Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

Talen en Compilers

2017 - 2018, period 2

João Pizani Flor

Department of Information and Computing Sciences
Utrecht University

November 20, 2017

3. Parser combinators - Very simple usage

Parser Combinators

For the first lab assignment (P1a - DateTime), you will need to use parser combinators.

Details on what they are and the theory comes on Wednesday.

But first: just how to start **using** them?

The type of a parser

```
data Parser s r
```

Two arguments:

- First (s) is the type of **symbol** (for now, Char)
- Second (r) is the type of the **result** (Date, Bool, etc.)

We'll use some **basic parsers** as well as some **combinators**.
Some pre-existing special-purpose parsers can also be handy.

Basic parsers

From ParseLib.Abstract.Core:

$\text{satisfy} :: (s \rightarrow \text{Bool}) \rightarrow \text{Parser}\ s\ s$

From ParseLib.Abstract.Derived:

$\text{symbol} :: \text{Eq}\ s \Rightarrow s \rightarrow \text{Parser}\ s\ s$
$\text{token} :: \text{Eq}\ s \Rightarrow [s] \rightarrow \text{Parser}\ s\ [s]$

Some combinators

From ParseLib.Abstract.Core, sequence, choice, and processing the result:

$$(<\!\!*\!\!>) :: \text{Parser s } (a \rightarrow b) \rightarrow \text{Parser s a} \rightarrow \text{Parser s b}$$
$$(<\!|\!>) :: \text{Parser s a} \rightarrow \text{Parser s a} \rightarrow \text{Parser s a}$$
$$(<\$>) :: (a \rightarrow b) \rightarrow \text{Parser a} \rightarrow \text{Parser b}$$

Example on how to use them:

$$\text{ints} = (\lambda a\ \_\ b \rightarrow (a, b))\ <\$>\ \text{integer}\ <\!\!*\!\!>\ \text{symbol '},\text{'}\ <\!\!*\!\!>\ \text{integer}$$

Running a Parser

To run a parser, you must use the parse function (from ParseLib.Abstract.Core, give it the parser and some input.

$\text{parse} :: \text{Parser s a} \rightarrow [\text{s}] \rightarrow [(\text{a}, [\text{s}])]$

It returns a list of the successful parses, along with possibly unused tails of the input (empty list means failure).

Example:

```
parse ints "23,11" == [((23, 11), "")]
parse ints "23,11bla" == [((23, 11), "bla")]
parse ints "whatever" == []
```

Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

Last remarks

- This was just a little spoiler, Wednesday will be more thorough
- In the lab: use the help of the assistants! And the documentation!
  - `https://hackage.haskell.org/package/uu-tc`