

# Scala - XML, combinator parsing, actors (dodatek)

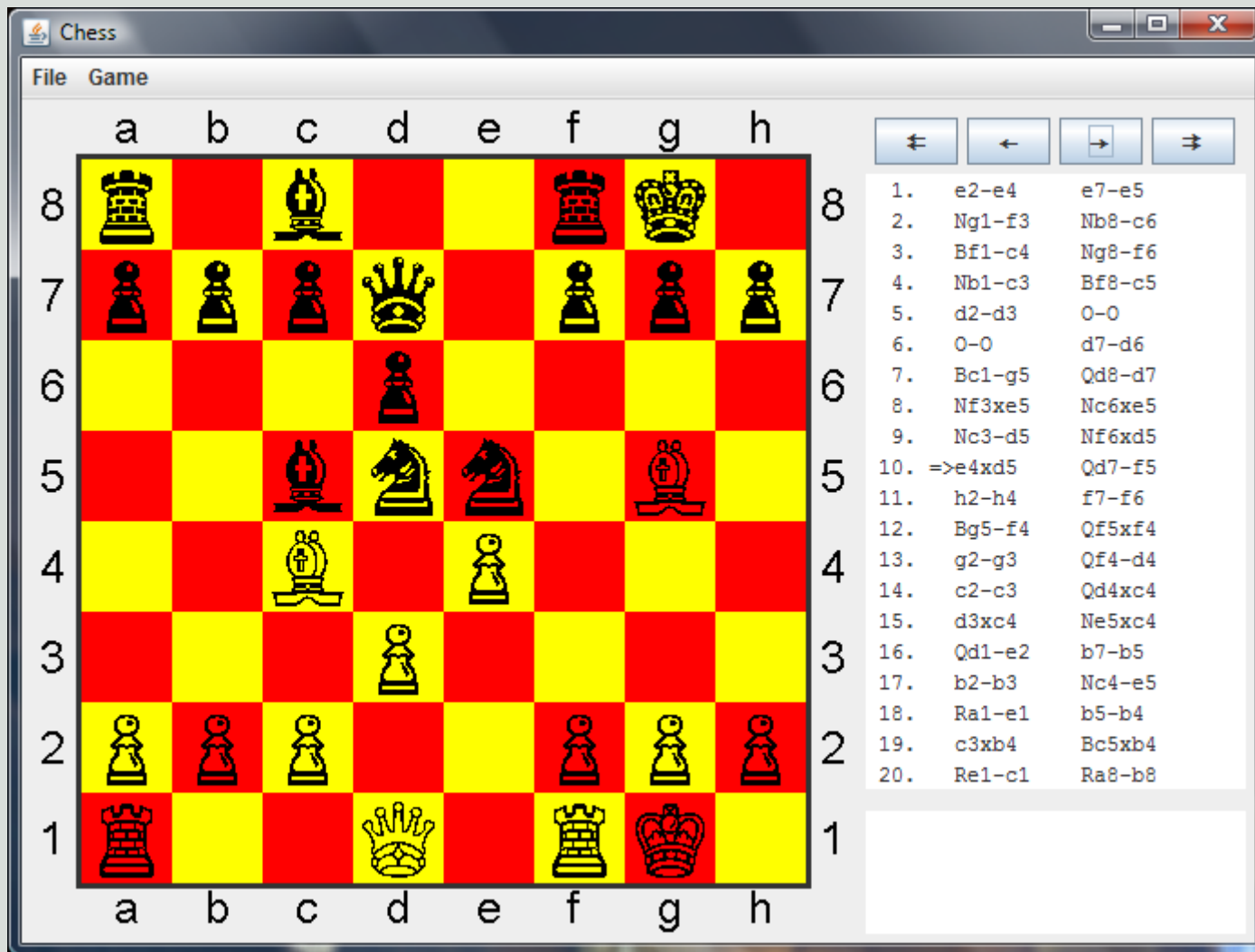
Grzegorz Balcererek

gbalcererek@echostar.pl

This work is licensed under a  
Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License  
(<http://creativecommons.org/licenses/by-nc-sa/3.0/>).

