

Logical Semantics for Stability^{*}

Luca Paolini¹

*Dipartimento di Informatica
Università di Torino
Torino, Italy*

Mauro Piccolo²

*Dipartimento di Informatica
Università di Torino
Torino, Italy
Laboratoire PPS, Université de Paris VII
Paris, France*

Simona Ronchi Della Rocca³

*Dipartimento di Informatica
Università di Torino
Torino, Italy*

Abstract

Type assignment systems for λ -calculus based on intersection types are a general framework for building models of λ -calculus (known as filter-models) which are useful tools for reasoning in a finitary way about the denotational interpretation of terms. Indeed the denotation of a term is the set of types derivable for it and a type is a “finite piece” of information on such a denotation. This approach to the λ -calculus semantics is called in the literature *logical semantics*, and it has been intensively studied in relation with λ -models in the Scott’s domain setting. In this paper we define two intersection type assignment systems for λ -calculus, parametric with respect to a coherence relation between types. We prove that, when the instantiation of the parameter satisfies a given condition, they induce models of λ -calculus, that we call *clique-models*. Lastly we show that such systems give a logical characterization of two classes of models built on the category of Girard’s coherence spaces and stable functions.

Keywords: λ -calculus, logical semantics, intersection type assignment systems, coherence spaces.

1 Introduction

In the general framework of denotational semantics for programming languages, logical semantics is a tool for building models of various kinds of λ -calculus, and

^{*} Paper partially supported by MIUR-PRIN’07 CONCERTO Project.

¹ Email: paolini@di.unito.it

² Email: piccolo@di.unito.it

³ Email: ronchi@di.unito.it

for reasoning in finitary way about the interpretation of terms. Logical semantics is based on intersection type assignment systems [8]. Namely, an intersection type assignment system assigns types to terms of λ -calculus, and typing rules are closed under a pre-order relation. If the pre-order satisfies some constraints then the system gives rise to a λ -model, where the interpretation of a term is the set of types derivable for it. Such constraints are designed for assuring that the interpretation of a term enjoys the good properties we expect, like the closure under the operational equality, the closure under contexts, and so on. In all the known instances, the constraints can be expressed by few, very easy, syntactical rules.

Logical semantics has been intensively studied in connections with denotational model based on Scott's domains which are ω -algebraic complete lattices (Scott-models). Indeed, the constraints can be formalized in such a way that the induced model is isomorphic to a Scott-model. The models built in this way are called *filter λ -models*. The isomorphism between filter λ -models and Scott-models can be view as a particular instance of domain theories in logical form [2]. The isomorphism relates types with compact elements of the complete lattice on which the Scott-model is based, in particular arrow-types correspond to a Scott-continuous step-function. The pre-order between types is the partial order of the domain. The construction is made by using the intersection connective between types for mimicking the join operation in domains. Moreover, the type assignment system provides a logical description of the interpretation function, i.e. to assign a type (say σ) to a term M is the logical counterpart of the fact that the compact element corresponding to σ is less than or equal to the interpretation of M . Examples of filter models designed in order to study particular properties of λ -calculus are in [5,10,11,23]. This approach has been applied also to the call-by-value λ -calculus in [12]. For the study of the isomorphism between filter λ -models and Scott-models the reader can see [9,18,22], behind other. In [23] a notion of *parametric filter model* has been defined, which can generate models of various kinds of λ -calculus, when the parameter has been specified in a suitable way (particular choices generate the classical and the call-by-value λ -calculus).

In this paper we want to extend the above approach to the λ -models based on coherence spaces defined by Girard [13] (originally they was named binary qualitative domains, their renaming in coherence spaces has been given in [14]). Coherence spaces are based on Berry's stable functions [7]. We call such models *stable λ -models*. We consider two particular classes of stable λ -models, the linear and the lazy one. In order to describe such classes, let us recall that a stable λ -model is based on a coherence space X containing as retract the space $X \Rightarrow X$, where \Rightarrow denotes the stable functions constructor. A stable model is linear, if both the functions realizing the retraction are linear (and so they are strict, since they map the empty set into itself). Lazy stable models are particular cases of stable models where one of the functions realizing the retraction is not strict. This difference in the space on which the models are based is reflected in the λ -theories that they induce. In fact, the λ -theory induced by a linear λ -model is always such that, if a term M is interpreted as the empty set, then also the interpretation of $\lambda x.M$ is the empty set. So the theory cannot be lazy, in the sense of [3]. While lazy stable models

can generate lazy λ -theories. We want to stress the fact that these two classes do not represent strong restrictions: indeed, in our knowledge, all stable λ -models on coherence spaces considered in literature belong to one of them [6,13,17].

We define two type assignment systems, parametric with respect to two relations between types, an equivalence and a strict coherence relation. We provide a *legality condition*, and we prove that every choice of such relations satisfying this condition gives rise to a λ -model. We call respectively *clique models* and *lazy clique models* the λ -models obtained by the two type assignments. Then, we prove that the class of clique models is isomorphic to the class of linear stable models, while the class of lazy clique models is isomorphic to that one of lazy stable models. Such isomorphisms are based on the fact that equivalent classes of types can be put in correspondence with tokens, in such a way that the equivalence and strict coherence relation between types reflect respectively the equality and the strict coherence relation in the space on which the (lazy) stable model is based, when the legality condition is satisfied. Lastly, the type assignment system is a logical description of the interpretation function. In fact, if a type σ is derivable for a term M , then it turns out that the token corresponding to the equivalence class of σ belongs to the clique which is the interpretation of M in the isomorphic stable model.

The main difference between our type assignments and the classical intersection type assignments reflects the difference between continuous and stable functions. Classical intersection types represent compact elements of complete lattices, where all elements are consistent. In fact the join is a total operation on Scott domains, and so the intersection is a total constructor on types. In coherence spaces the join is a partial operation, defined only between coherent elements, so we need to introduce in types a notion reflecting this fact, which is the strict coherence relation. Moreover, we choose to not use explicitly the intersection constructor on types, but our types are of the shape $[\sigma_1, \dots, \sigma_n] \rightarrow \sigma$ where the left-hand side of the arrow contains types which are strictly coherent among each-other (morally, in intersection). The fundamental difference between clique models and lazy clique models is the introduction of a particular type constant ν denoting the least functional behaviour. So, in order to type an application, we ask that the term in functional position can be assigned both the type ν and the type describing its correct functionality. We want to stress the fact that the definition of (lazy)-clique models is given in a completely syntactical way, so they can be built without any acquaintance with stable functions and stable models.

In the literature some works has been done to connect intersection type assignment systems and stable λ -models, namely [6,17]. In the last section we will discuss the relation between the results of the present paper and the previous ones.

Outline of the paper.

The paper is organized as follows. In Section 2 the notion of model of λ -calculus is recalled. In Section 3 the two parametric type assignment systems and the related clique models are defined. Section 4 contains a short survey on the principal notions about coherence spaces and stable functions. In Section 5 we prove the isomorphism between clique-models (lazy clique models) and linear and lazy stable λ -models

respectively. Finally Section 6 contains a comparison between the present paper and the papers [6,17].

2 λ -models

In this section we will briefly recall the definition of λ -calculus and λ -model.

Definition 2.1 (i) Terms of the λ -calculus are defined by the following syntax:

$$M ::= x \mid \lambda x.M \mid MM$$

where x belongs to a countable set Var of variables. Variables are ranged over by x, y, z and terms by M, N, P .

