()(()))())())`
  - `abacabadabacaba`

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

- No recursion (regular languages only)
  - `((((()())(()))())())`
  - `((((()()(()))())())`
  - `abacabadabacaba`
  - `abacabadabacada`

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

- No recursion (regular languages only)
  - `((((()())(()))())())`
  - `((((()()(()))())())`
  - `abacabadabacaba`
  - `abacabadabacada`
- Q: how to prove a language regular?

RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

- No recursion (regular languages only)
  - `((((()())(()))())())`
  - `((((()()(()))())())`
  - `abacabadabacaba`
  - `abacabadabacada`
- Q: how to prove a language regular?
- Q: how to prove a language not regular?

RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

- No recursion (regular languages only)
  - `((((()())(()))())())`
  - `((((()()(()))())())`
  - `abacabadabacaba`
  - `abacabadabacada`
- Q: how to prove a language regular?
- Q: how to prove a language not regular?

  A: L regular $\Rightarrow \exists\ l \in \mathbb{N}.\ \forall\ w \in L.\ |w| \geq l \Rightarrow \exists\ x,y,z.\ w=xyz \wedge \forall\ n \in \mathbb{N}.\ xy^n z \in L$

RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

- No recursion (regular languages only)
  - `((((()())(()))())())`
  - `((((()()(()))())())`
  - `abacabadabacaba`
  - `abacabadabacada`
- Q: how to prove a language regular?
- Q: how to prove a language not regular?
  A: Come back for pumping lemma lecture!

RegExp problems

- Write-only code
- No parsing

```
matchRegExp "\w+ on (toast|bread)"
            "beans on toasted potato"
        == [("beans on toast","ed potato")]
```

- No recursion (regular languages only)
  - `((((()())(()))())())`
  - `((((()()(()))())())`
  - `abacabadabacaba`
  - `abacabadabacada`
- Q: how to prove a language regular?
- Q: how to prove a language not regular?
- Performance

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Performance

```
matchRegExp "a*a*" "aaaaaaaaaaaaaaa"
```

Performance

```
matchRegExp "a*a*" "aaaaaaaaaaaaaaa"
  == [ ("","aaaaaaaaaaaaaaa")
     , ("a","aaaaaaaaaaaaaa")
     , ("aa","aaaaaaaaaaaaa")
     , ("aaa","aaaaaaaaaaaa")
     , ("aaaa","aaaaaaaaaa")
     , ("aaaaa","aaaaaaaaa")
     , ("aaaaaa","aaaaaaaa")
     , ("aaaaaaa","aaaaaaa")
     , ("aaaaaaaa","aaaaaa")
     , ("aaaaaaaaa","aaaaa")
     , ("aaaaaaaaaa","aaaa")
     , ("aaaaaaaaaaa","aaa")
     , ("aaaaaaaaaaaa","aa")
     , ("aaaaaaaaaaaaa","a")
     , ("aaaaaaaaaaaaaa","")
     , ("","aaaaaaaaaaaaaa")
```

Performance

```
matchRegExp "a*a*$" "aaaaaaaaaaaaaaa"
```

Performance

```
matchRegExp "a*a*$" "aaaaaaaaaaaaaaa"
  == [ ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     , ("aaaaaaaaaaaaaaa","")
     ]
```

Performance

```
matchRegExp "a*a*$" "aaaaaaaaaaaaaaa"
  == [ (""++"aaaaaaaaaaaaaaa","")
     , ("a"++"aaaaaaaaaaaaaa","")
     , ("aa"++"aaaaaaaaaaaaa","")
     , ("aaa"++"aaaaaaaaaaaa","")
     , ("aaaa"++"aaaaaaaaaaa","")
     , ("aaaaa"++"aaaaaaaaaa","")
     , ("aaaaaa"++"aaaaaaaaa","")
     , ("aaaaaaa"++"aaaaaaaa","")
     , ("aaaaaaaa"++"aaaaaaa","")
     , ("aaaaaaaaa"++"aaaaaa","")
     , ("aaaaaaaaaa"++"aaaaa","")
     , ("aaaaaaaaaaa"++"aaaa","")
     , ("aaaaaaaaaaaa"++"aaa","")
     , ("aaaaaaaaaaaaa"++"aa","")
     , ("aaaaaaaaaaaaaa"++"a","")
     , ("aaaaaaaaaaaaaaa"++"","")
     ]
```

