A Calculus for Resource Relationships

Robert Atkey

LFCS, Division of Informatics, University of Edinburgh

bob.atkey@ed.ac.uk

Separation

- A function which analyses statistical data:

$$\text{analyse} : \mathit{Data}, \mathit{Data} \to \mathit{Result}$$

- For the result to be valid the two arguments must not refer to data from overlapping sources.

- Expressible in (affine) $\alpha\lambda$-calculus:

$$\text{analyse} : \mathit{Data} * \mathit{Data} \to \mathit{Result}$$

- 3 pairs to be run in sequence, over 4 items:

$$(\text{analyse}(a * b), \text{analyse}(b * c), \text{analyse}(c * d))$$

Expressing Separation in the affine $\alpha\lambda$-calculus

- λ-calculus corresponding to BI logic.
- Affine $\alpha\lambda$-calculus has two product types:
  - $A \times B$: normal pairing, allowing sharing of resources;
  - $A * B$: pairing, prohibiting sharing.
- In contexts these are replaced by ";" and ",":

$$(a : A; (b : B, c : C)) \vdash e : E$$

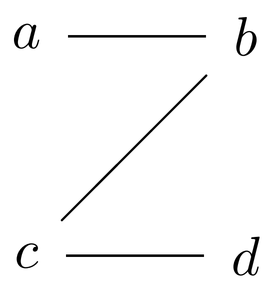
- Program e requires (at least) that b and c do not share.
- "Affine" allows imposition of stronger pre-conditions (Dereliction):

$$(a : A, (b : B, c : C)) \vdash e : E$$

Separation

$$(\text{analyse}(a * b), \text{analyse}(b * c), \text{analyse}(c * d))$$

- How to describe the required separation?
  - a separate from b;
  - b separate from c;
  - c separate from d



- Not directly expressible in $\alpha\lambda$;
  - Attempt: $(a : Data \times d : Data) * (b : Data * c : Data)$

Pulling out the Separation Constraints

- Basic Idea: Distinction between context members and **relationships** between them.
- Express example as:

$$[a\#b, b\#c, c\#d](a : Data, b : Data, c : Data, d : Data) \vdash ...$$

- Allowing nesting of contexts:

$$[1\#2]([2\#3](a : A, b : B, c : C), d : D) \vdash \dots$$

- Similar bunching of contexts to BI/$\alpha\lambda$-calculus.

Structural Rules

- **Constraint preserving transformations**
  give
  Structural rules

$$\Delta \Rightarrow \Delta' \quad \text{gives} \quad \frac{\Gamma(\Delta') \vdash e : A}{\Gamma(\Delta) \vdash e : A}$$

- (Un)Flattening of nested contexts:

$$[1\#2]\,([2\#3](a,b,c),d) \Leftrightarrow [1\#4,2\#4,3\#4,2\#3](a,b,c,d)$$

- Removal of constraints, when $\mathrm{S} \subseteq \mathrm{S}'$:

$$\mathrm{S}'(\Gamma_1,\ldots,\Gamma_n) \Rightarrow \mathrm{S}(\Gamma_1,\ldots,\Gamma_n)$$

- Permutation

Weakening and Contraction

- We may forget about parts of the context (and their relationships):

$$[1\#2, 2\#3](a, b, c) \Rightarrow [1\#2](a, b)$$

- Contraction preserves the correct separation:

$$\mathrm{S}(a, b, c) \Rightarrow [](\mathrm{S}(a, b, c), \mathrm{S}(a', b', c'))$$

- **But**:

$$\mathrm{S}(a, b, c) \not\Rightarrow [1\#2](\mathrm{S}(a, b, c), \mathrm{S}(a', b', c'))$$

Products

$$\frac{\Gamma_1 \vdash e_1 : A_1 \qquad \ldots \qquad \Gamma_n \vdash e_n : A_n}{\mathrm{S}(\Gamma_1, \ldots, \Gamma_n) \vdash \mathrm{S}(e_1, \ldots, e_n) : \mathrm{S}(A_1, \ldots, A_n)}$$

$$\frac{\Gamma \vdash e_1 : \mathrm{S}(A_1, \ldots, A_n) \qquad \Delta(\mathrm{S}(x_1 : A_1, \ldots, x_n : A_n)) \vdash e_2 : B}{\Delta(\Gamma) \vdash \mathrm{let}\ \mathrm{S}(x_1, \ldots, x_n) = e_1\ \mathrm{in}\ e_2 : B}$$

Functions

$$\frac{\mathrm{S}(\Gamma, x_1 : A_1, \ldots, x_n : A_n) \vdash e : B}{\Gamma \vdash \lambda^{\mathrm{S}}(x_1, \ldots, x_n).e : A_1, \ldots, A_n \xrightarrow{\mathrm{S}} B}$$

$$\frac{\Gamma \vdash f : A_1, \ldots, A_n \xrightarrow{\mathrm{S}} B \qquad \Delta_1 \vdash a_1 : A_1 \qquad \ldots \qquad \Delta_n \vdash a_n : A_n}{\mathrm{S}(\Gamma, \Delta_1, \ldots, \Delta_n) \vdash f @_{\mathrm{S}}(a_1, \ldots, a_n) : B}$$

