London Mathematical Society ISSN 1461–1570

CATEGORICAL MODELS OF SYNTACTIC CONTROL OF INTERFERENCE REVISITED, REVISITED

GUY MCCUSKER

Abstract

The question of what categorical structure is required to give semantics to O'Hearn *et al.*'s type system *Syntactic Control of Interference Revisited* (SCIR) is considered. The previously proposed notion of bireflective model is rejected as being too restrictive to accommodate important concrete models based on game semantics and object spaces; furthermore it is argued that the existing proof-sketch of the important property of coherence for these models is incorrect. A new, more general notion of model is proposed and the coherence property proved.

1. *Introduction*

Interference lies at the heart of imperative programming. Two program phrases are said to interfere if the execution of one of them may have an impact on the later execution of another. Of course, this is the very purpose of the assignment statement: to have impact on the result of later dereferencing of the assigned variable. However, when interfering phrases are combined in more sophisticated ways, such as through procedure calls, reasoning about programs becomes difficult. Indeed, the rule for procedure calls in Reynolds's Specification Logic for Algol-like languages [**11**] has a side condition requiring that the procedure and its arguments are non-interfering.

It is therefore important to be aware of which program phrases potentially interfere and which do not. The simplest source of interference is through shared identifiers: if two phrases both refer to program variable `x`, there is a good chance they interfere. However, because of aliasing, shared identifiers are not the only source of interference: program phrases with completely disjoint sets of free identifiers may nonetheless interfere if those identifiers become bound to some common resource. In *Syntactic Control of Interference* [**10**], Reynolds proposes a discipline of programming in a higher-order imperative language which eliminates this difficulty, reducing interference once again to the presence of shared identifiers. This renders all interference syntactically obvious, and thus allows one to use the reasoning principle from Specification Logic more readily.

The key idea behind the Syntactic Control of Interference (SCI) system is to eliminate aliasing at source. In the Algol-like language under consideration, the only way to bind an identifier to a pre-existing resource is through a procedure call: $(\lambda x.M)N$. The only other binding construct in the language is the allocation of fresh local variables, which by definition cannot introduce aliasing.

Received 13 October 2006, revised 27 February 2007; *published* 1 June 2007.
2000 Mathematics Subject Classification 68Q55, 18C50

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

In the program phrase above, the identifier x in M becomes bound to N. Suppose now that M and N have a free identifier y in common. After this binding takes place, x and y within M may interfere, so prior reasoning about M which may have assumed x and y to be non-interfering resources is rendered incorrect.

Reynolds eliminates this difficulty by insisting that procedures and their arguments should be non-interfering. Thus it is safe to assume that distinct identifiers are non-interfering, and therefore it is easy to establish the non-interference property! In modern terms, this restriction amounts to the use of a *multiplicative* typing rule for application:

$$\frac{\Gamma \vdash M : A \to B \quad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B}.$$

Here the contexts (set of free variables) of the function and its argument, Γ and Δ, are required to be disjoint.

An important extension to these ideas, also proposed by Reynolds, is the recognition of *passive types*. Certain types contain only phrases which have no effect on the store; for example, the type of arithmetic expressions. Identifiers within such phrases may be freely shared without risk of interference. Reynolds described this relaxed system, but noted that the condition on programs to be admitted in the system was not closed under reduction. Though this poses no problems in practice, it does lead to a troubling situation: a syntactically safe program can reduce to an apparently unsafe one, although no harm will in fact be done.

In *Syntactic Control of Interference Revisited* [8], O'Hearn *et al.* gave a type system, SCIR, which incorporates all of Reynolds's ideas and does not suffer from the difficulties with subject reduction discovered by Reynolds. The key innovation is to use *structured contexts*: the judgements of the system divide contexts into two zones, one for identifiers which are used only passively in the term being typed, the other for arbitrary identifiers. This setup is similar to that of Barber's DILL [1]. They also proposed a notion of categorical model for the system, *bireflective models*, and sketched the proof of an important correctness result for such models, *coherence*. A functor category model was given as an example of this structure. Yang and Huang have studied the question of type inference in the SCIR system [15], building on previous work for the original SCI system [5]; and the categorical properties of bireflective models have been studied by Freyd *et al.* [4, 3].

The notion of bireflective model is quite restrictive. In particular, bireflectivity forces certain computationally unnatural elements to exist in the model, and renders the property of *definability* (every element of the model being the denotation of some program phrase) unattainable (see [14] for a proof). Moreover, two new models of the SCIR type system have been discovered which do not quite fit the definition of bireflective model: Reddy's object-spaces model [9] and a model based on game semantics [14]. Swarup, Reddy and Ireland have developed a related type system, *Imperative Lambda-Calculus Revisited* (ILCR), and given a domain-theoretic model which is not bireflective [12, 13].

Additionally, the sketched proof of coherence for bireflective models of SCIR given in the original paper [8] appears to suffer from a couple of technical flaws, which we outline in Section 4 below. Reddy has indicated in a personal communication that he believes the problem of coherence for SCIR and ILCR is open, despite the proof sketches given in the literature.

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

This paper addresses both of these problems together. We present a generalization of the notion of bireflective model, obtained by weakening some of the conditions on such models: instead of asking for a bireflection we merely demand a reflection and a coreflection. (Some categorical properties of this kind of structure are studied in [2].) This setup is flexible enough to incorporate the new, non-bireflective models mentioned above, and subsumes the bireflective case. We go on to give a detailed proof of coherence for this new notion of model, subject to one additional constraint, which is enjoyed by all bireflective models as well as the games and object-spaces models.

2. *The SCIR type system*

The types of the language are given by the grammar

$$A ::= \gamma \mid A \to A \mid A \times A \mid PA$$

where γ ranges over a given set of *ground types*. Among these types, some are termed *passive types*. The passive types are those generated by the grammar

$$X ::= PA \mid X \times X \mid A \to X.$$

We will use A, B, C, … as metavariables for arbitrary types, and X, Y, and Z for passive types.

The intuition behind passive types is that they contain only phrases which have no effect on the store. These types form an exponential ideal: since the only way to use a function is to apply it, if the return type of a function is passive, the function itself must be passive.

The P constructor allows us to create a passive type from any other type. Intuitively, PA contains certain phrases of type A which happen not to affect the store. Our presentation of the type system generalizes the original in admitting the P operation on all types: the original SCIR contained only a 'passive function space' construct, corresponding to types of the form $P(A \to B)$ in our system. Our generalization is harmless and can be encoded in the original system with some syntactic sugar; including the P operation directly has the advantage of bringing the language slightly closer to its categorical model.

The terms of the language are given by the grammar

$$M ::= x \mid \lambda x : A.M \mid MM \mid \langle M, M\rangle \mid \pi_1 M \mid \pi_2 M \mid \mathsf{prom}(M) \mid \mathsf{der}(M) \mid \mathbf{k}$$

where x ranges over a countable set of variables and $\mathbf{k}$ over a specified set of constants. We assume each constant $\mathbf{k}$ is associated with a type $A_{\mathbf{k}}$.

The type system is given by an inductively defined set of judgements of the form

$$\Gamma \mid \Delta \vdash M : A.$$

Here Γ and Δ are contexts, that is to say, lists of identifier-type pairs written as $x : B$ for example, with each identifier appearing at most once in Γ, Δ; M is a term; and A is a type. A judgement such as this means that term M has type A, with free identifiers drawn from Γ and Δ, and that the identifiers in Γ are used only inside passive subterms.

The rules defining the type system are as follows.

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

Variable

$$\frac{}{- \mid x : A \vdash x : A}$$

Functions

$$\frac{\Gamma \mid \Delta, x : A \vdash M : B}{\Gamma \mid \Delta \vdash \lambda x : A.M : A \to B} \qquad \frac{\Gamma_1 \mid \Delta_1 \vdash M : A \to B \quad \Gamma_2 \mid \Delta_2 \vdash N : A}{\Gamma_1, \Gamma_2 \mid \Delta_1, \Delta_2 \vdash MN : B}$$

Products

$$\frac{\Gamma \mid \Delta \vdash M : A \quad \Gamma \mid \Delta \vdash N : B}{\Gamma \mid \Delta \vdash \langle M, N \rangle : A \times B}$$

$$\frac{\Gamma \mid \Delta \vdash M : A \times B}{\Gamma \mid \Delta \vdash \pi_1 M : A} \qquad \frac{\Gamma \mid \Delta \vdash M : A \times B}{\Gamma \mid \Delta \vdash \pi_2 M : B}$$

Permeability rules

$$\frac{\Gamma, x : A \mid \Delta \vdash M : B}{\Gamma \mid \Delta, x : A \vdash M : B}$$

$$\frac{\Gamma \mid \Delta, x : A \vdash M : X}{\Gamma, x : A \mid \Delta \vdash M : X}\ X \text{ passive}$$

Promotion and dereliction

$$\frac{\Gamma \mid - \vdash M : A}{\Gamma \mid - \vdash \mathsf{prom}(M) : PA}$$

$$\frac{\Gamma \mid \Delta \vdash M : PA}{\Gamma \mid \Delta \vdash \mathsf{der}(M) : A}$$

Weakening

$$\frac{\Gamma \mid \Delta \vdash M : A}{\Gamma' \mid \Delta' \vdash M : A}\ \Gamma \subseteq \Gamma', \Delta \subseteq \Delta'$$

Contraction

$$\frac{\Gamma, x : A, y : A \mid \Delta \vdash M : B}{\Gamma, x : A \mid \Delta \vdash M[x/y] : B}$$

Exchange

$$\frac{\Gamma \mid \Delta \vdash M : A}{\Gamma' \mid \Delta' \vdash M : A}\ \Gamma' \text{ a permutation of } \Gamma, \Delta' \text{ a permutation of } \Delta$$

Constants

$$\frac{}{- \mid - \vdash \mathbf{k} : A_{\mathbf{k}}}$$

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

The way contexts are combined means that in a function application MN, M and N do not have identifers in common, whereas in a pair $\langle M, N\rangle$ they may. Contraction is allowed in the passive zone only, so passively used identifiers may be shared. A term containing no actively used identifiers can be *promoted* to a passive type; such a term can then be derelicted back to its original type for later use.

The permeability rules are central to the system. The first, which we call *activation*, allows the type system to forget the information that an identifier is passively used. The second states that inside a term of passive type, all identifiers may be deemed passively used.

2.1. *An illustrative language*

O'Hearn *et al.* describe an SCIR-based language, that is to say, a collection of base types and constants, which illustrates the ideas behind the system. The language is essentially an extension of Reynolds's Idealized Algol.

The base types are `exp`, the type of arithmetic expressions; `com`, the type of commands which affect the store and return no result; and `var`, the type of assignable variables which store numerical values.

