A Categorical Semantics of Resource Separation

Robert Atkey

LFCS Edinburgh, Mayfield Rd, Edinburgh EH9 3JZ, UK
bob.atkey@ed.ac.uk

1 Introduction

We describe an extension of Bunched Typing, or the $\alpha\lambda$-calculus [1, 2], intended to increase the flexibility of the system and to suggest further possible routes for investigation of bunched type systems. The system, called λ_{sep}, extends the two binary context formers of the $\alpha\lambda$-calculus to n-place context formers with arbitrary *separation* relations between their members. These relations express pairwise separation constraints on the resources used by the members of the context. The system then allows a distinction between the combination of resources and the expression of constraints between resources. This system can express constraints that the $\alpha\lambda$-calculus cannot, and can express them in a clearer way.

The system has a categorical semantics which is used to fix the requirements for sound and coherent models.

2 Separation Typing

We record the relationships between members of the context by attaching a *separation relation* to the context:

$$[1\#2](a : A, b : B, c : C)$$

This is a context representing a collection of items a, b and c, where a and b are guaranteed to be using separate resources. We can nest contexts:

$$[]([1\#2](a : A, b : B), c : C)$$

The structural rules of the system, i.e. the ways in which contexts may be manipulated, are derived from separation preserving manipulations of the underlying collections of objects. The structural rules of the system are shown in figure 1. The notation $\mathrm{S}\{\mathrm{S}'\}$ denotes substitution of one separation relation into another.

The typing rules are shown in figure 2. The structural rules from figure 1 are embedded in the system via the Struct rule. The connective rules essentially mirror the structure of the contexts to the right hand side of the turnstile. The SI and SE rules introduce and eliminate tuple types, one for each possible separation relation. Note that the tuple type (and context) $[]_0()$ acts as a unit type and the type $[]_1(A)$ acts identically to A. The $\xrightarrow{\mathrm{S}}$I and $\xrightarrow{\mathrm{S}}$E rules introduce and eliminate multi argument functions. The multiple arguments are necessary to retain the correct relationships between the abstracted variables and the remaining context.

Rule	Effect on the context	$\phi(e)$
Flatten	$\mathrm{S}(\overrightarrow{\Gamma}, \mathrm{S}'(\overrightarrow{\Delta}), \overrightarrow{\Gamma'}) \Leftrightarrow \mathrm{S}\{\mathrm{S}'\}(\overrightarrow{\Gamma}, \overrightarrow{\Delta}, \overrightarrow{\Gamma'})$	e
S-Weak ($\mathrm{S}' \subseteq \mathrm{S}$)	$\mathrm{S}(\overrightarrow{\Gamma}) \Rightarrow \mathrm{S}'(\overrightarrow{\Gamma})$	e
Contr ($\Gamma \equiv_\alpha \Gamma'$)	$\Gamma \Rightarrow []_2(\Gamma, \Gamma')$	$e[\mathrm{v}(\Gamma)/\mathrm{v}(\Gamma')]$
Weak	$\Gamma \Rightarrow []_0()$	e
Perm	$\mathrm{S}(\overrightarrow{\Gamma}) \Leftrightarrow \sigma\mathrm{S}(\sigma\overrightarrow{\Gamma})$	e

Fig. 1. Structural Rules

$$\frac{}{x : A \vdash x : A}\ (\textsc{Id}) \qquad \frac{\Gamma(\Delta') \vdash e : A \qquad \Delta \overset{\phi}{\Rightarrow} \Delta'}{\Gamma(\Delta) \vdash \phi(e) : A}\ (\textsc{Struct})$$

$$\frac{\overline{\Gamma \vdash e : A}}{\mathrm{S}(\overrightarrow{\Gamma}) \vdash \mathrm{S}(\overrightarrow{e}) : \mathrm{S}(\overrightarrow{A})}\ (\mathrm{SI}) \qquad \frac{\Gamma \vdash e : \mathrm{S}(\overrightarrow{A}) \qquad \Delta(\mathrm{S}(\overline{x : A})) \vdash e' : B}{\Delta(\Gamma) \vdash \mathrm{let}\ \mathrm{S}(\overrightarrow{x}) = e\ \mathrm{in}\ e' : B}\ (\mathrm{SE})$$

$$\frac{\mathrm{S}(\Gamma, \overline{x : A}) \vdash e : B}{\Gamma \vdash \lambda_{\mathrm{S}}(\overrightarrow{x}).e : \overrightarrow{A} \xrightarrow{\mathrm{S}} B}\ (\xrightarrow{\mathrm{S}}\mathrm{I}) \qquad \frac{\Gamma \vdash f : \overrightarrow{A} \xrightarrow{\mathrm{S}} B \qquad \overline{\Delta \vdash a : A}}{\mathrm{S}(\Gamma, \overrightarrow{\Delta}) \vdash f @_{\mathrm{S}}(\overrightarrow{a}) : B}\ (\xrightarrow{\mathrm{S}}\mathrm{E})$$

Fig. 2. The rules of λ_{sep}

3 Semantics

The semantics of the λ_{sep} system are primarily given by a categorical interpretation. This fixes the structure we require and provides generic coherence and soundness results for models.

We require a category $\mathcal{C}$ with endofunctors $\underline{\mathrm{S}} : \mathcal{C}^{|\mathrm{S}|} \to \mathcal{C}$ for each separation relation S. These model the product types and contexts of the syntax. The structural rules are natural transformations between these functors, subject to several coherence conditions. Function types are modelled as functors, $[A_1, \ldots, A_n \xrightarrow{S} -] : \mathcal{C} \to \mathcal{C}$, right adjoint to the $\underline{\mathrm{S}}$ functors. The semantics extends a map $\mathcal{I}$ of primitive types and operators to objects and arrows in $\mathcal{C}$ to a map $[\![-]\!]_{\mathcal{I}}$ from derivations to arrows in $\mathcal{C}$. We have the following results:

Theorem 1 (Coherence). *If π_1 and π_2 are two derivations of the judgement $\Gamma \vdash e : A$ then $[\![\pi_1]\!]_{\mathcal{I}} = [\![\pi_2]\!]_{\mathcal{I}}$.*

Theorem 2 (Soundness and Completeness). *$\Gamma \vdash e_1 = e_2 : A$ if and only if in all categorical models $[\![\Gamma \vdash e_1 : A]\!]_{\mathcal{I}} = [\![\Gamma \vdash e_2 : A]\!]_{\mathcal{I}}$.*

References

1. P. W. O'Hearn. On bunched typing. *Journal of Functional Programming*, 13(4):747–796, 2003.
2. D. J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*, volume 26 of *Applied Logic Series*. Kluwer Academic Publishers, 2002.