(ii) The β -reduction is the contextual closure of the following rule:

$$(\lambda x.M)N \rightarrow_{\beta} M[N/x]$$

where $M[N/x]$ denotes the capture free substitution of all occurrences of x in M by N . $=_{\beta}$ is the minimal equivalence induced by \rightarrow_{β} .

We will use notations from [4]. \equiv denotes the syntactical identity between terms. $\text{FV}(M)$ denotes the set of variables occurring free in M . As usual terms are considered up to α -equivalence, i.e. bound variables renaming avoiding variable clashes.

Let us recall two particular classes of λ -terms, the head normal form and the weak head normal forms. A term M has head normal form if $M =_{\beta} \lambda x_1 \dots x_n. zM_1 \dots M_m$, ($n, m \geq 0$) while it has weak head normal form if either $M =_{\beta} \lambda x.N$ or $M =_{\beta} zM_1 \dots M_m$, for some z, M_1, \dots, M_m, N ($m \geq 0$).

In the next definition, we will give the properties that a structure must satisfy in order to be used as denotation space for λ -calculus, or, equivalently, to be a *model* for λ -calculus [4,16,19,20]. In particular, we use the definition given in [23] which is a light equivalent variant of the one in [16].

Definition 2.2 A λ -model is a triple $\mathcal{M} = \langle \mathbb{D}, \circ, \llbracket \cdot \rrbracket \rangle$, such that \mathbb{D} is a set and \circ is a map from \mathbb{D}^2 to \mathbb{D} . Moreover, if Env is the collection of functions (environments) from Var to \mathbb{D} , ranged over by ρ, ρ' , then the *interpretation function* $\llbracket \cdot \rrbracket : \Lambda \times \text{Env} \rightarrow \mathbb{D}$ satisfies the following conditions:

1. $\llbracket x \rrbracket_{\rho} = \rho(x)$,
2. $\llbracket MN \rrbracket_{\rho} = \llbracket M \rrbracket_{\rho} \circ \llbracket N \rrbracket_{\rho}$,
3. $\llbracket \lambda x.M \rrbracket_{\rho} \circ d = \llbracket M \rrbracket_{\rho[d/x]}$,
4. if $\llbracket M \rrbracket_{\rho[d/x]} = \llbracket M' \rrbracket_{\rho'[d/y]}$ for each $d \in \mathbb{D}$, then $\llbracket \lambda x.M \rrbracket_{\rho} = \llbracket \lambda y.M' \rrbracket_{\rho'}$,

where $\rho[d/x](y) = \rho(y)$ if $y \neq x$ and d if $y = x$.

This definition ensures that a λ -model respects some elementary key properties, namely the interpretation of a term depends only on the behaviour of the environment on the free variables of the term itself, the α -rule is respected, the syntactical substitution is modeled by the environment and the interpretation is contextually closed.

Proposition 2.3 *Let $\langle \mathbb{D}, \circ, \llbracket \cdot \rrbracket \rangle$ be a λ -model.*

- (i) *If $\rho(\mathbf{x}) = \rho'(\mathbf{x})$, for all $\mathbf{x} \in \text{FV}(\mathbf{M})$, then $\llbracket \mathbf{M} \rrbracket_\rho = \llbracket \mathbf{M} \rrbracket_{\rho'}$;*
- (ii) *If $y \notin \text{FV}(\mathbf{M})$ then $\llbracket \mathbf{M} \rrbracket_{\rho[d/x]} = \llbracket \llbracket \mathbf{M}[y/x] \rrbracket_{\rho[d/y]} \rrbracket$;*
- (iii) *If $y \notin \text{FV}(\mathbf{M})$ then $\llbracket \lambda \mathbf{x}. \mathbf{M} \rrbracket_\rho = \llbracket \lambda y. \llbracket \mathbf{M}[y/x] \rrbracket_\rho \rrbracket$;*
- (iv) $\llbracket \llbracket \mathbf{M}[N/x] \rrbracket_\rho \rrbracket = \llbracket \llbracket \mathbf{M} \rrbracket_{\rho[\llbracket N \rrbracket_\rho/x]} \rrbracket$;
- (v) *If $\llbracket \mathbf{M} \rrbracket_\rho = \llbracket \mathbf{N} \rrbracket_\rho$ then, for every context $C[\cdot]$, $\llbracket C[\mathbf{M}] \rrbracket_\rho = \llbracket C[\mathbf{N}] \rrbracket_\rho$.*

As consequence of the previous proposition, condition (iii) of Definition 2.2 is the semantics counterpart of the β -reduction rule, so the interpretation of a term is closed under $=_\beta$.

Corollary 2.4 *Let $\langle \mathbb{D}, \circ, \llbracket \cdot \rrbracket \rangle$ be a λ -model. If $\mathbf{M} =_\beta \mathbf{N}$ then $\llbracket \mathbf{M} \rrbracket_\rho = \llbracket \mathbf{N} \rrbracket_\rho$, for all ρ .*

Given a λ -model \mathcal{M} , the interpretation function $\llbracket \cdot \rrbracket^{\mathcal{M}}$ induces a denotational semantics on Λ . Namely, two terms \mathbf{M} and \mathbf{N} are denotationally equivalent in \mathcal{M} (and we write $\mathbf{M} \sim_{\mathcal{M}} \mathbf{N}$) if and only if:

$$\llbracket \mathbf{M} \rrbracket_\rho^{\mathcal{M}} = \llbracket \mathbf{N} \rrbracket_\rho^{\mathcal{M}}, \text{ for all environments } \rho.$$

Corollary 2.4 ensure us that $\sim_{\mathcal{M}}$ is a λ -theory, i.e., a congruence relation on terms closed under $=_\beta$.

3 Clique Models

In this section we will define two classes of type assignment systems, and we will prove that both give rise to λ -models, under particular conditions.

In order to present the systems in a clean way, we first define a superset of types (row types) which enable us to formalize *some relations on them* needed to define well-formed types.

Definition 3.1 (i) Let C be a non-empty countable set of *type constants*, ranged over by p, q, r . The set $\text{Row}(C)$ of *row types*, ranged over by σ, τ, π, μ , is defined as follows:

$$\sigma ::= p \mid [\sigma_1, \dots, \sigma_n] \rightarrow \sigma \quad (n \geq 0)$$

where p belongs to C . The identity relation on $\text{Row}(C)$ is denoted by $=$.

(ii) A *stable type theory* on $\text{Row}(C)$ is a congruence \simeq (i.e. a reflexive, symmetric, transitive and contextual equivalence) on row types, satisfying

$$\frac{\forall i \exists j \sigma_i \simeq \tau_j \quad \forall i \exists j \tau_i \simeq \sigma_j \quad \sigma \simeq \tau \quad n, m \geq 0}{[\sigma_1, \dots, \sigma_n] \rightarrow \sigma \simeq [\tau_1, \dots, \tau_m] \rightarrow \tau}$$

We denote by $\text{Row}(C)/\simeq$ the set of equivalence classes induced by \simeq on $\text{Row}(C)$. By abuse of notation, a row type will denote also its class membership.