Constants include:

- numerals and arithmetic operations: e.g. $\texttt{3} : P\texttt{exp}$ and $+ : P\texttt{exp} \times P\texttt{exp} \to P\texttt{exp}$. Note the use of the P constructor to ensure that arithmetic expressions do not affect the store.
- basic imperative constructs: e.g. $\texttt{assign} : \texttt{var} \times P\texttt{exp} \to \texttt{com}$ for assignment; $\texttt{assign}\langle \texttt{x}, 4\rangle$ would more commonly be written as $\texttt{x} := 4$. To look up a value stored in a variable we have a constant $\texttt{deref} : \texttt{var} \to P\texttt{exp}$. Further constants provide for sequential composition of commands ($\texttt{seq} : \texttt{com} \times \texttt{com} \to \texttt{com}$), `while`-loops etc.
- local variable allocation: $\texttt{new} : (\texttt{var} \to \texttt{com}) \to \texttt{com}$. The idea here is that $\texttt{new}(\lambda x.C)$ allocates a fresh variable, binds x to that variable, executes the command C, and finally deallocates the variable.
- block expressions: $\texttt{do} : P(\texttt{var} \to \texttt{com}) \to P\texttt{exp}$. The intuition is that $\texttt{do}(\texttt{prom}(\lambda x.C))$ allocates a fresh variable, binds x to it, executes C, and then returns the final value of x as a result. The P constructor on the type of the argument ensures that the command C has no effect on the store, except perhaps on x, so that the resulting expression is side-effect-free and can be given the passive type $P\texttt{exp}$. For instance, one could write the factorial function as follows.

$$\lambda a : P\texttt{exp}.\texttt{do}(\texttt{prom}(\lambda x.\texttt{new}(\lambda y.\ \begin{array}{l} y := a; \\ x := 1; \\ \texttt{while}(!y > 1) \\ \qquad x :=!x \times !y; \\ \qquad y :=!y - 1;)) \end{array}$$

Despite making crucial use of the imperative constants in the illustrative language, this term has the purely passive type $P\texttt{exp} \to P\texttt{exp}$. The games model of this illustrative language [14] demonstrates that the passive sublanguage is purely functional: it has the same denotational semantics as the Hyland–Ong fully abstract model of PCF [6]. Thus the block expression allows us to write purely functional programs in an imperative fashion. The type system guarantees that the use of state made by passively-typed programs is completely encapsulated, in a manner quite different from, for instance, Haskell's monad-based approach [7].

3. *Modelling SCIR categorically*

The idea behind our categorical model is the same as that in [8]: the passive types form a full subcategory of the category used to model the language, and the inclusion functor has both left and right adjoints which allow us to interpret the novel typing rules of promotion, dereliction and permeability.

We briefly illustrate the key ideas in this interpretation. Let $\mathbf{C}$ be our base category, and $\mathbf{P}$ the full subcategory of 'passive type objects', with inclusion functor

$$J : \mathbf{P} \hookrightarrow \mathbf{C}$$

having left adjoint S and right adjoint P:

$$S \dashv J \dashv P.$$

A judgement $x : A \mid y : B \vdash M : C$ will be interpreted as a map

$$JSA \otimes B \to C$$

in $\mathbf{C}$, and the type constructor P will be interpreted using the functor JP.

Thus a judgement like $x : A \mid - \vdash M : B$ with no actively-used identifiers will be interpreted as a map $JSA \to B$. Since $J \dashv P$, such maps are in 1-1 correspondence with maps $SA \to PB$ and since J is full and faithful, these are in 1-1 correspondence with maps $JSA \to JPB$. This allows us to interpret the promotion rule, while the counit $\epsilon' : JP \to \mathsf{Id}$ of the adjunction $J \dashv P$ gives rise to an interpretation of dereliction.

For the permeability rules, note that a judgement

$$- \mid x : A \vdash M : X$$

where X is passive will be interpreted as a map $A \to JX'$ for some object X' in the passive subcategory. Passing across the adjunction $S \dashv J$ and using the fullness of the embedding J, these maps are in 1-1 correspondence with maps $JSA \to JX'$ which interpret judegements of the form

$$x : A \mid - \vdash M : X$$

thus allowing us to interpret passification. The activation rule is handled by composition with the unit $\eta : \mathsf{Id} \to JS$ of the adjunction $S \dashv J$.

In [8], these ideas are taken further: it is insisted that S and P be the same functor, giving rise to the notion of *bireflective model*. However, this notion is overly restrictive, and indeed forces models to contain certain unnecessary elements. For instance, the block-expression constant `do` from the illustrative language should correspond to a map

$$JP(\mathtt{var} \to \mathtt{com}) \to JP\mathtt{exp}.$$

Since J is full, such maps are in correspondence with maps

$$P(\mathtt{var} \to \mathtt{com}) \to P\mathtt{exp}$$

in $\mathbf{P}$.

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

If $P = S$, we can now pass across the adjunction $S \dashv J$ to see that these maps are in 1-1 correspondence with maps

$$(\mathtt{var} \to \mathtt{com}) \to JP\mathtt{exp}.$$

That is to say, the P constructor on the argument type of `do` can be removed. But our intuition is that it is *essential* to use the P constructor there, in order to prevent the creation of terms of type $P\mathtt{exp}$ which have impact on the store. Thus the notion of bireflectivity introduces some counterintuitive degeneracy into the models.

Indeed, the work of Wall [14] shows that no bireflective model of the illustrative language can possess the property of *definability*, which asks that every (finite) element of the model be the denotation of some term. This is a serious obstacle if one wants to build a fully abstract model, for example, since full abstraction typically relies on some sort of definability result.

To counter these difficulties, we will not insist that the left and right adjoints are the same functor. This gives rise to a more flexible class of models, without the problems mentioned above. Moreover, two more recently discovered models of SCIR, which are not bireflective models, do fit into our generalized setting: the game-theoretical model of [14] and Reddy's event-based model [9]; as does the imperative-domains model of ILCR [12].

3.1. *Definition of a categorical model*

We ask for a symmetric monoidal category $(\mathbf{C}, \otimes, I, \alpha, \rho, \mathsf{symm})$, with finite products written as $\times$ for the binary product and 1 for the terminal. To interpret the passive types, we ask for a full subcategory $\mathbf{P}$ of $\mathbf{C}$ with inclusion functor

$$J : \mathbf{P} \hookrightarrow \mathbf{C}$$

having left adjoint S and right adjoint P:

$$S \dashv J \dashv P.$$

We write the unit and counit of $S \dashv J$ as η and ϵ, and use η' and ϵ' for the unit and counit of $J \dashv P$.

The following are standard.

Lemma 1. *Each component $\epsilon_X : SJX \to X$ of the counit of the adjunction $S \dashv J$ is an isomorphism. As a corollary, each component $\eta_{JX} : JX \to JSJX$ of the unit of this adjunction is also an isomorphism, and $\eta_{JSA} = JS\eta_A : JSA \to JSJSA$ for all objects A of $\mathbf{C}$.*

Dually, each component $\eta'_X : X \to PJX$ of the unit of the adjunction $J \dashv P$ is an isomorphism, as is each $\epsilon'_{JX} : JPJX \to JX$, and $\epsilon'_{JPA} = JP\epsilon'_A$.

Proof. Since J is full and faithful, it induces a natural isomorphism $\mathbf{P}(X, -) \cong \mathbf{C}(JX, J-)$ for any passive object X. The adjuction $J \dashv S$ induces a natural isomorphism $\mathbf{C}(JX, J-) \cong \mathbf{P}(SJX, -)$. Chasing id_X across this chain of natural isomorphisms gives ϵ_X, so by the Yoneda lemma the composite natural isomorphism $\mathbf{P}(X, -) \cong \mathbf{P}(SJX, -)$ is $\mathbf{P}(\epsilon_X, -)$, and therefore ϵ_X is itself an isomorphism. Since $\eta_{JX}; J\epsilon_X = \mathsf{id}_{JX}$ by adjuction laws, η_{JX} is an isomorphism too.

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

Finally we have that

$$\begin{aligned} JS\eta_A; J\epsilon_{SA} &= J(S\eta_A; \epsilon_{SA}) \\ &= J(\mathsf{id}_{SA}) && \text{by adjunction laws} \\ &= \mathsf{id}_{JSA} \end{aligned}$$

and also $\eta_{JSA}; J\epsilon_{SA} = \mathsf{id}_{JSA}$. Since $J\epsilon_{SA}$ is an isomorphism, it follows that $JS\eta_A = \eta_{JSA}$.

The proofs for the other adjunction are dual. □

An immediate consequence of our definitions is that $\mathbf{P}$ has finite products. The product of passive objects X and Y is given by $P(JX \times JY)$:

$$\begin{aligned} \mathbf{P}(Z, P(JX \times JY)) &\cong \mathbf{C}(JZ, JX \times JY) \\ &\cong \mathbf{C}(JZ, JX) \times \mathbf{C}(JZ, JY) \\ &\cong \mathbf{P}(Z, X) \times \mathbf{P}(Z, Y). \end{aligned}$$

We will again use $\times$ for the product structure in $\mathbf{P}$. It is also immediate that J preserves products, since it is a right-adjoint.

A judgement $x : A \mid y : B \vdash M : C$ will be interpreted as a map

$$[\![x : A \mid y : B \vdash M : C]\!] : JSA \otimes B \to C$$

and the passive-type constructor P will be modelled using the functor JP.

Since the functors J and S are used on contexts, we require them to be *strong monoidal*, that is, we ask for natural isomorphisms

$$s_{A,B} : SA \times SB \to S(A \otimes B) \qquad s' : 1 \to SI$$

and

$$j_{X,Y} : JX \otimes JY \to J(X \times Y) \qquad j' : I \to J1$$

subject to the following coherence constraints for s and s', and similar ones for j and j'.

$$\begin{tikzcd} SA \times (SB \times SC) \arrow[r, "\alpha"] \arrow[d, "\mathsf{id} \times s"'] & (SA \times SB) \times SC \arrow[d, "s \times \mathsf{id}"] \\ SA \times S(B \otimes C) \arrow[d, "s"'] & S(A \otimes B) \times SC \arrow[d, "s"] \\ S(A \otimes (B \otimes C)) \arrow[r, "S\alpha"'] & S((A \otimes B) \otimes C) \end{tikzcd}$$

$$\begin{tikzcd}
SA \times 1 \arrow[r, "\mathsf{id} \times s'"] \arrow[rrd, "\rho"'] & SA \times SI \arrow[r, "s"] & S(A \otimes I) \arrow[d, "S\rho"] \\
 & & SA
\end{tikzcd}
\qquad
\begin{tikzcd}
SA \times SB \arrow[r, "s"] \arrow[d, "\mathsf{symm}"'] & S(A \otimes B) \arrow[d, "S\mathsf{symm}"] \\
SB \times SA \arrow[r, "s"'] & S(B \otimes A)
\end{tikzcd}$$

Since J is a right adjoint, $J1$ is a terminal object, so the isomorphism j' means that I is terminal, and the isomorphism s' means that SI is terminal too.

The fact that I is terminal implies that there are 'projection' maps out of the tensor: the left projection is given by

$$\begin{tikzcd}
A \otimes B \arrow[r, "\mathsf{id} \otimes !"] & A \otimes I \arrow[r, "\rho"] & A
\end{tikzcd}$$

and right projection similarly. We will write these projections as π_1 and π_2, and use the same notation for projection from the product $\times$.

Lemma 2. *The natural isomorphisms s and j commute with projections; that is*

$$\begin{tikzcd}
SA \times SB \arrow[r, "s"] \arrow[d, "\pi_1"'] & S(A \otimes B) \arrow[ld, "S\pi_1"] \\
SA.
\end{tikzcd}$$

and similar diagrams for π_2 and for j.