Performance

```
matchRegExp "a*a*$" "aaaaaaaaaaaaaaa"
  == [ (""++"aaaaaaaaaaaaaaa","")
     , ("a"++"aaaaaaaaaaaaaa","")
     , ("aa"++"aaaaaaaaaaaaa","")
     , ("aaa"++"aaaaaaaaaaaa","")
     , ("aaaa"++"aaaaaaaaaaa","")
     , ("aaaaa"++"aaaaaaaaaa","")
     , ("aaaaaa"++"aaaaaaaaa","")
     , ("aaaaaaa"++"aaaaaaaa","")
     , ("aaaaaaaa"++"aaaaaaa","")
     , ("aaaaaaaaa"++"aaaaaa","")
     , ("aaaaaaaaaa"++"aaaaa","")
     , ("aaaaaaaaaaa"++"aaaa","")
     , ("aaaaaaaaaaaa"++"aaa","")
     , ("aaaaaaaaaaaaa"++"aa","")
     , ("aaaaaaaaaaaaaa"++"a","")
     , ("aaaaaaaaaaaaaaa"++"","")
     ]
```

Performance

```
head $ matchRegExp "a*a*$" "aaaaaaaaaaaaaaa"
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Performance

```
head $ matchRegExp "a*a*$" "aaaaaaaaaaaaaaa"
  == [ ("aaaaaaaaaaaaaaa","")]
```

Performance

```
head $ matchRegExp "a*a*$" "aaaaaaaaaaaaaaa"
  == [ (""++"aaaaaaaaaaaaaaa","")]
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Performance

```
head $ matchRegExp "a*a*$" "aaaaaaaaaaaaaaa"
  == [ (""++"aaaaaaaaaaaaaaa","")]
```

Fast?

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Performance

```
head $ matchRegExp "a*a*$" "aaaaaaaaaaaaaaa"
  == [ (""++"aaaaaaaaaaaaaaa","")]
```

Fast?

Performance

```
head $ matchRegExp "a*a*$" "aaaaaaaaaaaaaaa"
  == [ (""++"aaaaaaaaaaaaaaa","")]
```

Fast?

Laziness!

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaaab"
```

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
  == [ ("aaaaaaab","")]
```

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
  == [ ("aaaa"++"aaab","")]
```

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
  == [ ("aaaa"++"aaab","")]
```

Fast?

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
  == [ ("aaaa"++"aaab","")]
```

Fast?

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
  == [ ("aaaa"++"aaab","")]
```

Fast?

aaaa aaab

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
  == [ ("aaaa"++"aaab","")]
```

Fast?

🐢

aaaa aaab

aaa❌

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
  == [ ("aaaa"++"aaab","")]
```

Fast?

aaaa aaab

aaa❌

a aaa❌

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
  == [ ("aaaa"++"aaab","")]
```

Fast?

🐢

aaaa aaab

aaa❌

a aaa❌

aa aaa❌

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
  == [ ("aaaa"++"aaab","")]
```

Fast?

🐢

aaaa aaab

aaa❌

a aaa❌

aa aaa❌

aaa aaa❌

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
  == [ ("aaaa"++"aaab","")]
```

Fast?

🐢

aaaa aaab

aaa❌

a aaa❌

aa aaa❌

aaa aaa❌

aaaa aaab ✅

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Performance

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
  == [ ("aaaa"++"aaab","")]
```

Fast?

🐢

aaaa aaab

aaa❌

a aaa❌

aa aaa❌

aaa aaa❌

aaaa aaab ✅

Laziness not enough!

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Can't `greedy` fix this?

_* must match as much as possible

Can't `greedy` fix this?

_* must match as much as possible

Can't `greedy` fix this?

`_*` must match as much as possible

With `greedy` backing `_*`:

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Can't `greedy` fix this?

`_*` must match as much as possible

With `greedy` backing `_*`:

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
```

```
aaaaaaa ❌
```

Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Can't `greedy` fix this?

`_*` must match as much as possible without making the match fail

With `greedy` backing `_*`:

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
```

```
aaaaaaa ❌
```

Can't `greedy` fix this?

`_*` must match as much as possible without making the match fail

With `greedy` backing `_*`:

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
```

`aaaaaaa` ❌

Challenges

- Craft regex with `O(length^2)` matching time
- Craft regex with `O(2^length)` matching time
- Hint: swtch.com/~rsc/regexp/regexp1.html

Can't greedy fix this?

`_*` must match as much as possible without making the match fail

With `greedy` backing `_*`:

```
head $ matchRegExp "a*aaab$" "aaaaaaab"
```

`aaaaaaa` ❌

Challenges

- Craft regex with `O(length^2)` matching time
- Craft regex with `O(2^length)` matching time
- Hint: swtch.com/~rsc/regexp/regexp1.html

Next lecture

`O(length * regexp size)` matching time