(iii) Let C be a set of type constants and \simeq a stable type theory on $\text{Row}(C)$. A *typing relation* is a binary endorelation on $\text{Row}(C)/\simeq$. We define four typing relations, denoted by $\wedge, \vee, \supset, \asymp$. \wedge (*strict coherence*) is symmetric and

antireflexive (equivalent types are not put in relation)

$$\frac{\sigma \frown \tau}{[\sigma_1, \dots, \sigma_n] \rightarrow \sigma \frown [\tau_1, \dots, \tau_m] \rightarrow \tau} \quad (a) \qquad \frac{\exists i \leq n, \exists j \leq m \text{ such that } \sigma_i \smile \tau_j}{[\sigma_1, \dots, \sigma_n] \rightarrow \sigma \frown [\tau_1, \dots, \tau_m] \rightarrow \tau} \quad (b)$$

$\sigma_i \smile \tau_j$ (*strict incoherence*) is defined as $\sigma_i \not\approx \tau_j$ and $\sigma_i \not\prec \tau_j$. \circ (*coherence*) is the reflexive closure (all equivalent types are put in relation) of \frown , finally \asymp (*incoherence*) is the reflexive closure of \smile .

If \mathbb{R} is either a typing relation or \simeq , then we write $\mathbb{R}\{\sigma_1, \dots, \sigma_n\}$ to denote that $\sigma_i \mathbb{R} \tau_j$, for $i \neq j$ ($1 \leq i, j \leq n$).

The previous definition deserves some comments.

We define row types only in order to provide a coarse syntax on which to define useful tools, as type theory and typing relations, that will be used in order to define types. A non constant row types has always the shape $[\sigma_1, \dots, \sigma_n] \rightarrow \sigma$, where $[\sigma_1, \dots, \sigma_n]$ is a sequence of row types, that can be empty and can contain repetitions. For instance, both $[] \rightarrow \sigma$ and $[\sigma, \tau, \sigma] \rightarrow \sigma$ are row types. A stable type theory imposes an equivalence on row types, in particular sequences on the left-side of an arrow are managed up to set-theoretical equivalence. So, for instance $\tau' \simeq \tau$ implies that $[\sigma, \tau, \sigma] \rightarrow \sigma \simeq [\tau', \sigma] \rightarrow \sigma$. We could have done a different choice, denoting directly the left hand of an arrow as a set, but we chosed to use sequences (denoted by square brackets) in place of sets (usually denoted by curly bracket) in order to emphasize the difference between the syntax of row types and the metalanguage.

Strict coherence will be used in type formation, since it formalizes what bits of type-information are consistent (in classical sense of domain theory⁴). We remark that strict coherence is given up to equivalence between types, thus if $\sigma \simeq \tau$ and $\tau \frown \pi$ then $\sigma \frown \pi$. Let us observe that, in Definition 3.1.(iii), rules (a) and (b) together require that, given two not constant row types, either their right hand are strictly consistent or they have at least two inconsistent elements in the left hand side. So, for instance $[] \rightarrow \sigma$ is incoherent with all not constant row types having σ in the right hand side.

Definition 3.2 (i) A *stable type system* ∇ is a triple $\langle C_\nabla, \simeq_\nabla, \frown_\nabla \rangle$, where C_∇ is a set of type constants, \simeq_∇ is a stable type theory on $Row(C_\nabla)$ and \frown_∇ is a strict coherence relation on $Row(C_\nabla)/\simeq_\nabla$.

(ii) A *lazy stable type system* ∇ is a quadruple $\langle C_\nabla, \simeq_\nabla, \frown_\nabla, \nu_\nabla \rangle$ which extends the stable type system $\langle C_\nabla, \simeq_\nabla, \frown_\nabla \rangle$ by selecting a special type constant $\nu_\nabla \in C_\nabla$ such that $\nu_\nabla \frown_\nabla \sigma$, for all $\sigma \in Row(C_\nabla) \setminus \{\nu_\nabla\}$. We will call ν_∇ the pivot of ∇ .

It should be clear that a stable type system can contain zero, one or many type constants that can be used as pivot. Hence, a stable type system can induce zero, one or many lazy type systems.

Given a stable type system (possibly lazy), the definition of row types can be refined in order to obtain *types*. Hence, a type assignment system can be designed by formalizing rules assigning such types to terms of λ -calculus.

⁴ Remember that for classical filter model a consistence relation is not necessary, since they live in the world of complete lattices [23] where all elements are consistent.

$$\boxed{
 \begin{array}{c}
 \frac{}{\mathbf{x} : \sigma \vdash_{\nabla} \mathbf{x} : \sigma} \text{ (var)} \quad \frac{B \vdash_{\nabla} \mathbf{M} : \tau \quad \sigma \simeq_{\nabla} \tau}{B \vdash_{\nabla} \mathbf{M} : \sigma} (\simeq) \\
 \\
 \frac{B \vdash_{\nabla} \mathbf{M} : [\sigma_1, \dots, \sigma_n] \rightarrow \sigma \quad (B_i \vdash_{\nabla} \mathbf{N} : \sigma_i)_{1 \leq i \leq n} \quad \star(B, B_1, \dots, B_n)}{(\bigcup_{1 \leq i \leq n} B_i) \cup B \vdash_{\nabla} \mathbf{M} \mathbf{N} : \sigma} (\rightarrow E) \\
 \\
 \frac{B \cup \{\mathbf{x} : \sigma_1, \dots, \mathbf{x} : \sigma_n\} \vdash_{\nabla} \mathbf{M} : \tau \quad \mathbf{x} \notin \text{dom}(B)}{B \vdash_{\nabla} \lambda \mathbf{x}. \mathbf{M} : [\sigma_1, \dots, \sigma_n] \rightarrow \tau} (\rightarrow I)
 \end{array}
 }$$

 Fig. 1. The Type Assignment Systems \vdash_{∇} .

Definition 3.3 Let ∇ be a stable type system (possibly lazy).

- (i) The *set of types* \mathbb{T}_{∇} is the subset of $\text{Row}(C_{\nabla})$ such that

$$\frac{p \in C_{\nabla}}{p \in \mathbb{T}_{\nabla}} \quad \frac{\tau, \tau_1, \dots, \tau_n \in \mathbb{T}_{\nabla} \quad \wedge_{\nabla} \{\tau_1, \dots, \tau_n\}}{[\tau_1, \dots, \tau_n] \rightarrow \tau \in \mathbb{T}_{\nabla}}$$

- (ii) A type assignment is a pair of the shape $\mathbf{x} : \sigma$, where \mathbf{x} is a variable and $\sigma \in \mathbb{T}_{\nabla}$. A ∇ -basis B is a finite set of type assignments such that, if $\mathbf{x} : \sigma_1, \dots, \mathbf{x} : \sigma_n \in B$ then $\wedge_{\nabla} \{\sigma_1, \dots, \sigma_n\}$. Let $\text{dom}(B) = \{\mathbf{x} \mid \exists \sigma. \mathbf{x} : \sigma \in B\}$. If $B = B' \cup \{\mathbf{x} : \sigma_1, \dots, \mathbf{x} : \sigma_n\}$ and $\mathbf{x} \notin \text{dom}(B')$ then $B(\mathbf{x}) = \{\sigma_1, \dots, \sigma_n\}$. Let B_i be a basis for $1 \leq i \leq n$; we write $\star(B_1, B_2, \dots, B_n)$ meaning that if $\mathbf{x} : \sigma \in B_i$ and $\mathbf{x} : \tau \in B_j$, for some $i \neq j$ ($1 \leq i, j \leq n$) then either $\sigma \wedge_{\nabla} \tau$ or $\sigma = \tau$.
- (iii) A *stable intersection type assignment system* \vdash_{∇} is a formal system proving statements of the shape:

$$B \vdash_{\nabla} \mathbf{M} : \sigma$$

where \mathbf{M} is a term, $\sigma \in \mathbb{T}_{\nabla}$ and B is a ∇ -basis.