Proof. Consider the diagram below.

$$\begin{tikzcd}
SA \times SB \arrow[r, "s"] \arrow[d, "S\mathsf{id} \times S!"'] & S(A \otimes B) \arrow[d, "S(\mathsf{id} \otimes !)"] \\
SA \times SI \arrow[r, "s"] \arrow[d, "\mathsf{id} \times !"'] & S(A \otimes I) \arrow[d, "S\rho"] \\
SA \times 1 \arrow[r, "\rho"'] & SA
\end{tikzcd}$$

The upper rectangle commutes by naturality of s, and the lower is an instance of the unit coherence diagram for s, using the fact that the unique map $! : S1 \to 1$ is the inverse of s'. The top-right route around the rectangle is $s; S\pi_1$, and the left-bottom route is π_1. □

To model SCIR, we will also require that the unit $\eta : \mathsf{Id} \to JS$ and counit $\epsilon : SJ \to \mathsf{Id}$ of the adjunction $S \dashv J$ be monoidal natural transformations; that is, that the following diagrams commute.

$$\begin{tikzcd}
SJX \times SJY \arrow[r, "\epsilon_X \times \epsilon_Y"] \arrow[d, "s"'] & X \times Y & 1 \arrow[r, "\mathsf{id}"] \arrow[d, "s'"'] & 1 \\
S(JX \otimes JY) \arrow[d, "Sj"'] & & SI \arrow[d, "Sj'"'] & \\
SJ(X \times Y) \arrow[uur, "\epsilon_{X \times Y}"'] & & SJ1 \arrow[uur, "\epsilon_1"'] &
\end{tikzcd}$$

$$\begin{tikzcd}
A \otimes B \arrow[r, "\eta_A \otimes \eta_B"] \arrow[ddr, "\eta_{A \otimes B}"'] & JSA \otimes JSB \arrow[d, "j"] & I \arrow[r, "\eta_I"] \arrow[d, "j'"'] & JSI \\
& J(SA \times SB) \arrow[d, "Js"] & J1 \arrow[d, "Js'"'] & \\
& JS(A \otimes B) & JSI \arrow[uur, "\mathsf{id}"'] &
\end{tikzcd}$$

In fact these equations hold automatically.

Lemma 3. *The unit $\eta : \mathsf{Id} \to JS$ and counit $\epsilon : SJ \to \mathsf{Id}$ of the adjunction $S \dashv J$ are monoidal natural transformations.*

Proof. The diagrams involving the units are trivial since 1 and JSI are terminal. Of the other two, we will consider only the case of the unit η. The case for ϵ is similar.

First note that since J preserves products, $J(SA \times SB)$ is a product, with projections given by $J\pi_1$ and $J\pi_2$. Thus $JSA \otimes JSB$ is also a product, and by Lemma 2 its projections are π_1 and π_2. Similarly $JS(A \otimes B)$ is a product, with projections $JS\pi_1$ and $JS\pi_2$.

Therefore to check that the diagram commutes, we need only check that

$$\eta_{A \otimes B}; JS\pi_1 = \eta_A \otimes \eta_B; j; Js; JS\pi_1$$

and similarly for π_2.

By Lemma 2 again, the right hand side above is equal to $\eta_A \otimes \eta_B; \pi_1$, which is $\pi_1; \eta_A$ by naturality of projections, which is equal to the left hand side by naturality of η. □

3.1.1. Example of the structure

The following construction gives a large class of examples of the structure we desire, which illustrate the intuition behind the categorical model, albeit in a somewhat degenerate way. Given any cartesian closed category **B**, construct a category **C** as follows. Objects of **C** are pairs (A, X) of **B**-objects, and an arrow from (A, X) to

(B, Y) is given by a commuting square

$$\begin{tikzcd}
A \times X \arrow[r, "f"] \arrow[d, "\pi_2"'] & B \times Y \arrow[d, "\pi_2"] \\
X \arrow[r, "f'"'] & Y
\end{tikzcd}$$

in $\mathbf{B}$. Composition is by pasting squares together. The category $\mathbf{B}$ itself plays the role of $\mathbf{P}$, with the inclusion functor J taking an object X to $(1, X)$. The intuition behind this construction is that the object (A, X) consists of some active data A and passive data X; the commuting square expresses the idea that maps may only use passive input data in computing passive output data.

The category $\mathbf{C}$ has products, given componentwise, and exponentials, given by

$$(A, X) \Rightarrow (B, Y) = (A \times X \Rightarrow B, X \Rightarrow Y)$$

and is hence cartesian closed. This construction therefore does not provide an accurate account of SCIR: there is no restriction on contraction in this model, since the monoidal structure is in fact cartesian. It is nevertheless an informative construction, because of the nature of the left and right adjoints to J.

The left adjoint S takes an object (A, X) to X and a map as in the diagram above to the lower component f'. The right adjoint P takes (A, X) to $A \times X$ and a map as in the above diagram to the upper component f. It is straightforward to verify that these are indeed left and right adjoints to J.

This example provides useful intuition about the roles of S and P: the functor S restricts an object to its passive part, while P declares the entire object passive. This is analogous to the way in which these functors work in the games model and the object-spaces model. In models such as these, S and P have quite different roles, whereas in a bireflective model the two are identified. This is one reason why we believe it is important to introduce this more general notion of model: bireflectivity enforces a view of the SCIR system which does not correspond to computational intuition.

3.2. *Passification and promotion*

LEMMA 4. *Let $f : A \to B$ in $\mathbf{C}$. If B is isomorphic to JX for some X in $\mathbf{P}$, there is a unique* $\mathsf{pass}(f) : JSA \to B$ *such that*

$$\begin{tikzcd}
A \arrow[r, "\eta_A"] \arrow[dr, "f"'] & JSA \arrow[d, "\mathsf{pass}(f)"] \\
& B.
\end{tikzcd}$$

Proof. Let $i : B \to JX$ be any isomorphism, and define

$$\mathsf{pass}(f) = JS(f; i); J\epsilon; i^{-1}.$$

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

To see that $\eta; \mathsf{pass}(f) = f$, consider the following diagram.

$$\begin{tikzcd}
A \arrow[r, "\eta"] \arrow[d, "f"'] & JSA \arrow[d, "JSf"] \\
B \arrow[r, "\eta"] \arrow[d, "i"'] & JSB \arrow[d, "JSi"] \\
JX \arrow[r, "\eta"] \arrow[d, "i^{-1}"'] \arrow[dr, "\mathsf{id}"] & JSJX \arrow[d, "J\epsilon"] \\
B & JX \arrow[l, "i^{-1}"]
\end{tikzcd}$$

The upper two squares commute by naturality of η. One of the triangles is trivial, and the other is one of the triangular equations for an adjunction.

To see that $\mathsf{pass}(f)$ is unique, let g be such that $\eta; g = f$. Then

$$\begin{aligned}
\mathsf{pass}(f) &= \mathsf{pass}(\eta; g) \\
&= JS\eta_A; JSg; JSi; J\epsilon_X; i^{-1} \\
&= \eta_{JSA}; JSg; JSi; J\epsilon_X; i^{-1} && \text{by Lemma 1} \\
&= g; i; \eta_{JX}; J\epsilon_X; i^{-1} && \text{by naturality of } \eta \\
&= g; i; \mathsf{id}; i^{-1} && \text{by adjunction laws} \\
&= g.
\end{aligned}$$

The proof is complete. □

Because the adjunction $S \dashv J$ is monoidal, this result can be parameterized as follows.

LEMMA 5. *Let $f : A \otimes B \to C$ be any map in $\mathbf{C}$, and suppose that C is isomorphic to JX for some object X. Then there exists a unique $\mathsf{pass}_B(f) : A \otimes JSB \to C$ such that $\mathsf{id} \otimes \eta_B; \mathsf{pass}_B(f) = f$.*

Proof. Define $\mathsf{pass}_B(f)$ as

$$A \otimes JSB \xrightarrow{\eta_A \otimes \mathsf{id}} JSA \otimes JSB \xrightarrow{j} J(SA \times SB) \xrightarrow{Js} JS(A \otimes B) \xrightarrow{\mathsf{pass}(f)} C.$$

Then $\mathsf{id} \otimes \eta_B; \mathsf{pass}_B(f) = \eta_A \otimes \eta_B; j; Js; \mathsf{pass}(f) = \eta_{A \otimes B}; \mathsf{pass}(f) = f$ using the monoidalness of η and the defining property of $\mathsf{pass}(f)$. Uniqueness follows similarly to that of $\mathsf{pass}(f)$. □

In what follows we will often abuse notation and write simply $\mathsf{pass}(f)$ for $\mathsf{pass}_B(f)$.

The other adjunction gives us the following dual result.

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

LEMMA 6. *Let $f : A \to B$ in $\mathbf{C}$. If A is isomorphic to JX for some X in $\mathbf{P}$, there is a unique $\mathsf{prom}(f) : A \to JPB$ such that*

$$\begin{tikzcd} A \arrow[r, "\mathsf{prom}(f)"] \arrow[dr, "f"'] & JPB \arrow[d, "\epsilon'_B"] \\ & B. \end{tikzcd}$$

3.3. *Interpreting types*

We have products in our category, to interpret the × operation on types, and we have a functor JP which will allow us to interpret the P operation. To interpret the function types, we require our category $\mathbf{C}$ to have certain exponentials. Rather than asking for all exponentials to exist, we ask only for the ones we need: this is to accommodate the object-space and game-based models, which do not have all exponentials.

We ask instead that there exists a collection of objects, called *type objects*, containing at least one object to interpret each ground type, closed under products and the functor JP, and such that for any object A and type object B, the exponential $A \to B$ (w.r.t. the tensor structure) exists, and is itself a type object.

Interpretation of types is now straightforward: γ is interpreted by $[\![\gamma]\!]$, $[\![A \times B]\!] = [\![A]\!] \times [\![B]\!]$, $[\![PA]\!] = JP[\![A]\!]$ and $[\![A \to B]\!] = [\![A]\!] \to [\![B]\!]$. We will often drop the semantic brackets and use A to mean both the type A and the type-object $[\![A]\!]$.

Given a map $f : A \otimes B \to C$, C is a type object, we will write $\Lambda(f) : A \to (B \to C)$ for the curried map given by the exponential. The evaluation map we write as $\mathsf{ev}_{\mathsf{B},\mathsf{C}} : (B \to C) \otimes B \to C$.

3.4. *Interpreting derivations*

For each derivation of a typed term $\Gamma \mid \Delta \vdash M : C$ we will define, by induction on derivations, a map

$$JSA_1 \otimes \cdots \otimes JSA_n \otimes B_1 \otimes \cdots \otimes B_m \to C.$$

To ease notation, we often abbreviate this as $JS\Gamma \otimes \Delta \to C$.

First, some observations are in order. We say that a type-object is *passive* if it is isomorphic to JX for some X. Note that, since $JX \cong JSJX$, it is the case that any passive object A is isomorphic to JSA.

LEMMA 7. *Let A and B be type objects. If both A and B are passive, so is $A \times B$. If B is passive, then $A \to B$ is passive (even if A is not).*

Proof. By definition of passive object, $A \cong JX$ and $B \cong JY$ for some objects X and Y of $\mathbf{P}$. Then we have

$$A \times B \cong JX \times JY \cong J(X \times Y)$$

since J preserves products.

For $A \to B$ where $B \cong JY$, consider the following sequence of natural isomorphisms.

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

$$\begin{aligned}
\mathbf{C}(-, A \to B) &\cong \mathbf{C}(- \otimes A, B) \\
&\cong \mathbf{C}(- \otimes A, JY) \\
&\cong \mathbf{P}(S(- \otimes A), Y) \\
&\cong \mathbf{C}(JS(- \otimes A), JY) && \text{since } J \text{ is full} \\
&\cong \mathbf{C}(JS - \otimes JSA, B) \\
&\cong \mathbf{C}(JS-, JSA \to B) \\
&\cong \mathbf{P}(S-, P(JSA \to B)) \\
&\cong \mathbf{C}(-, JP(JSA \to B))
\end{aligned}$$

By the Yoneda lemma, $A \to B \cong JP(JSA \to B)$. □

Thus the type object interpreting any passive type is itself passive. This is vital for our interpretation of the typing rules involving passive types.

The inductive definition of the semantics of derivations is as follows.

Variable. The derivation

$$\frac{}{- \mid x : A \vdash x : A}$$

is interpreted by the identity map $\mathsf{id} : A \to A$.

