Encoding the Factorisation Calculus

Representing the Intensional in the Extensional

Reuben N. S. Rowe

EXPRESS/SOS, Madrid,
Monday 31st August 2015

Programming Principles, Logic & Verification Group
Department of Computer Science
University College London

Motivation

- We are interested in the relationship between the *Factorisation Calculus* and more familiar models of computation (viz. the λ-calculus)

- Factorisation Calculus is:
    - A combinatory rewrite system
    - A basis for a general theory of pattern matching
    - A model of *intensional* computations
        - cf. λ-calculus is an *extensional* theory of functions

The Factorisation Calculus

- Introduced by Jay and Given-Wilson (2011)
- A combinatory calculus comprising two *operators*: S and F
- We identify two 'special' sets of terms:
    - *Atomic* terms: unapplied operators, i.e. $\{\mathsf{S}, \mathsf{F}\}$
    - *Compound* terms: partially applied operators, e.g. $\mathsf{S}\,(\mathsf{F}\,\mathsf{F})\,\mathsf{S}$
- S is the familiar combinator from Combinatory Logic:

$$\mathsf{S}\,X\,Y\,Z \to X\,Z\,(Y\,Z)$$

- The F operator distinguishes atomic terms from compounds, also *factorising* the latter:

$$\begin{aligned} \mathsf{F}\,X\,M\,N &\to M && \text{if } X \text{ atomic} \\ \mathsf{F}\,(P\,Q)\,M\,N &\to N\,P\,Q && \text{if } P\,Q \text{ compound} \end{aligned}$$

The Factorisation Calculus: Important Properties

- It is *combinatorially complete*, since $\mathsf{F}\,\mathsf{F}$ represents K:

$$\mathsf{F}\,\mathsf{F} X\,Y \rightarrow X$$

- The *internal structure* of terms can be analysed, so:
    - *Intensionally distinct* terms can be distinguished
    - The equality predicate on normal forms is representable
- Compare with Combinatory Logic (and so λ-calculus):
    - Equality of arbitrary normal forms *not* representable
    - Factorisation of combinators is not representable
        - e.g. there is no CL term T such that $T\,(\mathsf{S}\,\mathsf{K} X) \rightarrow^{*} X$ for any X

Characterising Expressiveness: Structure Completeness

- Consider using arbitrary linear normal terms as patterns for matching, e.g.

$$\{\mathsf{S}\,M\,(\mathsf{F}\,N\,\mathsf{S})/\mathsf{S}\,x\,(\mathsf{F}\,y\,\mathsf{S})\} = [x \mapsto M, y \mapsto N]$$
$$\{\mathsf{S}\,M\,N/\mathsf{F}\,x\,y\} = \mathsf{fail}$$

- A *case* $\mathcal{G}(P) = M$ (for pattern P and term M) defines a *symbolic function* $\mathcal{G}$ on combinators:

$$\mathcal{G}(U) = \begin{cases} \sigma(M) & \text{if } \{U/P\} = \sigma \\ \text{some default term} & \text{if } \{U/P\} = \mathsf{fail} \end{cases}$$

- A combinatory calculus is *structure complete* if every case $\mathcal{G}$ is represented by some term G, i.e. $G\,U =_\beta \mathcal{G}(U)$ for all U

Theorem: Factorisation Calculus is structure complete

- Jay and Given-Wilson use structural completeness as a way to characterise the expressive power of Factorisation Calculus
- Factorisation Calculus is structurally complete; CL isn't
    - **Conclusion:** Factorisation Calculus is *more* expressive
- There are symbolic functions representable in Factorisation Calculus but not in CL
    - e.g. Factorisation, equality of normal forms
- So, does the Factorisation Calculus *compute* more things?
    - The standard way to answer this is by showing the (non-) existence of an *encoding*

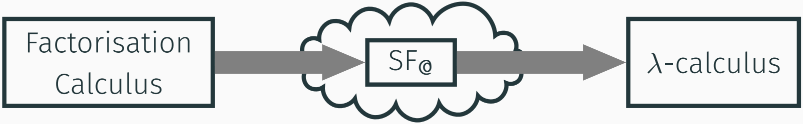
Overview of our Encoding

Factorisation Calculus

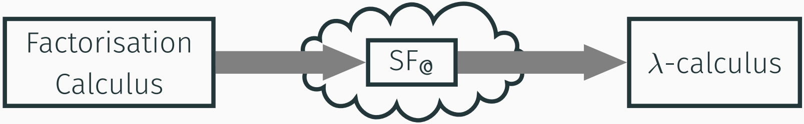
λ-calculus

Overview of our Encoding

- We use a construction due to Berrarducci and Böhm which encodes certain types of term rewriting system in λ-calculus



- We use a construction due to Berrarducci and Böhm which encodes certain types of term rewriting system in λ-calculus
- We show how to implement Factorisation Calculus as a suitable rewrite system



- We use a construction due to Berrarducci and Böhm which encodes certain types of term rewriting system in λ-calculus
- We show how to implement Factorisation Calculus as a suitable rewrite system
- The encoding is *faithful*
    - i.e. preserves both reduction and termination