The rules of the system are in Figure 1.

- (iv) Let ∇ be lazy and let ν_{∇} be its pivot. A *lazy stable intersection type assignment system* \vdash_{∇}^{ℓ} is a formal system proving statements of the shape:

$$B \vdash_{\nabla}^{\ell} \mathbf{M} : \sigma$$

where \mathbf{M} is a term, $\sigma \in \mathbb{T}_{\nabla}$ and B is a ∇ -basis.

The rules of the system are in Figure 2.

Differently from the syntax of row types, square brackets on the left-side of an arrow contain a sequence of types without multiple occurrences of equivalent elements. Note, for instance, that if $\sigma \simeq_{\nabla} \sigma'$ then $[\sigma, \sigma'] \rightarrow \tau \notin \mathbb{T}_{\nabla}$.

Note that a basis cannot assign equivalent types to the same variable. If B_i are bases, then it is easy to check that $\star(B_1, B_2, \dots, B_n)$ implies that $\bigcup_{1 \leq i \leq n} B_i$ is a well-defined basis. In particular, we note that this would be no longer true replacing $=$ by \simeq_{∇} in the definition of \star .

Both the type assignment systems are linear, in the sense that weakening does not hold. As we will see in a formal way, a basis contains the minimal information

$$\boxed{
 \begin{array}{c}
 \frac{}{\mathbf{x} : \sigma \vdash_{\nabla}^{\ell} \mathbf{x} : \sigma} \text{ (var)} \qquad \frac{B \vdash_{\nabla}^{\ell} \mathbf{M} : \tau \quad \sigma \simeq_{\nabla} \tau}{B \vdash_{\nabla}^{\ell} \mathbf{M} : \sigma} (\simeq) \\
 \\
 \frac{
 \begin{array}{c}
 B' \vdash_{\nabla}^{\ell} \mathbf{M} : \nu_{\nabla} \\
 B \vdash_{\nabla}^{\ell} \mathbf{M} : [\sigma_1, \dots, \sigma_n] \rightarrow \sigma \quad (B_i \vdash_{\nabla}^{\ell} \mathbf{N} : \sigma_i)_{1 \leq i \leq n} \quad \star(B, B', B_1, \dots, B_n)
 \end{array}
 }{
 (\bigcup_{1 \leq i \leq n} B_i) \cup B \cup B' \vdash_{\nabla}^{\ell} \mathbf{MN} : \sigma
 } (\rightarrow E) \\
 \\
 \frac{
 \begin{array}{c}
 B \cup \{\mathbf{x} : \sigma_1, \dots, \mathbf{x} : \sigma_n\} \vdash_{\nabla}^{\ell} \mathbf{M} : \tau \quad \mathbf{x} \notin \text{dom}(B) \\
 B \vdash_{\nabla}^{\ell} \lambda \mathbf{x}. \mathbf{M} : [\sigma_1, \dots, \sigma_n] \rightarrow \tau
 \end{array}
 }{
 B \vdash_{\nabla}^{\ell} \lambda \mathbf{x}. \mathbf{M} : [\sigma_1, \dots, \sigma_n] \rightarrow \tau
 } (\rightarrow I) \qquad \frac{}{\emptyset \vdash_{\nabla}^{\ell} \lambda \mathbf{x}. \mathbf{M} : \nu_{\nabla}} (\text{lazy})
 \end{array}
 }$$

 Fig. 2. The Type Assignment Systems \vdash_{∇}^{ℓ} .

necessary for the typing.

In rule $(\rightarrow E)$ all bases need to be strictly coherent to each other, and this is essential for preserving the correct syntax of types. The lazy system \vdash_{∇}^{ℓ} uses all the rules of \vdash_{∇} , but a modified $(\rightarrow E)$ rule, and a new rule (lazy) , the latter assigning ν_{∇} to all abstractions and the former asking that only terms for which the type ν_{∇} is derivable can be used in functional position. So ν_{∇} characterizes terms with a functional behaviour.

Notation 3.1 Let ∇ be a stable type system and let \mathbb{R} be a typing relation. $\{\sigma_1, \dots, \sigma_n\} \mathbb{R} \{\tau_1, \dots, \tau_m\}$ is an abbreviation for $\forall i \exists j \sigma_i \mathbb{R} \tau_j$, $\forall i \exists j \tau_i \mathbb{R} \sigma_j$. Moreover $B_1 \mathbb{R} B_2$ shortens $B_1(\mathbf{x}) \mathbb{R} B_2(\mathbf{x})$, for all \mathbf{x} .

Lastly, we use \vdash_{∇}^* to denote both type assignment systems.

Note that $B_1 \simeq_{\nabla} B_2$ implies that the cardinality of the set $B_1(\mathbf{x})$ is the same as the set $B_2(\mathbf{x})$, for all variables \mathbf{x} .

The legality condition given in the next definition characterizes (lazy) stable type systems that give rise to models of λ -calculus.

Definition 3.4 A (lazy) stable type system ∇ is *legal* whenever, if $[\sigma_1, \dots, \sigma_n] \rightarrow \sigma \in \mathbb{T}_{\nabla}$ and $[\tau_1, \dots, \tau_m] \rightarrow \tau \in \mathbb{T}_{\nabla}$ then both:

- (i) $[\sigma_1, \dots, \sigma_n] \rightarrow \sigma \simeq_{\nabla} [\tau_1, \dots, \tau_m] \rightarrow \tau$ implies both $\sigma \simeq \tau$ and $\{\sigma_1, \dots, \sigma_n\} \simeq_{\nabla} \{\tau_1, \dots, \tau_m\}$,
- (ii) $[\sigma_1, \dots, \sigma_n] \rightarrow \sigma \wedge_{\nabla} [\tau_1, \dots, \tau_m] \rightarrow \tau$ implies either $\sigma \wedge_{\nabla} \tau$ or $\exists i \leq n, \exists j \leq m$ such that $\sigma_i \vee_{\nabla} \tau_j$.

We remark that point (i) of legality corresponds to reversibility of the rule of Definition 3.1.(ii), while the point (ii) of legality corresponds to reversibility of the rules of Definition 3.1.(iii). In particular, if ∇ is legal, then both $[\tau_1, \dots, \tau_n] \rightarrow \tau, [\sigma_1, \dots, \sigma_m] \rightarrow \sigma \in \mathbb{T}_{\nabla}$ and $[\tau_1, \dots, \tau_n] \rightarrow \tau \simeq_{\nabla} [\sigma_1, \dots, \sigma_m] \rightarrow \sigma$ imply $n = m$.

Lemma 3.5 (Generation) Let ∇ be a legal stable type system.

- (i) $B \vdash_{\nabla}^* \mathbf{x} : \sigma$ implies $B = \{\mathbf{x} : \tau\}$ and $\sigma \simeq_{\nabla} \tau$.
- (ii) $B \vdash_{\nabla}^* \mathbf{M} : \sigma$ implies $\text{dom}(B) \subseteq FV(\mathbf{M})$.

