Parsing F# with PetitParser

F# indentation sensitive
lightweight syntax

Milan Kubicek
SCG Seminar
06.01.2015
some F# code..

```fsharp
#light "off"

let rec fibonacci n =
    if n = 1 || n = 2 then ( 1 )
    else ( let result = fibonacci(n-1) + fibonacci(n-2) in
             | result )

for i in 1 .. 10 do ( printfn "%d: %d" i (fibonacci i) ) done
```
..some nicer F# code

```fsharp
#light "off"

let rec fibonacci n =
    (if (n = 1 || n = 2) then
    (1)
    else
    (let result = fibonacci(n-1) + fibonacci(n-2) in
    result))

for i in 1 .. 10 do
    (printfn "%d %d" i (fibonacci i)) done

Output:

1: 1
2: 1
3: 2
4: 3
5: 5
6: 8
7: 13
8: 21
9: 34
10: 55
```
indentation as secondary notation

```plaintext
#light "off"

let rec fibonacci n =
  if (n = 1 || n = 2) then
    1
  else
    let result = fibonacci(n-1)
    + fibonacci(n-2)
    in
    result

for i in 1 .. 10 do
  printfn "%d: %d" i (fibonacci i)
done
```

indentation as formal notation

```plaintext
#light "on"

let rec fibonacci n =
  if (n = 1 || n = 2) then
    1
  else
    let result = fibonacci(n-1)
    + fibonacci(n-2)
    in
    result

for i in 1 .. 10 do
  printfn "%d: %d" i (fibonacci i)
done
```
indentation in programming

“curly bracket programming languages”

secondary notation / syntactic sugar

define structure

Haskell

Python

“indentation sensitive programming languages”

F#
off-side rule

off-side line

off-side  on-side

valid pass!

foul pass!
off-side rule

```haskell
#light "on"

let rec fibonacci n =
    if (n = 1 || n = 2) then
        1
    else
        let result = fibonacci(n-1) + fibonacci(n-2)
        result

for i in 1 .. 10 do
    printfn "%%d: %%d" i (fibonacci i)
```
off-side rule

off-side line

off-side  on-side

```
1  #light "on"
2
3  let rec fibonacci n =
4     if (n = 1 || n = 2) then
5         1
6     else
7         let result = fibonacci(n-1)
8         let result = fibonacci(n-2)
9     result
10
11  for i in 1 .. 10 do
12     printfn "%%d: %d" i (fibonacci i)
```
off-side rule

```agda
#light "on"

let rec fibonacci n =
  if (n = 1 || n = 2) then
    1
  else
    let result = fibonacci(n-1) + fibonacci(n-2)
    in result

for i in 1 .. 10 do
  printf "\%d: \%d" i (fibonacci i)
```
off-side rule exceptions

off-side line

off-side  on-side

```haskell
#light "on"

let rec fibonacci n =
  if (n = 1 || n = 2) then
    1
  else
    let result = fibonacci(n-1) + fibonacci(n-2)
    in result
  
for i in 1 .. 10 do
  printfn "%d: %d" i (fibonacci i)
```
off-side rule exceptions

“then-&else-body on-side aligned”

“else-body aligned”

“then-body aligned”
challenges

(Petit) Parsing knowledge

F# language definition

mini F# PetitParser grammar (using PetitIndent)
F# lightweight syntax

indentation as primary notation:
..code position changes semantics!

off-side line
aligned
on-side
off-side
WHITESPACE

IT'S NOT JUST FOR FORMATTING ANYMORE

questions?
secondary notation

“...An example of secondary notation is in computer programming, where code is often displayed with positioning, indentation, color and symmetry. This does not affect the behaviour of the program, but it makes it easier to read and understand the code...”

Off-side rule

“Any non-whitespace token to the left of the first such token on the previous line is taken to be the start of a new declaration.”

–Peter J. Landin, “The Next 700 Programming Languages”