Abstraction.

$$\frac{\Gamma \mid \Delta, x : A \vdash M : B}{\Gamma \mid \Delta \vdash \lambda x : A.M : A \to B}$$

Given an interpretation $m : JS\Gamma \otimes \Delta \otimes A \to B$ of the premise, the conclusion is interpreted by

$$\Lambda(m) : JS\Gamma \otimes \Delta \to (A \to B).$$

Application.

$$\frac{\Gamma_1 \mid \Delta_1 \vdash M : A \to B \quad \Gamma_2 \mid \Delta_2 \vdash N : A}{\Gamma_1, \Gamma_2 \mid \Delta_1, \Delta_2 \vdash MN : B}$$

If m interprets the first premise and n the second, the conclusion is interpreted by

$$JS\Gamma_1 \otimes JS\Gamma_2 \otimes \Delta_1 \otimes \Delta_2 \xrightarrow{\cong} JS\Gamma_1 \otimes \Delta_1 \otimes JS\Gamma_2\Delta_2 \xrightarrow{m \otimes n} (A \to B) \otimes A \xrightarrow{\mathsf{ev}_{A,B}} B.$$

Pairing.

$$\frac{\Gamma \mid \Delta \vdash M : A \quad \Gamma \mid \Delta \vdash N : B}{\Gamma \mid \Delta \vdash \langle M, N \rangle : A \times B}$$

If m interprets the first premise and n the second, the conclusion is interpreted by $\langle m, n \rangle$.

Projection.

$$\frac{\Gamma \mid \Delta \vdash M : A \times B}{\Gamma \mid \Delta \vdash \pi_1 M : A}$$

If $m : JS\Gamma \otimes \Delta \to A \times B$ interprets the premise, the conclusion is interpreted by $m; \pi_1$. Right projection is interpreted similarly.

Activation.

$$\frac{\Gamma, x : A \mid \Delta \vdash M : B}{\Gamma \mid \Delta, x : A \vdash M : B}$$

If the premise is interpreted by $m : JS\Gamma \otimes JSA \otimes \Delta \to B$, the conclusion is interpreted by

$$\begin{tikzcd} JS\Gamma \otimes \Delta \otimes A \arrow[r, "\cong"] & JS\Gamma \otimes A \otimes \Delta \arrow[r, "\mathsf{id} \otimes \eta_A \otimes \mathsf{id}"] & JS\Gamma \otimes JSA \otimes \Delta \arrow[r, "m"] & B. \end{tikzcd}$$

Passification.

$$\frac{\Gamma \mid \Delta, x : A \vdash M : X}{\Gamma, x : A \mid \Delta \vdash M : X}\ X \text{ passive}$$

If the premise is interpreted by $m : JS\Gamma \otimes \Delta \otimes A \to X$, the conclusion is interpreted by

$$\begin{tikzcd} JS\Gamma \otimes JSA \otimes \Delta \arrow[r, "\cong"] & JS\Gamma \otimes \Delta \otimes JSA \arrow[r, "\mathsf{pass}_A(m)"] & X. \end{tikzcd}$$

Note that we are relying on the fact that $[\![X]\!]$ is a passive object, so that $\mathsf{pass}_A(m)$ exists.

Promotion.

$$\frac{\Gamma \mid - \vdash M : A}{\Gamma \mid - \vdash \mathsf{prom}(M) : PA}$$

If the premise is interpreted by $m : JS\Gamma \to A$, the conclusion is interpreted by $\mathsf{prom}(m) : JS\Gamma \to JPA$.

Dereliction.

$$\frac{\Gamma \mid \Delta \vdash M : PA}{\Gamma \mid \Delta \vdash \mathsf{der}(M) : A}$$

If m interprets the premise, the conclusion is interpreted by $m; \epsilon'_A$.

Weakening.

$$\frac{\Gamma \mid \Delta \vdash M : B}{\Gamma' \mid \Delta' \vdash M : B}$$

where Γ' and Δ' are extensions of Γ and Δ.

If m interprets the premise, the conclusion is interpreted by

$$\begin{tikzcd} JS\Gamma' \otimes \Delta' \arrow[r, "\pi_{\Gamma',\Gamma} \otimes \pi_{\Delta',\Delta}"] & JS\Gamma \otimes \Delta \arrow[r, "m"] & B \end{tikzcd}$$

where the π maps are appropriate projections.

Contraction.

$$\frac{\Gamma, x : A, y : A \mid \Delta \vdash M : B}{\Gamma, x : A \mid \Delta \vdash M[x/y] : B}$$

If m interprets the premise, the conclusion is interpreted by

$$JS\Gamma \otimes JSA \otimes \Delta \xrightarrow{\mathsf{id} \otimes \mathsf{diag} \otimes \mathsf{id}} JS\Gamma \otimes JSA \otimes JSA \otimes \Delta \xrightarrow{m} B$$

where $\mathsf{diag} : JSA \to JSA \otimes JSA$ is a diagonal map, which exists since $JSA \otimes JSA \cong JSA \times JSA$.

Constants. For each constant $\mathbf{k}$ of type A we fix a map $[\![\mathbf{k}]\!] : 1 \to [\![A]\!]$ as its interpretation.

4. *The problem of coherence*

For any derivation of a judgement $\Gamma \mid \Delta \vdash M : A$, the definitions of the previous section give an interpretation as a map $JS\Gamma \otimes \Delta \to A$ in $\mathbf{C}$. However, because of the structural rules (weakening, contraction and exchange) and the permeability rules (passification and activation), a given judgement may have more than one derivation. Thus a natural question arises: is every derivation of a given judgement interpreted as the same map? We term this the *coherence* problem for models of SCIR, since it has something of the flavour of categorical coherence problems.

In the remainder of the paper we show that, subject to one additional condition, our notion of model does indeed enjoy coherence.

For bireflective models, a coherence result was stated and a proof sketched in [8]. Our approach to the proof is slightly different; in fact much of the work in this paper was motivated by the fact that the present author was unable to recover the original proof from its sketch. We shall now outline the sources of difficulty in the original proof, with the aim of motivating the detailed constructions which follow.

The original proof comes in two parts. First it is shown that any two ways of extending a derivation of $\Gamma \mid \Delta \vdash M : A$ to one of $\Gamma' \mid \Delta' \vdash M' : A$ using structural rules and permeability rules give rise to the same semantics; this is termed 'coherence of structural extensions' and is quite straightforward to establish. Next consider two distinct derivations of a judgement $\Gamma \mid \Delta \vdash M : A$. These must consist of a derivation of a judgement $\Gamma_i \mid \Delta_i \vdash M_i : A$, ending with a term-forming rule, followed by a structural extension turning this judgement into $\Gamma \mid \Delta \vdash M : A$. The proof then proceeds by induction on the size of the derivations up to the last term-forming rule, and by cases according to what the last term-forming rule was.

The most interesting case is that of application. We have derivations

$$\frac{\begin{array}{c}\vdots\,\phi_i\\ \Gamma_i \mid \Delta_i \vdash M_i : A \to B\end{array} \qquad \begin{array}{c}\vdots\,\phi'_i\\ \Gamma'_i \mid \Delta'_i \vdash N_i : A\end{array}}{\Gamma_i\Gamma'_i \mid \Delta_i\Delta'_i \vdash M_iN_i : B}$$

(for $i = 1, 2$) and sequences of structural rules which turn each

$$\Gamma_i\Gamma'_i \mid \Delta_i\Delta'_i \vdash M_iN_i : B$$

into $\Gamma \mid \Delta \vdash MN : B$. The idea given in the original proof-sketch is to push some of these structural rules up above the application rule, so that M_1 and M_2 are unified, and N_1 and N_2 are unified, and then apply the inductive hypothesis.

However, this cannot be done in general, for two reasons.

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

1. When the type B is passive but A is active, the original structural extensions may involve passifying and contracting variables inside the N_i, which cannot be performed before the application rule.
2. There may be extraneous weakenings in the derivations which make it impossible to unify the judgements before the application rule is used.

To illustrate the first problem, consider the following derivations. We begin with the judgement

$$- \mid f : \mathtt{com} \rightarrow \mathtt{com} \rightarrow P\mathtt{exp}, x : \mathtt{com}, y : \mathtt{com} \vdash fxy : P\mathtt{exp}$$

which is clearly derivable. From it we can derive $z := fxx$ as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{- \mid f,x,y \vdash fxy : P\mathtt{exp}}{x,y \mid f \vdash fxy : P\mathtt{exp}}\,\text{passify}}{x \mid f \vdash fxx : P\mathtt{exp}}\,\text{contract}}{x \mid f,z \vdash fxx : P\mathtt{exp}}\,\text{weaken} \qquad \dfrac{\dfrac{}{- \mid z : \mathtt{var} \vdash z : \mathtt{var}}}{x \mid f,z \vdash z : \mathtt{var}}\,\text{weaken}}{x \mid f,z \vdash z := fxx : \mathtt{com}}\,\text{assignment}$$

(Assignment is a combination of syntactic sugar and an admissible rule involving two uses of the application rule.) We can also derive $z := fxy$ very straightforwardly:

$$\dfrac{\dfrac{- \mid f,x,y \vdash fxy : P\mathtt{exp}}{- \mid f,x,y,z : \mathtt{var} \vdash fxy : P\mathtt{exp}}\,\text{weaken} \qquad \dfrac{\dfrac{}{- \mid z : \mathtt{var} \vdash z : \mathtt{var}}}{- \mid f,x,y,z \vdash z : \mathtt{var}}\,\text{weaken}}{- \mid f,x,y,z \vdash z := fxy : \mathtt{com}}\,\text{assignment.}$$

Now consider applying some $g : \mathtt{com} \rightarrow P\mathtt{exp}$ to each of these, yielding derivations of

$$x \mid g,f,z \vdash g(z := fxx) : P\mathtt{exp}$$

and

$$- \mid g,f,x,y,z \vdash g(z := fxy) : P\mathtt{exp}.$$

It is possible to unify these two judgements by means of structural rules: one simply passifies x and y in the latter judgement and contracts them to x. However, it is *not* possible to unify the two argument-judgements by structural extensions: one cannot passify x and y in $- \mid f,x,y,z \vdash z := fxy : \mathtt{com}$ because the result type is not passive.

Thus the induction outlined in the original paper cannot be pushed through, at least not without further analysis. The coherence proof presented in this paper introduces the notion of *pseudo-passification* to allow for the early passification of variables we know will eventually appear in passively typed subterms, so that the induction can be completed.

The second problem is rather easier to overcome, but nevertheless was not mentioned in the original proof-sketch, so we illustrate it here.

Using the variable and weakening rules, one can clearly derive judgements such as

$$x \mid a \vdash a \qquad y \mid b \vdash b$$

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

and hence (assuming appropriate typing, with all types active)

$$x, y \mid a, b \vdash ab. \tag{1}$$

Similarly, we can clearly derive

$$- \mid a \vdash a \qquad - \mid x, b \vdash b$$

and from these

$$- \mid x, a, b \vdash ab. \tag{2}$$

Judgements 1 and 2 can be unified by means of structural rules, operating only on the first derivation:

$$\dfrac{\dfrac{x, y \mid a, b \vdash ab}{x \mid a, b \vdash ab}}{- \mid x, a, b \vdash ab\,.}$$

The original proof then suggests that we should be able to unify the judgements

$$x \mid a \vdash a \quad \text{with} \quad - \mid a \vdash a$$

and

$$y \mid b \vdash b \quad \text{with} \quad \mid x, b \vdash b$$

using structural rules, in such a way that performing application on the resulting judgements gives us a

$$\Gamma \mid \Delta \vdash MN$$

which can itself be turned into

$$- \mid x, a, b \vdash ab$$

using structural rules. This cannot be done. Any unification of the first pair of judgements must look like

$$\Gamma \mid \Delta, a \vdash a$$

with at least one variable (resulting from contracting x with some other variable) in Γ, Δ. Similarly any unification of the second pair of judgements must look like

$$\Gamma' \mid \Delta', x, b \vdash b$$

with at least one variable appearing in Γ', Δ'. After application, we obtain

$$\Gamma\Gamma' \mid \Delta\Delta', x, a, b \vdash ab.$$

But the $\Gamma, \Gamma', \Delta, \Delta'$ are non-empty, and it is not possible to perform contractions between these variables and x, a, b, so it is not possible to transform this judgement into $- \mid x, a, b \vdash ab$ by means of structural rules.