- (iii) • $B \vdash_{\nabla} \lambda \mathbf{x}.M : \sigma$ implies $\sigma \simeq [\sigma_1, \dots, \sigma_n] \rightarrow \tau$;
 • $B \vdash_{\nabla}^{\ell} \lambda \mathbf{x}.M : \sigma$ implies either $\sigma = \nu_{\nabla}$ or $\sigma \simeq [\sigma_1, \dots, \sigma_n] \rightarrow \tau$.
- (iv) • $B \vdash_{\nabla} MN : \sigma$ implies $B' \vdash_{\nabla} M : [\sigma_1, \dots, \sigma_n] \rightarrow \sigma$, $B_i \vdash_{\nabla} N : \sigma_i$, where $B = (\bigcup_{1 \leq i \leq n} B_i) \cup B'$;
 • $B \vdash_{\nabla}^{\ell} MN : \sigma$ implies $B' \vdash_{\nabla}^{\ell} M : [\sigma_1, \dots, \sigma_n] \rightarrow \sigma$, $B'' \vdash_{\nabla}^{\ell} M : \nu_{\nabla}$ and $B_i \vdash_{\nabla}^{\ell} N : \sigma_i$, where $B = (\bigcup_{1 \leq i \leq n} B_i) \cup B' \cup B''$.
- (v) $B \vdash_{\nabla}^* M : \sigma$ and $B \simeq_{\nabla} B'$ imply $B' \vdash_{\nabla}^* M : \sigma$.
- (vi) $B \vdash_{\nabla}^* \lambda \mathbf{x}.M : [\sigma_1, \dots, \sigma_n] \rightarrow \tau$ if and only if $B \cup \{\mathbf{x} : \sigma_1, \dots, \mathbf{x} : \sigma_n\} \vdash_{\nabla}^* M : \tau$ and $\mathbf{x} \notin \text{dom}(B)$.

Proof. The first five points follow easily by induction on derivation.

Note that in point (ii), the inclusion can be strict either in case of an application MN , where M has type $[\] \rightarrow \sigma$, or in case rule *(lazy)* has been applied.

(vi) Let us consider first the system \vdash_{∇} . If the last applied rule is $(\rightarrow I)$, then the proof is immediate. Otherwise, the derivation proving $B \vdash_{\nabla} \lambda \mathbf{x}.M : [\sigma_1, \dots, \sigma_n] \rightarrow \tau$ ends in the following way:

$$\frac{\frac{B \cup \{\mathbf{x} : \pi_1, \dots, \mathbf{x} : \pi_n\} \vdash_{\nabla} M : \mu}{B \vdash_{\nabla} \lambda \mathbf{x}.M : [\pi_1, \dots, \pi_n] \rightarrow \mu} (\rightarrow I)}{B \vdash_{\nabla} \lambda \mathbf{x}.M : [\sigma_1, \dots, \sigma_n] \rightarrow \tau} (\simeq)$$

where $[\pi_1, \dots, \pi_n] \rightarrow \mu \simeq_{\nabla} [\sigma_1, \dots, \sigma_n] \rightarrow \tau$, by transitivity of \simeq_{∇} . Moreover, since the system is legal, $\{\pi_1, \dots, \pi_n\} \simeq_{\nabla} \{\sigma_1, \dots, \sigma_n\}$ and $\mu \simeq_{\nabla} \tau$. Then the proof follows by point (v) and by rule (\simeq) .

In case of system \vdash_{∇}^{ℓ} , if the derivation proving $B \vdash_{\nabla} \lambda \mathbf{x}.M : [\sigma_1, \dots, \sigma_n] \rightarrow \tau$ is as before, then the same proof applies. The derivation cannot end by an application of rule *(lazy)*, since by definition $\nu_{\nabla} \frown_{\nabla} \sigma$ (and thus $\nu_{\nabla} \not\approx_{\nabla} \sigma$), for all $\sigma \in \text{Row}(C_{\nabla}) \setminus \{\nu_{\nabla}\}$, so $\nu_{\nabla} \not\approx_{\nabla} [\sigma_1, \dots, \sigma_n] \rightarrow \tau$. \square

We remark that Lemma 3.5 holds only under the legality hypotheses. The property stated by following theorem, namely that in a derivation the basis collects the minimal information on the premises necessary for the typing, is crucial for building a model starting from a type assignment system.

Theorem 3.6 *Let ∇ be a legal stable type system.*

- (i) If $B_0 \vdash_{\nabla}^* M : \sigma_0$, $B_1 \vdash_{\nabla}^* M : \sigma_1$ and $B_0 \supset_{\nabla} B_1$ then $\sigma_0 \supset_{\nabla} \sigma_1$.
- (ii) If $B_0 \vdash_{\nabla}^* M : \sigma_0$, $B_1 \vdash_{\nabla}^* M : \sigma_1$, $\sigma_0 \simeq_{\nabla} \sigma_1$ and $B_0 \supset_{\nabla} B_1$ then $B_0 \simeq_{\nabla} B_1$.
- (iii) If $B_0 \vdash_{\nabla}^* M : \sigma$, $B_1 \vdash_{\nabla}^* M : \sigma$ and $B_0 \subseteq B_1$ then $B_0 = B_1$.

Proof. The proof of first two points is given by mutual induction on M .

- (i) If $M \equiv \mathbf{x}$ then the proof follows by Lemma 3.5.(i).

If $M \equiv PQ$ then the proof follows by Lemma 3.5.(iv) and by induction, taking into account the following obvious property:

$$\star(B_1, \dots, B_n) \text{ and } \forall \{p_1, \dots, p_k\} \subseteq \{1, \dots, n\} \text{ imply } \star(B_{p_1}, \dots, B_{p_k}) \quad (1)$$

Let $M \equiv \lambda \mathbf{x}.N$. If $\sigma_0 = \nu_{\nabla}$ or $\sigma_1 = \nu_{\nabla}$ then the proof follows definition of lazy stable type system, namely $\nu_{\nabla} \frown_{\nabla} \sigma$, for all $\sigma \in \text{Row}(C_{\nabla}) \setminus \{\nu_{\nabla}\}$. Let us

assume $\sigma_i \equiv [\tau_1^i, \dots, \tau_{n_i}^i] \rightarrow \tau^i$ where $n_i \geq 0$ ($0 \leq i \leq 1$). In case there are $\mu_0 \smile_{\nabla} \mu_1$ such that $\mu_i \in \{\tau_1^i, \dots, \tau_{n_i}^i\}$ then by rule (b) of Definition 3.1.(iii) the proof follows. Otherwise, assume that $B'_i = B_i \cup \{\mathbf{x} : \tau_1^i, \dots, \mathbf{x} : \tau_{n_i}^i\}$ and $B'_0 \smile_{\nabla} B'_1$. Since, by Lemma 3.5.(vi), $B'_i \vdash_{\nabla}^* \mathbf{M} : \tau^i$ and $\mathbf{x} \notin \text{dom}(B_i)$, $\tau^0 \smile_{\nabla} \tau^1$ by induction. If $\tau^0 \smile_{\nabla} \tau^1$ then the proof follows by rule (a) of Definition 3.1.(iii). The case $\tau^0 \simeq_{\nabla} \tau^1$ follows by mutual induction, since $B'_0 \simeq_{\nabla} B'_1$ implies $\{\mathbf{x} : \tau_1^0, \dots, \mathbf{x} : \tau_{n_0}^0\} \simeq_{\nabla} \{\mathbf{x} : \tau_1^1, \dots, \mathbf{x} : \tau_{n_0}^1\}$.