Encoding affine $\alpha\lambda$-calculus

- Encoding of affine $\alpha\lambda$-calculus:
  - $(A \times B)^\dagger = [](A, B)$
  - $(A * B)^\dagger = [1\#2](A, B)$
  - $(A \to B)^\dagger = A \xrightarrow{[]} B$
  - $(A -\!\!* \; B)^\dagger = A \xrightarrow{[1\#2]} B$
- Associativity is given by flattening and unflattening:

$$\begin{aligned} \mathrm{S}(\mathrm{S}(A, B), C) &\Leftrightarrow \mathrm{S}\{\mathrm{S}/1\}(A, B, C) \\ &\Leftrightarrow \mathrm{S}\{\mathrm{S}/2\}(A, B, C) \Leftrightarrow \mathrm{S}(A, \mathrm{S}(B, C)) \end{aligned}$$

Categorical Semantics

- Family of functors for each separation relation:

$$\underline{\mathrm{S}} : \mathcal{C}^{|\mathrm{S}|} \to \mathcal{C}$$

  – Model contexts and product types.

- Flattening: $\alpha_{\mathrm{S}(A,\mathrm{S}'(B),C)} : \underline{\mathrm{S}}(\overrightarrow{A}, \underline{\mathrm{S}'}(\overrightarrow{B}), \overrightarrow{C}) \cong \underline{\mathrm{S}\{\mathrm{S}'\}}(\overrightarrow{A}, \overrightarrow{B}, \overrightarrow{C})$

- Coherence conditions, including:

$$\begin{tikzcd}
\underline{\mathrm{S}}(\overrightarrow{A}, \underline{\mathrm{S}'}(\overrightarrow{B}), \overrightarrow{C}, \underline{\mathrm{S}''}(\overrightarrow{D}), \overrightarrow{E}) \arrow[r, "\alpha"] \arrow[d, "\alpha"'] & \underline{\mathrm{S}\{\mathrm{S}'\}}(\overrightarrow{A}, \overrightarrow{B}, \overrightarrow{C}, \underline{\mathrm{S}''}(\overrightarrow{D}), \overrightarrow{E}) \arrow[d, "\alpha"] \\
\underline{\mathrm{S}\{\mathrm{S}''\}}(\overrightarrow{A}, \underline{\mathrm{S}'}(\overrightarrow{B}), \overrightarrow{C}, \overrightarrow{D}, \overrightarrow{E}) \arrow[r, "\alpha"] & \underline{\mathrm{S}\{\mathrm{S}'\}\{\mathrm{S}''\}}(\overrightarrow{A}, \overrightarrow{B}, \overrightarrow{C}, \overrightarrow{D}, \overrightarrow{E})
\end{tikzcd}$$

Categorical Semantics

- Structural rules modelled by natural transformations;
- $\underline{[]_0}()$ is the terminal object;
- Function types are usual right adjoints to products.
- Coherence conditions apply, to give:

  Theorem 1 *If π_1 and π_2 are derivations of $\Gamma \vdash e : A$ then $[\![\pi_1]\!] = [\![\pi_2]\!]$.*
- Also usual soundness and completeness.
- Complicated by the syntax-less structural rules
  - (in particular the commuting conversion rule)

Resource Semantics

- Functor category semantics
- Partially ordered set R of "resources" with:
  - $r_1 + r_2$, (joins), for combination of resources;
  - A separation relation between resources $r_1 \# r_2$:
    * symmetric;
    * If $r_1 \# r_2$ and $r'_1 \sqsubseteq r_1$ and $r'_2 \sqsubseteq r_2$ then $r'_1 \# r'_2$;
    * $r \#(r_1 + r_2)$ iff $r \# r_1$ and $r \# r_2$.
  - Example: sets of sources/memory locations.
- Types are functors from R to Set;
- Interpret product types using Day's constructions in Set^R.

Variation: Beyond Separation [Work in progress]

- Non-symmetric relationships such as allowable information flow:
  - Assume a set $\mathcal{S}$ of security identities
  - A relation $\triangleright \subseteq \mathcal{S} \times \mathcal{S}$ for allowable flow
  - Possible worlds are sets of security tokens, $W \subseteq \mathcal{S}$.
  - $W_1 \triangleright W_2$ if for all $w_1 \in W_1$, $w_2 \in W_2$, $w_1 \triangleright w_2$.
  - Combination by union.
- Judgements have non-symmetric relations:

$$[1 \triangleright 2](i : int, s : stream) \vdash put(i, s) : stream$$

- Problem: unwanted distributivity of sum types over non-separating product.

Conclusions and Further Work

- This calculus:
  - Has a coherent, sound and complete categorical semantics;
  - Has a semantics modelling resources and their relationships;
  - Can express more patterns of separation; and
  - Is more flexible wrt. changes than the $\alpha\lambda$-calculus.
- Further work:
  - Resource-insensitive types;
  - Integrating number-of-uses/destruction/global separation;
  - More on relationship to $\alpha\lambda$ (Conservativity?);
  - Completeness of functor category semantics;
  - More realistic examples.