# combinator parsing

## przykład: Chess



The screenshot shows a chess game window titled "Chess". The interface includes a menu bar with "File" and "Game", a chessboard with files a-h and ranks 1-8, and a move list on the right. The board shows a game in progress with various pieces placed. The move list contains 20 moves, with the 10th move highlighted in grey.

Rank	a	b	c	d	e	f	g	h
8	♖		♜			♖	♔	
7	♟	♞	♝	♚		♞	♝	♞
6				♞				
5			♜	♞	♞		♝	
4			♞		♞			
3				♞				
2	♞	♝	♞			♝	♞	♝
1	♖			♚		♖	♔	

Move list:

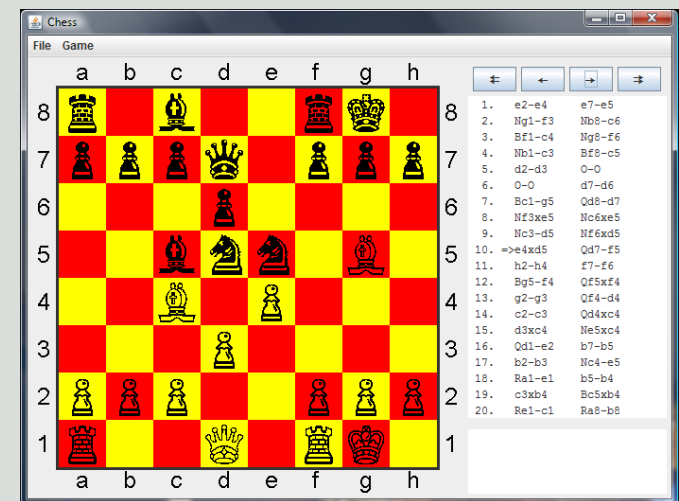
1. e2-e4 e7-e5
2. Ng1-f3 Nb8-c6
3. Bf1-c4 Ng8-f6
4. Nb1-c3 Bf8-c5
5. d2-d3 O-O
6. O-O d7-d6
7. Bc1-g5 Qd8-d7
8. Nf3xe5 Nc6xe5
9. Nc3-d5 Nf6xd5
10. =>e4xd5 Qd7-f5
11. h2-h4 f7-f6
12. Bg5-f4 Qf5xf4
13. g2-g3 Qf4-d4
14. c2-c3 Qd4xc4
15. d3xc4 Ne5xc4
16. Qd1-e2 b7-b5
17. b2-b3 Nc4-e5
18. Ra1-e1 b5-b4
19. c3xb4 Bc5xb4
20. Re1-c1 Ra8-b8

# combinator parsing

## Chess: odczyt notacji szachowej

1.	e2-e4	e7-e5
2.	Ng1-f3	Nb8-c6
3.	Bf1-c4	Ng8-f6
4.	Nb1-c3	Bf8-c5
5.	d2-d3	O-O
6.	O-O	d7-d6
7.	Bc1-g5	Qd8-d7
8.	Nf3xe5	Nc6xe5
9.	Nc3-d5	Nf6xd5
10.	e4xd5	Qd7-f5
11.	h2-h4	f7-f6
12.	Bg5-f4	Qf5xf4

(...)