This problem is handled by first eliminating extraneous weakenings from the derivations we consider: a *strengthening* lemma allows us to do that.

In the rest of this section, we perform a detailed analysis of the structural and permeability rules, which will aid us in the coherence proof to follow.

4.1. *Structural maps*

Let $\Gamma \mid \Delta$ and $\Gamma' \mid \Delta'$ be contexts. We say that a function ρ mapping the variables contained in $\Gamma' \mid \Delta'$ to those in $\Gamma \mid \Delta$ is a *syntactic structural map* if

- ρ respects types: if $x : A$ in $\Gamma' \mid \Delta'$ then $\rho(x) : A$ in $\Gamma \mid \Delta$;
- ρ does not make variables more passive: if $\rho(x)$ is in Γ then x is in Γ';
- ρ does not contract active variables: if $\rho(x) = \rho(y)$ then either $x = y$ or x and y appear in Γ'.

We write $\rho : \Gamma \mid \Delta \to \Gamma' \mid \Delta'$ when this is the case.

Contexts and syntactic structural maps clearly form a category, with the usual composition of functions.

A morphism $JS[\![\Gamma]\!] \otimes [\![\Delta]\!] \to JS[\![\Gamma']\!] \otimes [\![\Delta']\!]$ in $\mathbf{C}$ is a *semantic structural map* if and only if it is of the form

$$JS\Gamma \otimes \Delta \xrightarrow{\mathsf{id}_{JS\Gamma} \otimes \cong} JS\Gamma \otimes \Delta_1 \otimes \Delta' \xrightarrow{f \otimes \mathsf{id}_{\Delta'}} JS\Gamma' \otimes \Delta'$$

where $\cong$ is a permutation isomorphism and $f : JS\Gamma \otimes \Delta_1 \to JS\Gamma'$ is a tuple of maps of the form π_i or $\pi_i; \eta$ for some i. (Recall that $JS\Gamma' = JSA'_1 \otimes \cdots \otimes JSA'_n$ is a product).

It is easy to check that the composition of two semantic structural maps is again a semantic structural map, so there is a category of contexts and semantic structural maps. Note also that a semantic structural map is completely determined by its composite with the projections from $JS\Gamma' \otimes \Delta'$ to the various JSC and D objects.

These two categories of structural maps are equivalent.

Lemma 8. *Let $\Gamma \mid \Delta$ and $\Gamma' \mid \Delta'$ be contexts. There is a 1-1 correspondence between semantic structural maps $JS[\![\Gamma]\!] \otimes [\![\Delta]\!] \to JS[\![\Gamma']\!] \otimes [\![\Delta']\!]$ and syntactic structural maps $\Gamma \mid \Delta \to \Gamma' \mid \Delta'$. This correspondence gives rise to an equivalence of the categories of semantic and syntactic structural maps.*

Proof. Given a semantic structural map $f : JS[\![\Gamma]\!] \otimes [\![\Delta]\!] \to JS[\![\Gamma']\!] \otimes [\![\Delta']\!]$, consider the composite maps $f; \pi_x$, where π_x is the projection from the tensor product $JS[\![\Gamma']\!] \otimes [\![\Delta']\!]$ to the component corresponding to the variable x.

By definition of semantic structural maps, each $f; \pi_x$ is either π_y or $\pi_y; \eta$ for some y. Define a function ρ by

$$\rho(x) = y \text{ if } f; \pi_x = \pi_y \text{ or } \pi_y; \eta.$$

The constraints in the definition of semantic structural maps imply that ρ is a syntactic structural map $\Gamma \mid \Delta \to \Gamma' \mid \Delta'$.

For the converse, first note that if $\rho : \Gamma \mid \Delta \to \Gamma' \mid \Delta'$ is a syntactic structural map, then $\rho \restriction \Delta'$ is an injective function with image in Δ. There is therefore a permutation $\phi : \Delta \to \Delta_1 \otimes \Delta'$ such that for each variable x in Δ', $\phi; \pi_x = \pi_{\rho(x)}$. We then define $f : JS\Gamma \otimes \Delta_1 \to JS\Gamma'$ as the unique map such that, for each x in Γ',

$$f; \pi_x = \begin{cases} \pi_{\rho(x)} & \text{if } \rho(x) \in \Gamma; \\ \pi_{\rho(x)}; \eta & \text{if } \rho(x) \in \Delta. \end{cases}$$

The required semantic structural map is then given by

$$JS\Gamma \otimes \Delta \xrightarrow{\mathsf{id} \otimes \phi} JS\Gamma \otimes \Delta_1 \otimes \Delta' \xrightarrow{f \otimes \mathsf{id}} JS\Gamma' \otimes \Delta'.$$

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

It is routine to check that the passages between semantic and syntactic structural maps are mutually inverse. Furthermore, they are functorial, so the two categories of semantic and syntactic structural maps are equivalent. □

Given a syntactic structural map ρ, we write $[\![\rho]\!]$, or sometimes just ρ, for the corresponding semantic structural map.

4.2. *Structural maps and structural rules*

Our definition of syntactic structural map is intended to capture a general notion of manipulation of contexts using the rules of contraction, weakening and activation. Indeed, any such rule of the form

$$\frac{\Gamma \mid \Delta \vdash M : A}{\Gamma' \mid \Delta' \vdash M' : A}$$

corresponds to a syntactic structural map $\Gamma' \mid \Delta' \to \Gamma \mid \Delta$ as follows.

The rule of activation

$$\frac{\Gamma, x : A \mid \Delta \vdash M : B}{\Gamma \mid \Delta, x : A \vdash M : B}$$

corresponds to the identity function, which is a valid structural map $\Gamma \mid \Delta, x : A \to \Gamma, x : A \mid \Delta$.

The rule of weakening

$$\frac{\Gamma \mid \Delta \vdash M : B}{\Gamma' \mid \Delta' \vdash M : B}\, \Gamma \subseteq \Gamma', \Delta \subseteq \Delta'$$

corresponds to the injection from $\Gamma \cup \Delta$ to $\Gamma' \cup \Delta'$.

The rule of contraction

$$\frac{\Gamma, x : A, y : A \mid \Delta \vdash M : B}{\Gamma, x : A \mid \Delta \vdash M[x/y] : B}$$

corresponds to the function which acts as the identity on $\Gamma \cup \Delta$ and maps both x and y to x.

Since structural maps form a category, any sequence of these structural rules also induces a syntactic structural map ρ. This map is nothing more than the substitution induced by the various contractions which are performed, and we will write $\rho(M)$ for the result of applying the substitutions indicated by ρ to the term M.

LEMMA 9. *Let Θ be a derivation of a judgement $\Gamma \mid \Delta \vdash M : A$, and let ϕ be a sequence of structural rules extending this derivation to yield $\Gamma' \mid \Delta' \vdash M' : A$. We write $\Theta; \phi$ for this extended derivation. Let ρ be the syntactic structural map corresponding to ϕ. Then*

$$[\![\Theta; \phi]\!] = [\![\rho]\!]; [\![\Theta]\!].$$

Proof. By induction on the length of the sequence ϕ. The base case is trivial, and for the inductive step it suffices to check that the lemma holds for the case when ϕ has length one, i.e. for the case of a single structural rule. This is straightforward:

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

one need only check that the semantics of each structural rule is obtained by precomposition with the appropriate semantic structural map, which is indeed the case. □

As an immediate corollary, we have that any two ways of extending a derivation with activation, contraction and weakening rules which induce the same substitution have the same semantics.

Corollary 10 (Coherence of Structural Extensions). *Let Θ be a derivation of a judgement $\Gamma \mid \Delta \vdash M : A$, and let ϕ_1 and ϕ_2 be sequences of structural rules which can both be used to extend this derivation, yielding $\Gamma' \mid \Delta' \vdash M' : A$ and inducing the same syntactic structural map. Then $[\![\Theta;\phi_1]\!] = [\![\Theta;\phi_2]\!]$.*

5. *Coherence*

We are now in a position to set about the proof of our central result, that models of SCIR in the sense we have described here are *coherent*, i.e. that any two derivations of a given judgement have the same semantics. Thus we have a semantics of *judgements*, not merely of derivations.

We can now summarize what we require in order to model SCIR and establish coherence.

Definition. A *categorical model of SCIR* consists of a symmetric monoidal category $\mathbf{C}$ with finite products and a full subcategory $\mathbf{P}$ of $\mathbf{C}$ with inclusion functor J, such that J has both a right adjoint P and a left adjoint S, with J and S being strong monoidal functors. Furthermore, there should be an appropriate collection of type objects as defined in Section 3.3.

A categorical model of SCIR is called *retractive* if, for every object A of $\mathbf{C}$, there is a map $\alpha_A : JSA \to A$ such that

$$\begin{tikzcd}
JSA \arrow[r, "\alpha_A"] \arrow[dr, "\mathsf{id}_{JSA}"'] & A \arrow[d, "\eta_A"] \\
 & JSA.
\end{tikzcd}$$

This condition deserves a little discussion. The terminology obviously comes from the fact that retractivity makes JSA a retract of A. It holds in all models studied to date: in bireflective models it is part of the definition, and in the games-based and object-space models it is a consequence of the interpretation of JS as an operation which projects a type onto its passive part. For both these models, we can view a type as consisting of a passive part and an active part, conjoined by (something like) a product operation, just as in the construction described in Section 3.1.1. Thanks to the computational nature of these models, every type is inhabited by at least one element, the computation which diverges i.e. enters an infinite loop. Thus we can always embed the passive part of a type in the whole type, by acting as identity on the passive part and diverging in the active part. This yields the required retraction, which may informally be described by the diagram below.

$$\begin{tikzcd}
X \arrow[r, "\langle \mathsf{diverge}, \mathsf{id} \rangle"] & A \times X \arrow[r, "\pi_2"] & X
\end{tikzcd}$$

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

(For the construction of Section 3.1.1, retractiveness fails in general, but holds if we restrict attention to pairs (A, X) in which A is non-empty, along the same lines as above.)

Note that we do not insist that α be a natural transformation. In fact, the following lemma holds.

Lemma 11. *A retractive model of SCIR in which α is a natural transformation is a bireflective model.*

Proof. The definition of bireflective model implies a retractive model with $\alpha = \epsilon'$ the counit of $J \dashv P$, which is natural.

In [4, 3] it is shown that bireflective subcategories are given by split idempotent natural transformations on the identity functor, and in a retractive model with α natural, $\eta; \alpha$ provides such a natural transformation, and hence a bireflection. □

The existence of the maps α_A makes it straightforward to calculate the passification of a map, as follows.