- (ii) If $\mathbf{M} \equiv \mathbf{x}$ then the proof follows by Lemma 3.5.(i). If $\mathbf{M} \equiv \text{PQ}$ then $B'_i \vdash_{\nabla}^* \text{P} : [\tau_1^i, \dots, \tau_{n_i}^i] \rightarrow \sigma_i$, $B'_j \vdash_{\nabla}^* \text{Q} : \tau_j^i$ (and $B''^i \vdash_{\nabla}^* \text{P} : \nu_{\nabla}$ if the system is lazy) where $B_i = (\bigcup_{1 \leq j \leq n} B_j^i) \cup B'_i \cup B''^i$, by Lemma 3.5.(iv). But $[\tau_1^0, \dots, \tau_{n_0}^0] \rightarrow \sigma_0 \smile_{\nabla} [\tau_1^1, \dots, \tau_{n_1}^1] \rightarrow \sigma_1$ by mutual induction and property stated in equation (1). By Definition 3.4.(ii) (legality), $[\tau_1^0, \dots, \tau_{n_0}^0] \rightarrow \sigma_0 \smile_{\nabla} [\tau_1^1, \dots, \tau_{n_1}^1] \rightarrow \sigma_1$ is not possible, since the hypothesis $\sigma_0 \simeq_{\nabla} \sigma_1$ and since $\tau_j^i \smile_{\nabla} \tau_{j'}^{i'}$ where $0 \leq i, i' \leq 1$, $1 \leq j \leq n_i$ and $1 \leq j' \leq n_{i'}$, by mutual induction. Hence, $[\tau_1^0, \dots, \tau_{n_0}^0] \rightarrow \sigma_0 \simeq_{\nabla} [\tau_1^1, \dots, \tau_{n_1}^1] \rightarrow \sigma_1$. Thus the proof follows by legality and induction.

Let $\mathbf{M} \equiv \lambda \mathbf{x}. \mathbf{N}$. If $\sigma_0 = \nu_{\nabla}$ or $\sigma_1 = \nu_{\nabla}$ then $\sigma_0 = \sigma_1$ by definition of lazy stable type system. Hence $B_0 = B_1 = \emptyset$, since there is a unique rule assigning to an abstraction a non-arrow type. Otherwise, let us assume $\sigma_i \equiv [\tau_1^i, \dots, \tau_{n_i}^i] \rightarrow \tau^i$ for $0 \leq i \leq 1$. Thus $B_i \cup \{\mathbf{x} : \tau_1^i, \dots, \mathbf{x} : \tau_{n_i}^i\} \vdash_{\nabla}^* \mathbf{M} : \tau^i$ where $n_i \geq 0$ by Lemma 3.5.(vi). The legality implies that $\tau^0 \simeq_{\nabla} \tau^1$ and $\{\tau_1^0, \dots, \tau_{n_0}^0\} \simeq_{\nabla} \{\tau_1^1, \dots, \tau_{n_1}^1\}$, so the proof follows by induction.

- (iii) Easy, in fact $B_0 \simeq_{\nabla} B_1$ by applying the point (ii) of this Theorem. Since $B_0 \subseteq B_1$ the proof is done. \square

Now we will prove that a legal stable type system induces a λ -model. We call this kind of model *clique model* for its relation with coherence spaces, which will be proved in Section 5. Indeed, Theorem 3.6 formalizes a crucial ingredients of such a correspondence.

Definition 3.7 Let ∇ be a legal stable type system.

- (i) A *typing clique* on ∇ is a set of types, pairwise coherent, closed under \simeq_{∇} . Typing cliques are ranged over by s, t . Let $\mathcal{S}(\nabla)$ be the set of typing cliques on ∇ .
- (ii) \circ_{∇} is a binary operation defined on $\mathcal{S}(\nabla)$ in the following way:
 $s_1 \circ_{\nabla} s_2 = \{\tau \mid [\sigma_1, \dots, \sigma_n] \rightarrow \tau \in s_1 \text{ and } \sigma_i \in s_2 (1 \leq i \leq n)\}$.
- (iii) Let ∇ be lazy. \circ_{∇}^{ℓ} is a binary operation defined on $\mathcal{S}(\nabla)$ in the following way:
 $s_1 \circ_{\nabla}^{\ell} s_2 = \{\tau \mid \nu_{\nabla} \in s_1, [\sigma_1, \dots, \sigma_n] \rightarrow \tau \in s_1 \text{ and } \sigma_i \in s_2 (1 \leq i \leq n)\}$

We use \circ_{∇}^* to denote both compositions. Typing cliques play in this paper a role similar to filters in filter models (see [9,18]).

Lemma 3.8 Let ∇ be a legal stable type system.

If $s_1, s_2 \in \mathcal{S}(\nabla)$ then both $s_1 \circ_{\nabla} s_2 \in \mathcal{S}(\nabla)$ and $s_1 \circ_{\nabla}^{\ell} s_2 \in \mathcal{S}(\nabla)$.

Proof. We check that the elements of $s_1 \circ_{\nabla} s_2$ are pairwise coherent. Let $\tau_1, \tau_2 \in$

$s_1 \circ_{\nabla} s_2$. Thus, there are $[\sigma_1^0, \dots, \sigma_{n_0}^0] \rightarrow \tau_1, [\sigma_1^1, \dots, \sigma_{n_1}^1] \rightarrow \tau_2 \in s_1, \sigma_i^0 \in s_2$ ($1 \leq i \leq n_0$) and $\sigma_i^1 \in s_2$ ($1 \leq i \leq n_1$). Since s_1 and s_2 are typing cliques (so their elements are pairwise coherent) $\tau_1 \circ_{\nabla} \tau_2$. The closure under equivalence is immediate.

To prove $s_1 \circ_{\nabla}^{\ell} s_2 \in \mathcal{S}(\nabla)$, just note that if $\nu_{\nabla} \notin s_1$ then $s_1 \circ_{\nabla}^{\ell} s_2 = \emptyset$. Otherwise the composition is as the previous one. \square

The interpretation function associates to a term all types that can be assigned to it. If ∇ be a legal stable type system and $\rho : Var \rightarrow \mathcal{S}(\nabla)$, then we define

$$\llbracket \mathbf{M} \rrbracket_{\rho}^{\nabla} = \{ \sigma \in \mathcal{T}_{\nabla} \mid \exists B. \forall \mathbf{x}. B(\mathbf{x}) \subseteq \rho(\mathbf{x}) \wedge B \vdash_{\nabla} \mathbf{M} : \sigma \}.$$

Moreover, if ∇ is lazy then

$$\llbracket \mathbf{M} \rrbracket_{\rho}^{\nabla \ell} = \{ \sigma \in \mathcal{T}_{\nabla} \mid \exists B. \forall \mathbf{x}. B(\mathbf{x}) \subseteq \rho(\mathbf{x}) \wedge B \vdash_{\nabla}^{\ell} \mathbf{M} : \sigma \}.$$

Observe that, by Theorem 3.6, the two interpretations map λ -terms into typing cliques.

