A Calculus for Resource Relationships

Robert Atkey

LFCS, Division of Informatics, University of Edinburgh

`bob.atkey@ed.ac.uk`

This research was supported by the MRG project (IST-2001-33149) which is funded by the EC under the FET proactive initiative on Global Computing.

Expressing Separation in Affine $\alpha\lambda$-calculus

- 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

- A function which runs jobs in parallel:

$$\text{runPar} : Job, Job \rightarrow PJob$$

- To run them in parallel we require that the arguments do not access the same memory.
- Expressible in (affine) $\alpha\lambda$-calculus:

$$\text{runPar} : Job * Job \rightarrow PJob$$

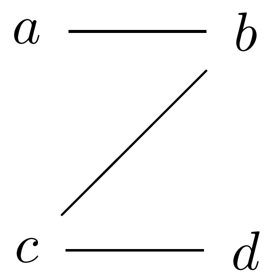
- 3 pairs to be run in sequence, over 4 jobs:

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

Separation

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

- 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: $(a : Job \times d : Job) * (b : Job * c : Job)$

Pulling out the Separation Constraints

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

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

- Allowing nesting of contexts:

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

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

Structural Rules

- **Constraint preserving transformations**
  give
  Structural rules

$$\Gamma \Rightarrow \Delta \quad \text{gives} \quad \frac{\Delta \vdash e : A}{\Gamma \vdash e : A} \tag{1}$$

- (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 $\mathbf{S} \subseteq \mathbf{S}'$:

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

- 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:

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

- **But**:

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

Tuples and Functions

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

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

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

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

Encoding affine $\alpha\lambda$-calculus

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

$$\mathbf{S}(\mathbf{S}(A, B), C) = \mathbf{S}\{\mathbf{S}/1\}(A, B, C) = \mathbf{S}(A, \mathbf{S}(B, C))$$

Semantics

- Possible world semantics
- Partially ordered set R of worlds (resources) with:
  - $r_1 \cup r_2$, 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 $r_1' \# r_2'$;
    * $r \# (r_1 \cup r_2)$ iff $r \# r_1$ and $r \# r_2$.
  - Example: sets of memory locations.
- Interpret types using Day's constructions in Set^R;
- Instance of a general categorical semantics.

Variation: Beyond Separation

- Extend to domains other memory regions;
- Non-symmetric relationships such as allowable information flow:
  - Assume a set $\mathcal{S}$ of security tokens
  - A relation $\rhd \subseteq \mathcal{S} \times \mathcal{S}$ for allowable flow
  - Possible worlds are sets of security tokens, $W \subseteq \mathcal{S}$.
  - $W_1 \rhd W_2$ if forall $w_1 \in W_1$, $w_2 \in W_2$, $w_1 \rhd w_2$.
  - Combination by union.
- Judgements have non-symmetric relations:

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

Variation: Separation and Number-of-uses

- Take inspiration from Linear Logic.
- Remove weakening and contraction;
- Add a new context former !:
  - $\mathbf{S}(\Gamma, !\Delta, \Theta)$
  - Reintroduce contraction and weakening on !'d bunches;
  - Add structural rules:

$$\frac{\Gamma(\Delta) \vdash e : A}{\Gamma(!\Delta) \vdash e : A} \qquad \frac{\Gamma(!!\Delta) \vdash e : A}{\Gamma(!\Delta) \vdash e : A} \qquad \frac{\Gamma(!(\Delta, \Delta')) \vdash e : A}{\Gamma(!\Delta, !\Delta') \vdash e : A}$$

- Also term syntax for introducing and eliminating types $!A$.
- Can do the same with $\alpha\lambda$, but lose flexibility:

$$A * (B \times C) \not\to (A * B) \times C$$

Conclusions and Further Work

- This calculus:
  - Has a semantics modelling resources and their relationships;
  - Can express more patterns of separation; and
  - Is more flexible wrt. changes in the structural rules than $\alpha\lambda$-calculus.
- Further work:
  - Resource-insensitive types;
  - Different ways of integrating number-of-uses/destruction;
  - More on relationship to $\alpha\lambda$:
    - Conservativity?