Lemma 12. *In a retractive model, given $f : A \otimes B \to C$ where C is passive i.e. $C \cong JX$ for some X,*

$$\mathsf{pass}_B(f) = \mathsf{id}_A \otimes \alpha_B; f : A \otimes JSB \to C.$$

Proof. By Lemma 5, it suffices to show that $\mathsf{id}_{\mathsf{A}} \otimes (\eta; \alpha); f = f$, and since η_C is an isomorphism for passive C, it is enough to show that

$$\mathsf{id}_{\mathsf{A}} \otimes (\eta; \alpha); f; \eta = f; \eta.$$

We calculate as follows:

$$\begin{aligned}
& (\mathsf{id}_{\mathsf{A}} \otimes (\eta; \alpha)); f; \eta_C \\
= \quad & (\mathsf{id}_{\mathsf{A}} \otimes (\eta; \alpha)); \eta_{A \otimes B}; JSf && (\eta \text{ natural}) \\
= \quad & (\mathsf{id}_{\mathsf{A}} \otimes (\eta; \alpha)); (\eta_A \otimes \eta_B); \cong; JSf && (\eta \text{ monoidal}) \\
= \quad & (\eta_A \otimes \eta_B; \alpha_B; \eta_B); \cong; JSf \\
= \quad & (\eta_A \otimes \eta_B); \cong; JSf && (\alpha; \eta = \mathsf{id}) \\
= \quad & \eta_{A \otimes B}; JSf && (\eta \text{ monoidal}) \\
= \quad & f; \eta_C && (\eta \text{ natural})
\end{aligned}$$

completing the proof. □

The image of a map under the functor JS can also be calculated using α.

Lemma 13. *In a retractive model, for any $f : A \to B$, $JSf = \alpha; f; \eta$.*

Proof. Consider the diagram below.

$$\begin{tikzcd}
JSA \arrow[r, "\alpha_A"] \arrow[dr, "\mathsf{id}"'] & A \arrow[r, "f"] \arrow[d, "\eta_A"] & B \arrow[d, "\eta_B"] \\
& JSA \arrow[r, "JSf"'] & JSB
\end{tikzcd}$$

The square commutes by naturality of η and the triangle by definition of α. □

The following lemma is proved by a strightforward inductive argument.

Lemma 14 (Strengthening). *Suppose Θ is a derivation of a judgement $\Gamma \mid \Delta \vdash M : A$, and that $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ are such that all free variables of M appear in $\Gamma' \cup \Delta'$. Then there is a derivation Θ' of $\Gamma' \mid \Delta' \vdash M : A$ such that*

$$\begin{tikzcd}
JS\Gamma \otimes \Delta \arrow[r, "\mathsf{weak}"] \arrow[dr, "\llbracket \Theta \rrbracket"'] & JS\Gamma' \otimes \Delta' \arrow[d, "\llbracket \Theta' \rrbracket"] \\
& A
\end{tikzcd}$$

where weak *is the appropriate weakening map. Moreover, the derivation Θ' is no larger (that is, uses no more rules) than Θ.*

Definition. Let $\Gamma \mid \Delta$ be a context and Δ_1, Δ_2 a permutation of Δ. In a retractive model, the maps α on the objects of Δ_1 together with some permutation maps give rise to a morphism

$$\begin{tikzcd}
JS\Gamma \otimes JS\Delta_1 \otimes \Delta_2 \arrow[r, "\mathsf{id} \otimes \alpha \otimes \mathsf{id}"] & JS\Gamma \otimes \Delta_1 \otimes \Delta_2 \arrow[r, "\cong"] & JS\Gamma \otimes \Delta.
\end{tikzcd}$$

We will use the notation

$$\alpha : \Gamma, \Delta_1 \mid \Delta_2 \to \Gamma \mid \Delta$$

for this map. We refer to such maps as *pseudo-passifications*, and say that α *makes passive* the variables in Δ_1.

Given such an α, if $\rho : \Gamma' \mid \Delta' \to \Gamma, \Delta_1 \mid \Delta_2$ is a structural map such that for every $x \in \Delta_1$, $\rho(x) \in \Gamma'$, we call the composite

$$\rho; \alpha : JS\Gamma' \otimes \Delta' \to JS\Gamma \otimes \Delta$$

a *pseudo-structural map (psm)*. That is, a psm is the composition of a structural map and a pseudo-passification, subject to the condition that any variable which α makes passive is left passive by ρ.

Our critique of the sketched proof of coherence for bireflective categories given in [8] hinged around the fact that one cannot push passifications up a derivation tree. We have introduced the (semantic) notion of pseudo-passification to overcome this difficulty. A derivation consisting of several structural rules followed by a passification corresponds semantically to a map $\alpha; \rho$ where ρ is a structural map and α is a pseudo-passification, here interpreting a genuine passification rule. The following lemma demonstrates that such a map can always be written as a psm $\rho'; \alpha'$. Thus if we work with psms rather than structural maps, we can push passifications up the (pseudo-)derivation, which facilitates an inductive proof of coherence.

Lemma 15. *The composition of a pseudo-structural map with a structural map yields a pseudo-structural map. That is, if $\rho_1; \alpha$ is a psm and ρ_2 is a structural map whose domain is the codomain of α, then there exists a psm $\rho'; \alpha'$ such that $\rho_1; \alpha; \rho_2 = \rho'; \alpha'$.*

Proof. It is straightforward to verify that any structural map can be written as a finite sequence of contractions, weakenings and activations. We proceed by induction on the length of such a sequence giving rise to the structural map ρ_2.

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

In the base case, ρ_2 is a single contraction, weakening or activation.

Suppose ρ_2 is an activation, activating variable z. If α makes z passive, the composite $\alpha;\rho_2$ takes the form

$$\Gamma,\Delta_1,z \mid \Delta_2 \xrightarrow{\alpha} \Gamma \mid \Delta, z \xrightarrow{\rho_2} \Gamma, z \mid \Delta.$$

This is equal to the pseudo-passification $\alpha' : \Gamma,\Delta_1,z \mid \Delta_2 \to \Gamma, z \mid \Delta$ which makes passive the same variables as α does, less z which is already passive in the codomain. Since α' makes fewer variables passive than does α, it is immediate that $\rho_1;\alpha'$ is a psm.

If α does not make z passive, the composite $\alpha;\rho_2$ takes the form

$$\Gamma,\Delta_1 \mid \Delta_2, z \xrightarrow{\alpha} \Gamma \mid \Delta, z \xrightarrow{\rho_2} \Gamma, z \mid \Delta.$$

There is a pseudo-passification $\alpha' : \Gamma,\Delta_1,z \mid \Delta_2 \to \Gamma, z \mid \Delta$, and an activation map $\rho_2' : \Gamma,\Delta_1 \mid \Delta_2, z \to \Gamma,\Delta_1,z \mid \Delta_2$ activating z. It is clear that $\alpha;\rho_2 = \rho_2';\alpha'$ so it just remains to show that $\rho_1;\rho_2';\alpha'$ is a psm, i.e. that the structural map $\rho_1;\rho_2'$ does not activate any variable made passive by α'. By construction $\rho_1;\rho_2'$ activates z and any variable activated by ρ_1. Since $\rho_1;\alpha$ is a psm, none of these variables is made passive by α and hence not by α'.

For the case when ρ_2 is a contraction, note that the composite

$$\Gamma,\Delta_1,x \mid \Delta_2 \xrightarrow{\alpha} \Gamma, x \mid \Delta \xrightarrow{\rho_2} \Gamma, x, y \mid \Delta$$

is equal to the psm

$$\Gamma,\Delta_1,x \mid \Delta_2 \xrightarrow{\rho_2'} \Gamma,\Delta_1,x,y \mid \Delta_2 \xrightarrow{\alpha'} \Gamma, x, y \mid \Delta$$

where ρ_2' is again a contraction and α' is a pseudo-passification which extends α by carrying the additional variable y along. Since $\rho_1;\alpha$ is a psm and α' makes the same variables passive as does α, $\rho_1;\rho_2';\alpha'$ is a psm as required.

The case when ρ_2 is a weakening is similar.

Finally for the inductive step, suppose $\rho_2 = \rho_2';\rho_2''$ and that the lemma holds for ρ_2' and ρ_2''. Then there is a psm $\rho';\alpha'$ such that $\rho';\alpha' = \rho_1;\alpha;\rho_2'$. Applying the inductive hypothesis again we find a psm $\rho'';\alpha''$ such that $\rho'';\alpha'' = \rho';\alpha';\rho_2'' = \rho_1;\alpha;\rho_2';\rho_2'' = \rho_1;\alpha;\rho_2$ as required. □

At last we are ready to embark on our inductive coherence proof.

LEMMA 16 (MAIN LEMMA). *Suppose Θ_1 and Θ_2 are derivations of $\Gamma_1 \mid \Delta_1 \vdash M_1 : A$ and $\Gamma_2 \mid \Delta_2 \vdash M_2 : A$ respectively. For $i = 1, 2$, let $\rho_i;\alpha_i : \Gamma \mid \Delta \to \Gamma_i \mid \Delta_i$ be psms such that $\rho_1(M_1) = \rho_2(M_2)$. Then in any retractive model,*

$$\rho_1;\alpha_1;[\![\Theta_1]\!] = \rho_2;\alpha_2;[\![\Theta_2]\!].$$

Proof. The proof is by induction on the sum of the sizes of the derivations Θ_1 and Θ_2. We approach the proof by cases on the last rules in these derivations; the base case is handled by the cases for constants and variables.

We first consider the case when one of the two derivations ends in a structural rule.

Suppose one of Θ_1, Θ_2 ends with a passification rule; without loss of generality let this be Θ_1, so our derivations have the following form.

$$\dfrac{\begin{array}{c}\vdots\;\Theta_1'\\ \Gamma_1 \mid \Delta_1, x \vdash M_1 : A\end{array}}{\Gamma_1, x \mid \Delta_1 \vdash M_1 : A} \qquad \begin{array}{c}\vdots\;\Theta_2\\ \Gamma_2 \mid \Delta_2 \vdash M_2 : A\end{array}$$

In this case A is a passive type and $\llbracket\Theta_1\rrbracket = \llbracket\alpha_x\rrbracket; \llbracket\Theta_1'\rrbracket$ where $\alpha_x : \Gamma_1, x \mid \Delta_1 \to \Gamma_1 \mid \Delta_1, x$ is the pseudo-passification map on the variable x.

We are given psms $\rho_1; \alpha_1$ and $\rho_2; \alpha_2$ such that $\rho_1(M_1) = \rho_2(M_2)$ and must show that

$$\rho_1; \alpha_1; \llbracket\Theta_1\rrbracket = \rho_2; \alpha_2; \llbracket\Theta_2\rrbracket.$$

Since the target type is passive, it will suffice to show that

$$\alpha; \rho_1; \alpha_1; \llbracket\Theta_1\rrbracket = \alpha; \rho_2; \alpha_2; \llbracket\Theta_2\rrbracket$$

where $\alpha : \Gamma\Delta \mid - \to \Gamma \mid \Delta$ is the pseudo-passification which passifies all active variables. By Lemma 15, there are psms $\rho_1; \alpha'$ and $\rho_2; \alpha''$ such that

$$\rho_1; \alpha' = \alpha; \rho_1$$

and

$$\rho_2; \alpha'' = \alpha; \rho_2.$$

It therefore suffices to show that

$$\rho_1; \alpha'; \alpha_1; \alpha_x; \llbracket\Theta_1'\rrbracket = \rho_2; \alpha''; \alpha_2; \llbracket\Theta_2\rrbracket.$$

The maps $\rho_1; \alpha'; \alpha_1; \alpha_x$ and $\rho_2; \alpha''; \alpha_2$ are psms since their source types contain no active variables, so we may apply the inductive hypothesis to the derivations Θ_1' and Θ_2 to complete this case.

Suppose now that (without loss of generality) Θ_1 ends in another structural rule: activation, contraction, weakening or exchange. Then Θ_1 has the form

$$\dfrac{\begin{array}{c}\vdots\;\Theta_1'\\ \Gamma_1' \mid \Delta_1' \vdash M_1' : A\end{array}}{\Gamma_1 \mid \Delta_1 \vdash M_1 : A}$$

and $\llbracket\Theta_1\rrbracket = \rho; \llbracket\Theta_1'\rrbracket$ for some structural map ρ, with $M_1 = \rho(M_1')$. Given psms $\alpha_i; \rho_i$ as in the statement of the lemma, we must show that

$$\rho_1; \alpha_1; \rho; \llbracket\Theta_1'\rrbracket = \rho_2; \alpha_2; \llbracket\Theta_2\rrbracket.$$

By Lemma 15, $\rho_1; \alpha_1; \rho$ can be written as a psm $\rho'; \alpha_1'$. We can now apply the inductive hypothesis to derivations Θ_1' and Θ_2 to conclude that

$$\rho'; \alpha_1'; \llbracket\Theta_1'\rrbracket = \rho_2; \alpha_2; \llbracket\Theta_2\rrbracket$$

which gives us the required result.

