TODO: *What's the title?*

João Paulo Fernandes[1], Pedro Martins[2], Alberto Pardo[3], João Saraiva[4], Marcos Viera[3], and Tom Westerhout[5]

[1] LISP/Release – Universidade da Beira Interior, Portugal `jpf@di.ubi.pt`
[2] University of California, Irvine, USA `pribeiro@uci.edu`
[3] Universidad de la República, Uruguay `{pardo,mviera}@fing.edu.uy`
[4] Universidade do Minho, Portugal `saraiva@di.uminho.pt`
[5] Radboud University, The Netherlands `twesterhout@student.ru.nl`

Abstract. **TODO:** *What's the abstract?*

Keywords: Embedded Domain Specific Languages · Zipper data structure · Memoization · Attribute Grammars · Higher-Order Attribute Grammars · Functional Programming

1 Introduction

```
main :: IO ()
main = putStrLn "Hello world!"
```

2 Functional Zippers

The zipper data structure was originally conceived by Huet[1] to solve the problem of representing a tree together with a subtree that is the focus of attention, where that focus may move left, right, up or down the tree. Bla-bla-bla...

Application to binary trees...

```
data Tree a
  = Fork (Tree a) (Tree a)
  | Leaf !a
data Path a
  = Top
  | Left !(Path a) (Tree a)
  | TreeRight (Tree a) !(Path a)
data Zipper a = Zipper !(Path a) (Tree a)
```

Application to lists...

```
data Path a = Path [a] [a]
data Zipper a = Zipper !(Path a) [a]
```

Generic zipper...

An application of generic zipper that we will consider is embedding of attribute grammars.

3 Attribute Grammars

What attribute grammars are...

Repmin as two traversals...

Repmin as a circular program...

Repmin as an AG...

4 Related Work

5 Conclusion

Acknowledgements

References

References

1. Huet, G.: The zipper. Journal of functional programming **7**(5), 549–554 (1997)