# combinator parsing ChessNotationParser (1)

```
object ChessNotationParser extends RegexParsers {  
  
  def apply(content: String) = parseAll(gameParser, content)  
  
  def gameParser: Parser[Game] =  
    opt(whitespace) ~> moveNumberAndRest(new Game)  
  
  def moveNumberAndRest(game: Game): Parser[Game] =  
    game.nextMoveNumber.toString ~ "." ~ whitespace ~>  
      whiteMoveAndRest(game)  
  
  def whiteMoveAndRest(game: Game): Parser[Game] =  
    move(game, White) ^^ (moveAddingFunction =>  
      (game, moveAddingFunction,  
        nextOrEnd(blackMoveAndRest _) _)) into  
      addMoveAndContinue
```

# combinator parsing

## ChessNotationParser (2)

```
def nextOrEnd(nextParser: (Game) => Parser[Game])
  (game: Game): Parser[Game] =
  whitespace~>nextParser(game) | ""\s*$"".r~>success(game)

def blackMoveAndRest(game: Game): Parser[Game] =
  move(game, Black) ^^ (moveAddingFunction =>
    (game, moveAddingFunction,
      nextOrEnd(moveNumberAndRest _) _)) into
  addMoveAndContinue

def addMoveAndContinue(data: (Game, () => Option[String],
  (Game) => Parser[Game])) = {
  val (game, moveAddingFunction, nextParser) = data
  moveAddingFunction() match {
    case Some(msg) => failure(msg)
    case None => nextParser(game) }}}
```

# combinator parsing

## ChessNotationParser (3)

```
def move(game: Game, color: FigureColor) =  
  regularMove(game, color) |  
  castlingOOO(game, color) |  
  castlingOO(game, color)  
  
def castlingOOO(game: Game, color: FigureColor) =  
  "O-O-O"<~opt(info) ^^^  
  (() => game.addCastlingOOOByParser(color))  
  
def castlingOO(game: Game, color: FigureColor) =  
  "O-O"<~opt(info) ^^^  
  (() => game.addCastlingOOByParser(color))
```

# combinator parsing

## ChessNotationParser (4)

```
def regularMove(game: Game, color: FigureColor) =
  opt (figure (color)) ~ position ~ moveSeparator ~ position ~
  opt (promotionFigure (color)) ~ opt (info) ^^
  { case Some (fig) ~ pos1 ~ sep ~ pos2 ~ None ~ _ =>
    (() => game.addRegularMoveByParser (pos1, pos2,
      fig, sep, None))
  case None ~ pos1 ~ sep ~ pos2 ~ None ~ _ =>
    (() => game.addRegularMoveByParser (pos1, pos2,
      Pawn (color), sep, None))
  case None ~ pos1 ~ sep ~ pos2 ~ Some (fig) ~ _ =>
    (() => game.addRegularMoveByParser (pos1, pos2,
      Pawn (color), sep, Some (fig))) }
```

# combinator parsing

## ChessNotationParser (5)

```
def figure(color: FigureColor): Parser[Figure] =  
    king(color) | promotionFigure(color)  
  
def promotionFigure(color: FigureColor): Parser[Figure] =  
    queen(color) | rook(color) | bishop(color) | knight(color)  
  
def king(color: FigureColor): Parser[Figure] =  
    King (color).symbol ^^^ King (color)  
def queen(color: FigureColor): Parser[Figure] =  
    Queen (color).symbol ^^^ Queen (color)  
def rook(color: FigureColor): Parser[Figure] =  
    Rook (color).symbol ^^^ Rook (color)  
def bishop(color: FigureColor): Parser[Figure] =  
    Bishop(color).symbol ^^^ Bishop(color)  
def knight(color: FigureColor): Parser[Figure] =  
    Knight(color).symbol ^^^ Knight(color)
```

# combinator parsing ChessNotationParser (6)

```
def info: Parser[String] = "+"|"X"  
def moveSeparator: Parser[Boolean] = capture | noCapture  
def capture: Parser[Boolean] = (":"|"x") ^^^ true  
def noCapture: Parser[Boolean] = "-" ^^^ false  
def position =  
  column~row ^^ { case col~row => Position(col+row) }  
def column = "[a-h]".r  
def row = "[1-8]".r  
def whitespace = """\s+""".r  
override def skipWhitespace = false  
}
```