If neither derivation ends in a structural rule, they must both end with a term-forming rule, and since $\rho_1(M_1) = \rho_2(M_2)$, the terms M_1 and M_2 have the same outermost term constructor, so their derivations must end in instances of the same rule. We therefore proceed by cases according to which rule was used.

Variable. For $i = 1, 2$, let Θ_i be the derivation

$$\frac{}{- \mid x_i \vdash x_i : A}$$

and suppose we have psms $\rho_i; \alpha_i : \Gamma \mid \Delta \to - \mid x_i$ such that $\rho_1(x_1) = \rho_2(x_2)$. We must show that

$$\rho_1; \alpha_1; \llbracket x_1 \rrbracket = \rho_2; \alpha_2; \llbracket x_2 \rrbracket$$

but since $\llbracket x_1 \rrbracket = \llbracket x_2 \rrbracket = \mathsf{id}$, we just need to show that $\rho_1; \alpha_1 = \rho_2; \alpha_2$.

If $\rho_1(x_1) \in \Delta$, then since $\rho_1; \alpha_1$ is a psm, α_1 does not passify x_1 and hence α_1 is the identity. Thus $\rho_1 : \Gamma \mid \Delta \to - \mid x_1$ is simply a projection; similarly α_2 is the identity and ρ_2 is the appropriate projection, and the result follows.

If $\rho_1(x_1) \in \Gamma$, then α_1 must passify x_1, and similarly α_2 must passify x_2. It must then be the case that ρ_1 and ρ_2 are both equal to the projection from $\Gamma \mid \Delta$ to JSA, and $\alpha_1 = \alpha_2 = \alpha_A : JSA \to A$.

Promotion. Suppose for $i = 1, 2$ that Θ_i is of the form

$$\frac{\begin{array}{c}\Theta_i' \\ \vdots \\ \Gamma_i \mid - \vdash M_i : A\end{array}}{\Gamma_i \mid - \vdash \mathsf{prom}(M_i) : PA}$$

and we have psms $\rho_i; \alpha_i : \Gamma \mid \Delta \to \Gamma_i \mid -$ for which $\rho_1(\mathsf{prom}(M_1)) = \rho_2(\mathsf{prom}(M_2))$.

In this case, since the target contexts have no active variables, the pseudo-passifications must be identities: each ρ_i is a structural map $\Gamma \mid \Delta \to \Gamma_i \mid -$ and we must show that

$$\rho_1; \llbracket \Theta_1 \rrbracket = \rho_2; \llbracket \Theta_2 \rrbracket : JS\Gamma \otimes \Delta \to JPA.$$

Since the target type is passive, it suffices to show that

$$\alpha; \rho_1; \llbracket \Theta_1 \rrbracket = \alpha; \rho_2; \llbracket \Theta_2 \rrbracket : JS\Gamma \otimes JS\Delta \to JPA$$

where $\alpha : \Gamma\Delta \mid - \to \Gamma \mid \Delta$ is the pseudo-passification which passifies all variables.

By Lemma 15, each $\alpha; \rho_i$ can be written as a psm $\rho_i; \alpha' : \Gamma\Delta \mid - \to \Gamma_i \mid -$, and again since the target type is purely passive, α' is the identity, so ρ_i may be seen as a structural map $\Gamma\Delta \mid - \to \Gamma_i \mid -$ and we must show that

$$\rho_1; \llbracket \Theta_1 \rrbracket = \rho_2; \llbracket \Theta_2 \rrbracket : JS\Gamma \otimes JS\Delta \to JPA.$$

By Lemma 6, it will be enough to show that $\rho_1; \llbracket \Theta_1 \rrbracket; \epsilon'_A = \rho_2; \llbracket \Theta_2 \rrbracket; \epsilon'_A$. Since by definition $\llbracket \Theta_i \rrbracket = \mathsf{prom}(\llbracket \Theta_i' \rrbracket)$, we have that $\llbracket \Theta_i \rrbracket; \epsilon'_A = \llbracket \Theta_i' \rrbracket$. Since $\rho_1(\mathsf{prom}(M_1)) = \rho_2(\mathsf{prom}(M_2))$, we also have $\rho_1(M_1) = \rho_2(M_2)$, and we can apply the inductive hypothesis to the derivations Θ_i' to conclude that

$$\rho_1; \llbracket \Theta_1' \rrbracket = \rho_2; \llbracket \Theta_2' \rrbracket$$

as required.

Dereliction, constants, pairing, projection, abstraction. The cases where the derivations end in a dereliction, constant, pairing, projection or abstraction rule are simple applications of the inductive hypothesis. We shall give the case of abstraction as an illustration.

Suppose the Θ_i have the form

$$\dfrac{\begin{array}{c}\Theta_i'\\ \vdots\\ \Gamma_i \mid \Delta_i, x : A \vdash M_i : B\end{array}}{\Gamma_i \mid \Delta_i \vdash \lambda x.M_i : A \to B}$$

and suppose there are psms $\rho_i ; \alpha_i : \Gamma \mid \Delta \to \Gamma_i \mid \Delta_i$ such that $\rho_1(\lambda x.M_1) = \rho_2(\lambda x.M_2)$. Let ρ_i' be the structural map which extends ρ_i with the identity action on the variable x, so that $\rho_1'(M_1) = \rho_2'(M_2)$. Similarly let α_i' extend α_i with the identity action on x. It is clear that $\rho_i' ; \alpha_i' : \Gamma \mid \Delta, x \to \Gamma_i \mid \Delta_i, x$ are psms, and by inductive hypothesis,

$$\rho_1' ; \alpha_1' ; [\![\Theta_1']\!] = \rho_2' ; \alpha_2' ; [\![\Theta_2']\!] : \Gamma \mid \Delta, x \to B.$$

Currying $\rho_i' ; \alpha_i' ; [\![\Theta_i']\!]$ yields $\rho_i ; \alpha_i ; [\![\Theta_i]\!]$ and the proof is complete.

Application. We now come to the most troublesome case, that of application. Suppose Θ_i has the form

$$\dfrac{\begin{array}{c}\phi_i\\ \vdots\\ \Gamma_i \mid \Delta_i \vdash M_i : A_i \to B\end{array} \qquad \begin{array}{c}\phi_i'\\ \vdots\\ \Gamma_i' \mid \Delta_i' \vdash N_i : A_i\end{array}}{\Gamma_i\Gamma_i' \mid \Delta_i\Delta_i' \vdash M_iN_i : B}$$

and we have psms

$$\begin{tikzcd}\Gamma \mid \Delta \arrow[r, "\rho_i"] & \Gamma_i\Gamma_i'\Delta_{i,1}\Delta_{i,1}' \mid \Delta_{i,2}\Delta_{i,2}' \arrow[r, "\alpha_i"] & \Gamma_i\Gamma_i' \mid \Delta_i\Delta_i'\end{tikzcd}$$

(where $\Delta_{i,1}, \Delta_{i,2}$ is a permutation of Δ_i and similarly for the $\Delta_{i,j}'$) such that $\rho_1(M_1) = \rho_2(M_2)$ and $\rho_1(N_1) = \rho_2(N_2)$.

First note that the argument types A_1 and A_2 must be the same: this holds because N_1 and N_2 can be unified by the substitutions ρ_1 and ρ_2, which preserve the types of free variables, and because a term can have at most one type once the types of its free variables are specified.

We shall assume for now that every variable in the contexts $\Gamma_i\Gamma_i' \mid \Delta_i\Delta_i'$ appears free in M_iN_i; we will extend our argument to the general case via the strengthening lemma (Lemma 14) later.

Define a new map σ_1 on the variables in $\Gamma_1\Delta_{1,1} \mid \Delta_{1,2}$ (exactly the free variables of M_1) by

$$\sigma_1(x) = \rho_1(x)_M$$

i.e. $\sigma_1(x)$ is a variable resulting from tagging $\rho_1(x)$ with the symbol M. We assume this yields a completely fresh variable symbol.

Similarly define σ_2 with domain $\Gamma_2\Delta_{2,1} \mid \Delta_{2,2}$ (the free variables of M_2) by

$$\sigma_2(x) = \rho_2(x)_M.$$

We shall use the following notations: if f is a function which relabels variables (such as the σ_i), and Γ is a context, then $f(\Gamma)$ is the context resulting from relabelling the variables in Γ according to f; we also use set-operations on contexts, so that $\Gamma \cap \Gamma'$ is the context containing those variables in both Γ and Δ, and so on. The order in which these variables appear is not relevant.

Define two new contexts as follows:

$$\begin{aligned} \Gamma_M &= \sigma_1(\Gamma_1\Delta_{1,1}) \cap \sigma_2(\Gamma_2\Delta_{2,1}) \qquad \text{and} \\ \Delta_M &= \sigma_1(\Gamma_1\Delta_{1,1} \mid \Delta_{1,2}) \setminus \Gamma_M. \end{aligned}$$

(Note that since $\rho_1(M_1) = \rho_2(M_2)$, the functions σ_1 and σ_2 have the same image, so Δ_M could just as well have been defined as $\sigma_2(\Gamma_2\Delta_{2,1} \mid \Delta_{2,2}) \setminus \Gamma_M$.)

We now claim that $\sigma_1 : \Gamma_M \mid \Delta_M \to \Gamma_1\Delta_{1,1} \mid \Delta_{1,2}$ is a structural map which does not activate any variable of $\Delta_{1,1}$.

- Suppose $\sigma_1(x) = \sigma_1(y)$ and $x \neq y$. Then $\rho_1(x) = \rho_1(y)$, so $x, y \in \Gamma_1\Gamma_1'\Delta_{1,1}\Delta_{1,1}'$ since ρ_1 is a structural map. Therefore $x, y \in \Gamma_1\Delta_{1,1}$ as required.
- Suppose $\sigma_1(x) \in \Gamma_M$. By definition of Γ_M, $\sigma_1(x)$ must be equal to some $\sigma_1(y)$ where $y \in \Gamma_1\Delta_{1,1}$. If $x = y$ then $x \in \Gamma_1\Delta_{1,1}$ as required. Otherwise we have $\sigma_1(x) = \sigma_1(y)$ and $x \neq y$, so we can argue that $x \in \Gamma_1\Delta_{1,1}$ as above.
- Finally, suppose $x \in \Delta_{1,1}$. Since $\rho_1; \alpha_1$ is a psm, and α_1 passifies all of $\Delta_{1,1}$ by definition, ρ_1 does not activate x, so $\rho_1(x) \in \Gamma$. Since ρ_1 and ρ_2 have the same image, $\rho_1(x) = \rho_2(y)$ for some $y \in \Gamma_2\Delta_{2,1} \mid \Delta_{2,2}$. Since $\rho_2(y) \in \Gamma$, it must be that $y \in \Gamma_2\Delta_{2,1}$, and so $\sigma_2(y) \in \sigma_2(\Gamma_2\Delta_{2,1})$. But $\sigma_2(y) = \rho_2(y)_M = \rho_1(x)_M = \sigma_1(x)$. Thus $\sigma_1(x) \in \sigma_2(\Gamma_2\Delta_{2,1})$. By definition we also have $\sigma_1(x) \in \sigma_1(\Gamma_1\Delta_{1,1})$, so that $\sigma_1(x) \in \Gamma_M$ as required.

