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:

\[ \text{runPar} : \text{Job}, \text{Job} \to P\text{Job} \]

• To run them in parallel we require that the arguments do not access the same memory.

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

\[ \text{runPar} : \text{Job} * \text{Job} \to P\text{Job} \]

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

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

\[(\text{runPar}(a \ast b), \text{runPar}(b \ast c), \text{runPar}(c \ast d))\]

- 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: \((a : \text{Job} \times d : \text{Job}) \ast (b : \text{Job} \ast c : \text{Job})\)
Pulling out the Separation Constraints

• Basic Idea: Distinction between context members and relationships between them.

• Express example as:

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

• Allowing nesting of contexts:

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

• Similar bunching of contexts to BI/αλ-calculus.
Structural Rules

- Constraint preserving transformations give
  Structural rules

  \[
  \frac{\Delta \vdash e : A}{\Gamma \Rightarrow \Delta \quad \text{gives} \quad \Gamma \vdash e : A}
  \] (1)

- (Un)Flattening of nested contexts:

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

- Removal of constraints, when \(S \subseteq S'\):

  \(S'(\Gamma_1, \ldots, \Gamma_n) \Rightarrow S(\Gamma_1, \ldots, \Gamma_n)\)

- 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:

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

- But:

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

\[
\begin{align*}
\Gamma_1 & \vdash e_1 : A_1 \quad \ldots \quad \Gamma_n & \vdash e_n : A_n \\
S(\Gamma_1, \ldots, \Gamma_n) & \vdash S(e_1, \ldots, e_n) : S(A_1, \ldots, A_n) \\
\Gamma & \vdash e_1 : S(A_1, \ldots, A_n) \quad \Delta(S(x_1 : A_1, \ldots, x_n : A_n)) & \vdash e_2 : B \\
\Delta(\Gamma) & \vdash \text{let } S(x_1, \ldots, x_n) = e_1 \text{ in } e_2 : B \\
S(\Gamma, x_1 : A_1, \ldots, x_n : A_n) & \vdash e : B \\
\Gamma & \vdash \lambda S(x_1, \ldots, x_n).e : A_1, \ldots, A_n \rightarrow S B \\
\Gamma & \vdash f : A_1, \ldots, A_n \rightarrow S B \quad \Delta_1 & \vdash a_1 : A_1 \quad \ldots \quad \Delta_n & \vdash a_n : A_n \\
S(\Gamma, \Delta_1, \ldots, \Delta_n) & \vdash f @ S(a_1, \ldots, a_n) : B
\end{align*}
\]
Encoding affine $\alpha\lambda$-calculus

- Encoding of affine $\alpha\lambda$-calculus:
  - $(A \times B) \uparrow = \emptyset(A, B)$
  - $(A \times B) \uparrow = [1\#2](A, B)$
  - $(A \rightarrow B) \uparrow = A \xrightarrow{\downarrow} B$
  - $(A \rightarrow B) \uparrow = A \xrightarrow{[1\#2]} B$

- Associativity is given by flattening and unflattening:

$$S(S(A, B), C) = S\{S/1\}(A, B, C) = S(A, S(B, C))$$
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 $r'_1 \subseteq r_1$ and $r'_2 \subseteq r_2$ 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 $\text{Set}^R$;

- 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 $S$ of security tokens
  - A relation $\triangleright\subseteq S \times S$ for allowable flow
  - Possible worlds are sets of security tokens, $W \subseteq S$.
  - $W_1 \triangleright W_2$ 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:

  $[1 \triangleright 2](i : int, s : stream) \vdash put(i, s) : stream$
Variation: Separation and Number-of-uses

- Take inspiration from Linear Logic.
- Remove weakening and contraction;
- Add a new context former !:
  - $S(\Gamma, !\Delta, \Theta)$
  - Reintroduce contraction and weakening on !’d bunches;
  - Add structural rules:
    \[
    \begin{align*}
    \Gamma(\Delta) & \vdash e : A & \Gamma(!\Delta) & \vdash e : A & \Gamma(!(\Delta, \Delta')) & \vdash e : A \\
    \Gamma(!\Delta) & \vdash e : A & \Gamma(!\Delta) & \vdash e : A & \Gamma(!\Delta, !\Delta') & \vdash e : A
    \end{align*}
    \]

- Also term syntax for introducing and eliminating types !A.
- Can do the same with $\alpha\lambda$, but lose flexibility:
  \[A \times (B \times C) \leftrightarrow (A \times B) \times C\]
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?