Theorem 3.9 *Let ∇ be a legal stable type system.*

- (i) *If $\nabla = \langle C, \simeq_{\nabla}, \wedge_{\nabla} \rangle$ then $\mathcal{M}^{\nabla} = \langle \mathcal{S}(\nabla), \circ_{\nabla}, \llbracket \cdot \rrbracket^{\nabla} \rangle$ is a λ -model.*
- (ii) *If $\nabla = \langle C, \simeq_{\nabla}, \wedge_{\nabla}, \nu_{\nabla} \rangle$ then $\mathcal{M}^{\nabla \ell} = \langle \mathcal{S}(\nabla), \circ_{\nabla}, \llbracket \cdot \rrbracket^{\nabla \ell} \rangle$ is a λ -model.*

Proof. In both cases we need to check that the four conditions required to be a λ -model, given in Definition 2.2, are respected.

1. Easy $\{ \sigma \in \mathcal{T}_{\nabla} \mid \exists B. (\forall \mathbf{y}. B(\mathbf{y}) \subseteq \rho(\mathbf{y})) \wedge B \vdash_{\nabla}^* \mathbf{x} : \sigma \} = \rho(\mathbf{x})$, by Lemma 3.5.(i).
2. By Lemma 3.5.(iv), it is easy to check that

$$\begin{aligned} \llbracket \mathbf{MN} \rrbracket_{\rho}^{\nabla} &= \{ \sigma \in \mathcal{T}_{\nabla} \mid \exists B. (\forall \mathbf{y}. B(\mathbf{y}) \subseteq \rho(\mathbf{y})) \text{ such that } B \vdash_{\nabla} \mathbf{MN} : \sigma \} \\ &= \left\{ \sigma \in \mathcal{T}_{\nabla} \mid \exists B. (\forall \mathbf{y}. B(\mathbf{y}) \subseteq \rho(\mathbf{y})) \text{ such that } \begin{array}{l} B' \vdash_{\nabla} \mathbf{M} : [\sigma_1, \dots, \sigma_n] \rightarrow \sigma, \\ B_i \vdash_{\nabla} \mathbf{N} : \sigma_i, \\ B = (\bigcup_{1 \leq i \leq n} B_i) \cup B' \end{array} \right\} \\ &= \{ \sigma \in \mathcal{T}_{\nabla} \mid [\sigma_1, \dots, \sigma_n] \rightarrow \sigma \in \llbracket \mathbf{M} \rrbracket_{\rho}^{\nabla} \wedge \sigma_i \in \llbracket \mathbf{N} \rrbracket_{\rho}^{\nabla} \} = \llbracket \mathbf{M} \rrbracket_{\rho}^{\nabla} \circ_{\nabla} \llbracket \mathbf{N} \rrbracket_{\rho}^{\nabla} \end{aligned}$$

The lazy case is similar.

3. Let $s \in \mathcal{S}(\nabla)$. By Lemma 3.5.(vi), it is easy to check that

$$\begin{aligned}
 & \llbracket \lambda \mathbf{x}. \mathbf{M} \rrbracket_{\rho}^{\nabla} \circ_{\nabla} s = \{ \sigma \in \mathbb{T}_{\nabla} \mid \sigma_i \in s \text{ and } [\sigma_1, \dots, \sigma_n] \rightarrow \sigma \in \llbracket \lambda \mathbf{x}. \mathbf{M} \rrbracket_{\rho}^{\nabla} \} = \\
 & = \left\{ \sigma \in \mathbb{T}_{\nabla} \mid \exists B. (\forall y. B(y) \subseteq \rho(y)) \text{ s.t. } \begin{array}{l} \sigma_i \in s \text{ for all } i \text{ and} \\ B \vdash_{\nabla} \lambda \mathbf{x}. \mathbf{M} : [\sigma_1, \dots, \sigma_n] \rightarrow \sigma \end{array} \right\} \\
 & = \left\{ \sigma \in \mathbb{T}_{\nabla} \mid \exists B. (\forall y. B(y) \subseteq \rho(y)) \text{ s.t. } \begin{array}{l} \sigma_i \in s \text{ for all } i \text{ and } \mathbf{x} \notin \text{dom}(B) \\ \text{and } B \cup \{ \mathbf{x} : \sigma_1, \dots, \mathbf{x} : \sigma_n \} \vdash_{\nabla} \mathbf{M} : \sigma \end{array} \right\} \\
 & = \{ \sigma \in \mathbb{T}_{\nabla} \mid \exists B'. (\forall y. B'(y) \subseteq \rho[s/\mathbf{x}](y)) \text{ s.t. } B' \vdash_{\nabla} \mathbf{M} : \sigma \} = \llbracket \mathbf{M} \rrbracket_{\rho[s/\mathbf{x}]}^{\nabla}
 \end{aligned}$$

The lazy case is similar, since $\nu_{\nabla} \in \llbracket \lambda \mathbf{x}. \mathbf{M} \rrbracket_{\rho}^{\nabla \ell}$.

4. Let $s \in \mathcal{S}(\nabla)$ and let $\llbracket \mathbf{M} \rrbracket_{\rho[s/\mathbf{x}]}^{\nabla} = \llbracket \mathbf{M}' \rrbracket_{\rho'[s/y]}^{\nabla}$. Thus, for all $\sigma \in \mathbb{T}_{\nabla}$:

$\exists B. (\forall y. B(y) \subseteq \rho[s/\mathbf{x}](y)) \text{ s.t. } B \vdash_{\nabla} \mathbf{M} : \sigma$ iff $\exists B'. (\forall y. B'(y) \subseteq \rho'[s/y](y)) \text{ s.t. } B' \vdash_{\nabla} \mathbf{M}' : \sigma$.

$B \vdash_{\nabla} \lambda \mathbf{x}. \mathbf{M} : [\sigma_1, \dots, \sigma_n] \rightarrow \tau$ if and only if $B \cup \{ \mathbf{x} : \sigma_1, \dots, \mathbf{x} : \sigma_n \} \vdash_{\nabla} \mathbf{M} : \tau$ and $\mathbf{x} \notin \text{dom}(B)$ by Lemma 3.5.(vi). So $\exists B'. (\forall y. B'(y) \subseteq \rho'[B(\mathbf{x})/\mathbf{x}](y))$ such that $B' \vdash_{\nabla} \mathbf{M}' : \sigma$. Likewise $\exists B^*. (\forall y. B^*(y) \subseteq \rho[B'(\mathbf{x})/\mathbf{x}](y))$ such that $B^* \vdash_{\nabla} \mathbf{M} : \sigma$ therefore $B(\mathbf{x}) \subseteq B'(\mathbf{x}) \subseteq B^*(\mathbf{x})$. Hence $B(\mathbf{x}) = B'(\mathbf{x})$, since $B(\mathbf{x}) = B^*(\mathbf{x})$ by Theorem 3.6.(iii). The proof follows by Lemma 3.5.(vi). The lazy case is similar by Definition 3.7.(iii), since $\nu_{\nabla} \in \llbracket \lambda \mathbf{x}. \mathbf{M} \rrbracket_{\rho}^{\nabla \ell}$. □

In such a way we defined two meta-models of λ -calculus, which can become models by choosing particular legal stable type systems. Some instances of such meta-models has been studied in [17] and in [6].