The Berrarducci-Böhm Representation Result

- A rewrite system $\mathcal{R}$ over a signature Σ is *canonical* if:
  - $\Sigma = \Sigma_C \uplus \Sigma_F$ with every rewrite rule of the form:

$$f(c(x_1, \ldots, x_n), y_1, \ldots, y_m) \to t \qquad (c \in \Sigma_C \text{ and } f \in \Sigma_F)$$

  - That is, Σ comprises *constructors* Σ_C and *programs* Σ_F
- Berrarducci and Böhm (1992) showed that every such $\mathcal{R}$ has a *representation* $\phi_{\mathcal{R}}$ in λ-calculus, i.e.

$$t \to_{\mathcal{R}} t' \Rightarrow \phi_{\mathcal{R}}(t) \to_{\lambda} \phi_{\mathcal{R}}(t')$$

- Moreover, for closed terms, $\phi_{\mathcal{R}}$ preserves strong normalisation

A Canonical Rewrite System for Factorisation

- Application is a *constructor-driven program*:

$$\mathsf{app}\,(\mathsf{S_0}, x) \to \mathsf{S_1}\,(x) \qquad \mathsf{app}\,(\mathsf{F_0}, x) \to \mathsf{F_1}\,(x)$$

$$\mathsf{app}\,(\mathsf{S_1}\,(x), y) \to \mathsf{S_2}\,(x, y) \quad \mathsf{app}\,(\mathsf{F_1}\,(x), y) \to \mathsf{F_2}\,(x, y)$$

$$\mathsf{app}\,(\mathsf{S_2}\,(x, y), z) \to \mathsf{app}\,(\mathsf{app}\,(x, z), \mathsf{app}\,(y, z))$$

$$\mathsf{app}\,(\mathsf{F_2}\,(x, y), z) \to \mathsf{factorise}\,(x, y, z)$$

- Factorisation is a program too:

$$\mathsf{factorise}\,(\mathsf{S_0}, y, z) \to y$$

$$\mathsf{factorise}\,(\mathsf{S_1}\,(q), y, z) \to \mathsf{app}\,(\mathsf{app}\,(z, \mathsf{S_0}), q)$$

$$\mathsf{factorise}\,(\mathsf{S_2}\,(p, q), y, z) \to \mathsf{app}\,(\mathsf{app}\,(z, \mathsf{app}\,(\mathsf{S_0}, p)), q)$$

+ symmetric rules for $\mathsf{F_0}, \mathsf{F_1}, \mathsf{F_2}$

Faithfully Encoding Factorisation Calculus

- The translation into our rewrite system $\mathsf{SF}_@$ is straightforward:

$$[\![\mathsf{S}]\!]_@ = \mathsf{S}_0 \qquad [\![\mathsf{F}]\!]_@ = \mathsf{F}_0 \qquad [\![MN]\!]_@ = \mathsf{app}\,([\![M]\!]_@,\, [\![N]\!]_@)$$

- We have shown that $[\![\cdot]\!]_@$ also preserves reduction and strong normalisation
- Thus $[\![\cdot]\!]_\lambda = \phi_{\mathsf{SF}_@} \circ [\![\cdot]\!]_@$ is a faithful encoding of Factorisation Calculus in λ-calculus

- Our encoding is *compositional*:

$$[\![MN]\!]_\lambda = \phi_{\mathsf{SF}_@}(\mathsf{app}\,([\![M]\!]_@,\, [\![N]\!]_@)) = \phi_{\mathsf{SF}_@}(\mathsf{app})\, [\![M]\!]_\lambda\, [\![N]\!]_\lambda$$

- It is not a *homomorphism* … however:
    - It looks like an instance of an *applicative morphism* (Longley)
- The ‘classical’ interpretation is that our encoding constitutes an equivalence
    - We need to look further to understand the notion of expressiveness captured by structural completeness

Felleisen's Framework for Comparing Expressiveness

- Felleisen (1991) defined a formal expressiveness criterion based on the concept of *eliminability* in logic
  - A logic $\mathcal{L}$ is more expressive than logic $\mathcal{L}'$ if:
    1. $\mathcal{L}$ is a *conservative extension* of $\mathcal{L}'$
    2. $\mathcal{L}$ contains a *non-eliminable* symbol
- By analogy, language L is more expressive than language L' if:
  1. it is a superset of L'
  2. it contains some construct which cannot be translated to L' using a *macro*
- Consider *SKF*-calculus as a more expressive superset of CL, since F is not representable using S and K (i.e. as a macro)

Boker & Dershowitz's Abstract Framework (2009)

- Take computational models to be pairs: a (semantic) domain and a set of functions (the *extensionality*)
    - A larger extensionality = more expressive
- Maps between domains induce *simulations* (i.e. encodings)
    - But some maps allow for simulations of strictly larger extensionalities!
- Different restrictions on the mappings between domains yield notions of (in)equivalence of varying strength
- Our encoding shows a weak form of equivalence
- Existing results would seem to imply inequivalence at a stronger level

Conclusions & Future Work

- Factorisation Calculus is a recent fundamental model of computation with expressive intensional properties
- We have demonstrated the existence of a faithful encoding of the Factorisation Calculus in the λ-calculus
- Our results point towards a nuanced relationship between the two paradigms which requires further investigation
- We believe that research into the denotational semantics of Factorisation Calculus is a logical next step