Introduction to EPIGRAM

or: how to turn a proof assistant into an IDE for *programs*

James McKinna
based on joint work with
Conor McBride

Foundations/Intelligent Systems, Radboud Universiteit Nijmegen

“Proof Assistants” presentations, Nijmegen 2010-06-09

problems-as-types

"Curry-Howard-de Bruijn": proofs of a proposition yield computations of associated data

but for us…

(certified) programming is
interactive,
type-directed,
problem solving

problems-as-types

"Curry-Howard-de Bruijn": proofs of a proposition yield computations of associated data
but for us…

(certified) programming is
interactive,
type-directed,
problem solving

problems-as-types

"Curry-Howard-de Bruijn": proofs of a proposition yield computations of associated data
but for us…

(certified) programming is
interactive,
type-directed,
problem solving

past

- LEGO (1987–1999): proof checker for Calculus of Constructions; inductive types only added as primitive 1992; script (tactic) based interaction, leading to ProofGeneral; not tactical language;
- COQ (1985–present): proof checker for Calculus of Constructions; inductive types only added as primitive 1992; script (tactic) based interaction, leading to CoqIDE; tactical language Ltac added 1999
- ALF (???–???): superseded by Agda(2) (2001–present); ***direct*** editing of proof-terms, context-sensitive/type-directed

past/present

- - OLEG (McBride, 1995–2000): to do (dependently-typed) ***programming*** in LEGO is a pain: tactics don't let you see the term being constructed;
  - tactics for: declaring a new function definition
  - applying primitive recursors and case analysis
  - solving open leaf 'problem's
- EPIGRAM (1) (McBride, McKinna, 2000–2004): "problems-as-types"
  - programming problems become (labelled) types
  - OLEG tactics become primitives
  - case analysis and recursion become ***programmable***

  "An ALF-like editor with extensible pattern matching"
- *The View from the Left*, journal paper (2004) describing how to translate back to type theory, motivation, examples *etc.*

past/present

-
    - OLEG (McBride, 1995–2000): to do (dependently-typed) ***programming*** in LEGO is a pain: tactics don't let you see the term being constructed;
    - tactics for: declaring a new function definition
    - applying primitive recursors and case analysis
    - solving open leaf 'problem's
- EPIGRAM (1) (McBride, McKinna, 2000–2004): "problems-as-types"
    - programming problems become (labelled) types
    - OLEG tactics become primitives
    - case analysis and recursion become ***programmable***

  "An ALF-like editor with extensible pattern matching"
- *The View from the Left*, journal paper (2004) describing how to translate back to type theory, motivation, examples *etc.*

past/present

- - OLEG (McBride, 1995–2000): to do (dependently-typed) ***programming*** in LEGO is a pain: tactics don't let you see the term being constructed;
  - tactics for: declaring a new function definition
  - applying primitive recursors and case analysis
  - solving open leaf 'problem's
- EPIGRAM (1) (McBride, McKinna, 2000–2004): "problems-as-types"
  - programming problems become (labelled) types
  - OLEG tactics become primitives
  - case analysis and recursion become ***programmable***

  "An ALF-like editor with extensible pattern matching"
- *The View from the Left*, journal paper (2004) describing how to translate back to type theory, motivation, examples *etc.*

past/present

- - OLEG (McBride, 1995–2000): to do (dependently-typed) ***programming*** in LEGO is a pain: tactics don't let you see the term being constructed;
  - tactics for: declaring a new function definition
  - applying primitive recursors and case analysis
  - solving open leaf 'problem's
- EPIGRAM (1) (McBride, McKinna, 2000–2004): "problems-as-types"
  - programming problems become (labelled) types
  - OLEG tactics become primitives
  - case analysis and recursion become ***programmable***

  "An ALF-like editor with extensible pattern matching"
- *The View from the Left*, journal paper (2004) describing how to translate back to type theory, motivation, examples *etc.*

past/present

-
  - OLEG (McBride, 1995–2000): to do (dependently-typed) ***programming*** in LEGO is a pain: tactics don't let you see the term being constructed;
  - tactics for: declaring a new function definition
  - applying primitive recursors and case analysis
  - solving open leaf 'problem's
- EPIGRAM (1) (McBride, McKinna, 2000–2004): "problems-as-types"
  - programming problems become (labelled) types
  - OLEG tactics become primitives
  - case analysis and recursion become ***programmable***