Similarly we can define structural maps

$$\sigma_i' : \Gamma_N \mid \Delta_N \to \Gamma_i'\Delta_{i,1}' \mid \Delta_{i,2}'$$

by $\sigma_i'(x) = \rho_i(x)_N$. As above, these maps do not activate any of the variables in the $\Delta_{i,1}'$.

Note immediately that $\sigma_1(M_1) = \sigma_2(M_2)$ and $\sigma_1'(N_1) = \sigma_2'(N_2)$. Furthermore, the pseudo-passifications α_i can each be split into two pseudo-passifications:

$$\beta_i : \Gamma_i\Delta_{i,1} \mid \Delta_{i,2} \to \Gamma_i \mid \Delta_i$$

$$\beta_i' : \Gamma_i'\Delta_{i,1}' \mid \Delta_{i,2}' \to \Gamma_i' \mid \Delta_i'$$

such that

$$\sigma_i; \beta_i : \Gamma_M \mid \Delta_M \to \Gamma_i \mid \Delta_i$$

and

$$\sigma_i'; \beta_i' : \Gamma_N \mid \Delta_N \to \Gamma_i' \mid \Delta_i'$$

are psms.

We can now apply the inductive hypothesis to the derivations ϕ_i and ϕ_i' to establish that

$$\begin{aligned} \sigma_1; \beta_1; [\![\phi_1]\!] &= \sigma_2; \beta_2; [\![\phi_2]\!] \qquad \text{and} \\ \sigma_1'; \beta_1'; [\![\phi_1']\!] &= \sigma_2'; \beta_2'; [\![\phi_2']\!]. \end{aligned}$$

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

By definition, $[\![\Theta_i]\!]$ is given by

$$\Gamma_i\Gamma'_i \mid \Delta_i\Delta'_i \xrightarrow{[\![\phi_1]\!] \otimes [\![\phi_2]\!]} (A \to B) \otimes A \xrightarrow{\mathsf{ev}} B.$$

Eliding permutations, we have psms

$$\Gamma_M\Gamma_N \mid \Delta_M\Delta_N \xrightarrow{\sigma_i \otimes \sigma'_i} \Gamma_i\Gamma'_i\Delta_{i,1}\Delta'_{i,1} \mid \Delta_{i,2}\Delta'_{i,2} \xrightarrow{\beta_i \otimes \beta'_i} \Gamma_i\Gamma'_i \mid \Delta_i\Delta'_i$$

and it follows that

$$\sigma_1 \otimes \sigma'_1; \beta_1 \otimes \beta'_1; [\![\Theta_1]\!] = \sigma_2 \otimes \sigma'_2; \beta_2 \otimes \beta'_2; [\![\Theta_2]\!].$$

We now define a map σ on the variables $\Gamma_M\Gamma_N \mid \Delta_M\Delta_N$ which removes the M and N tags:

$$\sigma(x_M) = x \qquad \text{and} \qquad \sigma(x_N) = x.$$

We claim that this is a structural map

$$\sigma : \Gamma \mid \Delta \to \Gamma_M\Gamma_N \mid \Delta_M\Delta_N.$$

- Suppose $\sigma(x) = \sigma(y) = z$ and $x \neq y$. We must show that $x, y \in \Gamma_M\Gamma_N$. Without loss of generality, x is z_M and y is z_N, and we need to show that $z_M \in \Gamma_M$ and $z_N \in \Gamma_N$.
  By definition, $\Gamma_M \mid \Delta_M$ is the image of σ_1, so $z_M = \sigma_1(w)$ for some $w \in \Gamma_1\Delta_{1,1} \mid \Delta_{1,2}$. Similarly $z_N = \sigma'_1(w')$ for some $w' \in \Gamma'_1\Delta'_{1,1} \mid \Delta'_{1,2}$. Thus $w \neq w'$ but $\rho_1(w) = z = \rho_1(w')$. Since ρ_1 is a structural map, $w \in \Gamma_1\Delta_{1,1}$ and $w' \in \Gamma'_1\Delta'_{1,1}$. Therefore $z_M \in \sigma_1(\Gamma_1\Delta_{1,1})$ and $z_N \in \sigma'_1(\Gamma'_1\Delta'_{1,1})$.
  Similarly we can establish that $z_M \in \sigma_2(\Gamma_2\Delta_{2,1})$ and $z_N \in \sigma'_2(\Gamma'_2\Delta'_{2,1})$. Therefore $z_M \in \Gamma_M$ and $z_N \in \Gamma_N$ as required.
- Suppose that $\sigma(x) \in \Gamma$. We must show that $x \in \Gamma_M\Gamma_N$. Without loss of generality let $x = y_M$; we shall show $y_M \in \Gamma_M$.
  Since $\Gamma_M \mid \Delta_M$ is the image of σ_1, $y_M = \sigma_1(z)$ for some $z \in \Gamma_1\Delta_{1,1} \mid \Delta_{1,2}$. Then $\rho_1(z) = y = \sigma(x) \in \Gamma$, so $z \in \Gamma_1\Gamma'_1\Delta_{1,1}\Delta'_{1,1}$, i.e. $z \in \Gamma_1\Delta_{1,1}$. Therefore $y_M = \sigma_1(z) \in \sigma_1(\Gamma_1\Delta_{1,1})$.
  Similarly we can argue that $y_M \in \sigma_2(\Gamma_2\Delta_{2,1})$, and therefore $y_M \in \Gamma_M$.

Since σ is a structural map, it has a semantic counterpart which we shall also write as σ. Clearly

$$\sigma; \sigma_i \otimes \sigma'_i; \beta_i \otimes \beta'_i = \rho_i; \alpha_i$$

and hence

$$\rho_1; \alpha_1; [\![\Theta_1]\!] = \rho_2; \alpha_2; [\![\Theta_2]\!]$$

completing the proof.

Finally we must show that our assumption that all variables in the context appear free in the term may be lifted. Suppose we have derivations Θ_1, Θ_2 as above, but without this stipulation, and psms $\rho_i; \alpha_i$ as before. By Lemma 14, there are derivations Θ'_1 and Θ'_2 in which all variables appearing in the context do appear free in the terms, and weakening maps weak_i, such that $\mathsf{weak}_i; [\![\Theta'_i]\!] = [\![\Theta_i]\!]$.

By Lemma 15, each $\rho_i; \alpha_i; \mathsf{weak}_i$ can be written as a psm $\rho'_i; \alpha'_i$. We can then apply the above argument to show that $\rho'_1; \alpha'_1; [\![\Theta'_1]\!] = \rho'_2; \alpha'_2; [\![\Theta'_2]\!]$, completing this case. □

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

Our coherence theorem is a straightforward corollary of this result.

THEOREM 17 (COHERENCE FOR RETRACTIVE MODELS). *If Θ_1 and Θ_2 are derivations of a judgement $\Gamma \mid \Delta \vdash M : A$ then in any retractive model, $[\![\Theta]\!] = [\![\Theta']\!]$.*

Proof. Apply the main lemma using the identity structural maps and pseudo-passifications. □

6. *Conclusions and further work*

We have presented a notion of categorical model for the SCIR system which subsumes the original notion of bireflective model, and additionally incorporates two new concrete models, the games model of [**14**] and the object-spaces model of [**9**]. We have established the coherence of such models, subject to the constraint of retractivity: any two derivations of the same judgement have the same denotation, so that our models provide a semantics of judgements rather than merely of derivations. Thus the foundational material required for the development of the games and object-spaces models is belatedly in place.

Our results apply only to retractive models. All models that have been studied to date do indeed enjoy the extra property of retractivity, but the type system does not seem to require it. This suggests two questions: are non-retractive models coherent, and is there a natural way to incorporate retractivity in the type-system?

While distinguishing the two functors S and P eliminates certain degeneracies from the notion of model, it also takes the definition of model away from the type system: there is no representation of the functor JS in the syntax. It would be interesting to develop a language which incorporates S as a type-constructor. Perhaps this would allow us to eliminate the two-zoned judgements in favour of more traditional ones (possibly even two kinds of judgement, corresponding to the two categories at hand) and to establish a model/theory correspondence. Such a study might also shed light on the role of the retractivity condition.

In another direction, a deeper investigation into the power of multiple-zoned contexts might yield interesting results. For example, can language-based security be treated in this way, with the various zones in the contexts representing different security levels? If so, can the kind of categorical setup described here be used as a semantics?

References

1. A. BARBER, ‘Dual intuitionistic linear logic’, Tech. Rep. ECS-LFCS-96-347, LFCS, University of Edinburgh, 1996. 177

2. R. E. BASHIR and J. VELEBIL, ‘Simultaneously reflective and coreflective subcategories of presheaves’, *Theory and Applications of Categories* 10 (2002) 410–423. 178

3. P. J. FREYD, P. W. O’HEARN, A. J. POWER, M. TAKEYAMA, R. STREET and R. D. TENNENT, ‘Bireflectivity’, *Theoretical Computer Science* 228 (1999) 49–76; a preliminary version appeared in the proceedings of MFPS XI. 177, 197

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press

4. P. J. Freyd, P. W. O'Hearn, A. J. Power, M. Takeyama and R. D. Tennent, 'Bireflectivity', *Mathematical foundations of programming semantics*, Eleventh annual conference, Tulane University, New Orleans, LA, March 29 – April 1, 1995 (Elsevier, 1995). 177, 197

5. H. Huang and U. Reddy, 'Type reconstruction for syntactic control of interference', *Glasgow Functional Programming Workshop,* BCS Electronic Workshops in Computer Science (1995). 177

6. J. M. E. Hyland and C.-H. L. Ong, 'On full abstraction for PCF: I, II and III', *Information and Computation* 162 (2000) 285–408. 180

7. S. P. Jones and J. Launchbury, 'State in haskell', *Lisp and Symbolic Computation* 8 (1995) 293–341. 181

8. P. W. O'Hearn, A. J. Power, M. Takeyama and R. D. Tennent, 'Syntactic control of interference revisited', *Theoretical Computer Science* 228 (1999) 211–252; a preliminary version appeared in the proceedings of MFPS XI. 177, 181, 191, 198

9. U. S. Reddy, 'Global state considered unnecessary: An introduction to object-based semantics', *Lisp and Symbolic Computation* 9 (1996) 7–76. 177, 182, 205

10. J. C. Reynolds, 'Syntactic control of interference', *Conf. Record 5th ACM Symposium on Principles of Programming Languages* (1978) 39–46. 176

11. J. C. Reynolds, 'Idealized Algol and its specification logic', *Tools and notions for program construction* (ed. D. Néel, Cambridge University Press, 1982) 121–161. 176

12. V. Swarup, U. S. Reddy and E. Ireland, 'Assignments for applicative languages', *Proceedings of the 5th ACM Conference on Functional Programming Languages and Computer Architecture* (Springer, New York, 1991) 192–214. 177, 182

13. V. Swarup, U. S. Reddy and E. Ireland, 'Assignments for applicative languages', *Algol-like languages* (ed. Peter W. O'Hearn and Robert D. Tennent, Birkhaüser, 1997) 235–271. 177

14. M. Wall, 'Games for syntactic control of interference', PhD thesis, University of Sussex, 2005. 177, 180, 182, 205

15. H. Yang and H. Huang, 'Type reconstruction for syntactic control of interference, part 2', *IEEE Computer Society International Conference on Computer Languages 1998*, Loyola University, Chicago (IEEE, Los Alamitos, CA, 1998) 164–173. 177

Guy McCusker G.A.McCusker@bath.ac.uk

Department of Computer Science
University of Bath
United Kingdom BA2 7AY

https://doi.org/10.1112/S1461157000001364 Published online by Cambridge University Press