  "An ALF-like editor with extensible pattern matching"
- *The View from the Left*, journal paper (2004) describing how to translate back to type theory, motivation, examples *etc.*

present: EPIGRAM (1)

- EPIGRAM is dead, long live EPIGRAM!
- EPIGRAM (1): codebase static; needs XEmacs21.4 (!); changes to the underlying haskell run-time mean death is inevitable, eventually
- but let's demo it anyway!
- the 'real' implementation of the VfL language is. . . Agda!

present: EPIGRAM (1)

- EPIGRAM is dead, long live EPIGRAM!
- EPIGRAM (1): codebase static; needs XEmacs21.4 (!); changes to the underlying haskell run-time mean death is inevitable, eventually
- but let's demo it anyway!
- the 'real' implementation of the VfL language is. . . Agda!

present: EPIGRAM (1)

- EPIGRAM is dead, long live EPIGRAM!
- EPIGRAM (1): codebase static; needs XEmacs21.4 (!); changes to the underlying haskell run-time mean death is inevitable, eventually
- but let's demo it anyway!
- the 'real' implementation of the VfL language is...Agda!

present: EPIGRAM (1)

- EPIGRAM is dead, long live EPIGRAM!
- EPIGRAM (1): codebase static; needs XEmacs21.4 (!); changes to the underlying haskell run-time mean death is inevitable, eventually
- but let's demo it anyway!
- the 'real' implementation of the VfL language is... Agda!

present: EPIGRAM (1)

- EPIGRAM is dead, long live EPIGRAM!
- EPIGRAM (1): codebase static; needs XEmacs21.4 (!); changes to the underlying haskell run-time mean death is inevitable, eventually
- but let’s demo it anyway!
- the ‘real’ implementation of the VfL language is. . . Agda!

alternative present

My former PhD student, Edwin Brady, has

- a theorem prover (Ivor),
- a programming language (Idris),
- a supercombinator compiler (Epic), and
- a haskell-like run-time system

It rocks: faster than Java, slower to within an order of magnitude of gcc (ICFP 2010)

present/future: EPIGRAM (2)

- underlying type theory insufficiently robust to deal with new phenomena
- co-induction on the same footing as induction
- *extensional* equality on function spaces
- *universes* for generic programming
- …

McBride leads a sizeable development team, based in Strathclyde/Tallinn/Nottingham: long-anticipated release of EPIGRAM (2)…soon (even if not Real Soon)

present/future: EPIGRAM (2)

- underlying type theory insufficiently robust to deal with new phenomena
- co-induction on the same footing as induction
- *extensional* equality on function spaces
- *universes* for generic programming
- ...

McBride leads a sizeable development team, based in Strathclyde/Tallinn/Nottingham: long-anticipated release of EPIGRAM (2)...soon (even if not Real Soon)

present/future: EPIGRAM (2)

- underlying type theory insufficiently robust to deal with new phenomena
- co-induction on the same footing as induction
- *extensional* equality on function spaces
- *universes* for generic programming
- …

McBride leads a sizeable development team, based in Strathclyde/Tallinn/Nottingham: long-anticipated release of EPIGRAM (2)…soon (even if not Real Soon)

present/future: EPIGRAM (2)

- underlying type theory insufficiently robust to deal with new phenomena
- co-induction on the same footing as induction
- *extensional* equality on function spaces
- *universes* for generic programming
- …

McBride leads a sizeable development team, based in Strathclyde/Tallinn/Nottingham: long-anticipated release of EPIGRAM (2)…soon (even if not Real Soon)

present/future: EPIGRAM (2)

- underlying type theory insufficiently robust to deal with new phenomena
- co-induction on the same footing as induction
- *extensional* equality on function spaces
- *universes* for generic programming
- …

McBride leads a sizeable development team, based in Strathclyde/Tallinn/Nottingham: long-anticipated release of EPIGRAM (2)… soon (even if not Real Soon)

present/future: EPIGRAM (2)

- underlying type theory insufficiently robust to deal with new phenomena
- co-induction on the same footing as induction
- *extensional* equality on function spaces
- *universes* for generic programming
- ...

McBride leads a sizeable development team, based in Strathclyde/Tallinn/Nottingham: long-anticipated release of EPIGRAM (2)...soon (even if not Real Soon)

Questions?