Definition 3.10

- (i) $\mathcal{M}^{\nabla} = \langle \mathcal{S}(\nabla), \circ_{\nabla}, \llbracket \cdot \rrbracket^{\nabla} \rangle$ is the *clique model* induced by the legal stable type system ∇ .
- (ii) $\mathcal{M}^{\nabla \ell} = \langle \mathcal{S}(\nabla), \circ_{\nabla}, \llbracket \cdot \rrbracket^{\nabla \ell} \rangle$ is the *lazy clique model* induced by the lazy legal stable type system ∇ .

The two meta-models give a non empty interpretation to two interesting classes of terms.

Theorem 3.11 *Let ∇ be a legal stable type system.*

- (i) *If \mathbf{M} has head normal form then there are B, σ such that $B \vdash_{\nabla} \mathbf{M} : \sigma$.*
- (ii) *If \mathbf{M} has weak head normal form then there are B, σ such that $B \vdash_{\nabla}^{\ell} \mathbf{M} : \sigma$.*

Proof. (i) The type assignment is closed under $=_{\beta}$, since \vdash_{∇} induces a λ -model. So we need to prove just that every term in head normal form can be typed. Let \mathbf{M} be $\lambda \mathbf{x}_1 \dots \lambda \mathbf{x}_m. \mathbf{x} \mathbf{M}_1 \dots \mathbf{M}_n$ and let $B = \{ \mathbf{x} : \underbrace{[] \rightarrow [] \rightarrow \dots []}_{n} \rightarrow \sigma \}$, for a given σ . Then $B \vdash_{\nabla} \mathbf{x} \mathbf{M}_1 \dots \mathbf{M}_n : \sigma$, by repeatedly applying (\rightarrow E) rule.

If $\mathbf{x} \notin \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ then $B \vdash_{\nabla} \mathbf{M} : \overbrace{[] \rightarrow [] \rightarrow \dots []}^m \rightarrow \sigma$ by m applications of rule (\rightarrow I). Otherwise, if $\mathbf{x} = \mathbf{x}_h$ then

$$B \vdash_{\nabla} \mathbf{M} : \overbrace{[] \rightarrow [] \rightarrow \dots []}^{h-1} \rightarrow \left(\overbrace{[] \rightarrow [] \rightarrow \dots []}^m \rightarrow \sigma \right) \rightarrow \overbrace{[] \rightarrow [] \rightarrow \dots []}^{m-h} \rightarrow \sigma.$$

(ii) The case of an abstraction is immediate, by rule (*lazy*). Otherwise, let \mathbf{M} be

$$\underbrace{\mathbf{xM}_1 \dots \mathbf{M}_n}_{n}, B_i = \{ \mathbf{x} : \overbrace{[] \rightarrow [] \rightarrow \dots []}^i \rightarrow \nu_{\nabla} \} \ (1 \leq i \leq n-1), \text{ and } B = \{ \mathbf{x} : [] \rightarrow [] \rightarrow \dots [] \rightarrow \sigma \}, \text{ for a given } \sigma. \text{ By rule } (\rightarrow\text{E}), \bigcup_{j < i} B_j \vdash_{\nabla}^{\ell} \mathbf{xM}_1 \dots \mathbf{M}_i : \nu_{\nabla} \text{ and}$$

$$B \cup \bigcup_{j < i} B_j \vdash_{\nabla}^{\ell} \mathbf{xM}_1 \dots \mathbf{M}_i : \underbrace{[] \rightarrow [] \rightarrow \dots []}_{n-i} \rightarrow \sigma. \text{ So } B \cup \bigcup_{j < n} B_j \vdash_{\nabla}^{\ell} \mathbf{xM}_1 \dots \mathbf{M}_n : \sigma,$$

since it is easy to check that $\star(B, B_1, \dots, B_{n-1})$.

□

Theorem 3.11 is a necessary ingredient in order to prove that a clique model is adequate with respect to some well-studied operational semantics of λ -calculus (see Property 10.1.15 in page 117 of [23]).

4 Coherence Spaces

Coherence spaces are a simple framework for Berry's stable functions [7], developed by Girard [14]; in this Section their basic definitions and properties are stated. Proof details can be found in [15].

First, some basic definitions are given. A *partial order* or *poset* is a pair $\langle D, \sqsubseteq \rangle$ where D is a set and \sqsubseteq is an order relation, often noted simply as D . An element of D is *bottom* and denoted \perp if and only if $\perp \sqsubseteq d$ for each $d \in D$. A nonempty subset X of D is *directed* if $\forall x, x' \in X \exists x'' \in X$ such that $x \sqsubseteq x''$ and $x' \sqsubseteq x''$, namely for each pair of elements of X there is an upper bound in X . A *cpo* is a poset D with bottom $\perp \in D$ such that if $X \subseteq D$ is directed then there is $\sqcup X \in D$ which is the least upper bound of X . Let A, B be cpos; a function $f : A \rightarrow B$ is *monotone* if and only if $\forall x, x' \in A$ if $x \sqsubseteq_A x'$ then $f(x) \sqsubseteq_B f(x')$.

Definition 4.1 A *coherence space* X is a pair $\langle |X|, \circ_X \rangle$ where $|X|$ is a set called the *web*, its elements are called *tokens* and \circ_X is called the *coherence relation* on X .

\circ_X is a binary, reflexive and symmetric relation between tokens. The set of *cliques* of X is $\mathcal{Cl}(X) = \{x \subseteq |X| \mid \forall a, b \in x, a \circ_X b\}$; moreover, $\mathcal{Cl}_{fin}(X)$ denotes the set of finite cliques of $\mathcal{Cl}(X)$.

The strict incoherence \smile_X is the complementary relation of \circ_X ; the incoherence \asymp_X is the union of relations \smile_X and $=$; the strict coherence \frown_X is the complementary relation of \asymp_X .

If X is a coherence space then $\mathcal{Cl}(X)$ is a poset with respect to the relation \subseteq .

Lemma 4.2 *Let X be a coherence space.*

- (i) $\emptyset \in \mathcal{Cl}(X)$.

- (ii) $\{a\} \in \mathcal{Cl}(X)$, for each $a \in |X|$.
- (iii) If $y \subseteq x$ and $x \in \mathcal{Cl}(X)$ then $y \in \mathcal{Cl}(X)$.
- (iv) If $D \subseteq \mathcal{Cl}(X)$ is directed then $\cup D \in \mathcal{Cl}(X)$.

Hence, cliques of a coherence space with set-inclusion form a cpo.
 Let x, x' be sets; $x \subseteq_{fin} x'$ means that $x \subseteq x'$ and x is finite.

Definition 4.3 Let X and Y be coherence spaces and $f : \mathcal{Cl}(X) \longrightarrow \mathcal{Cl}(Y)$ be a monotone function.

- f is *continuous* whenever $\forall x \in \mathcal{Cl}(X) \forall b \in f(x) \exists x_0 \subseteq_{fin} x$ such that $b \in f(x_0)$.
- f is *stable* whenever $\forall x \in \mathcal{Cl}(X) \forall b \in f(x) \exists x_0 \subseteq_{fin} x$ such that $b \in f(x_0)$ and $\forall x' \subseteq x$, if $b \in f(x')$ then $x_0 \subseteq x'$.

Continuity asks for the existence of a finite amount of input for which some amount of output is produced, while stability asks for a minimum finite amount input for which some amount of output is produced. Equivalent formulations of continuity and stability are formalized in the following Lemmas.

- Lemma 4.4** (i) Let X and Y be coherence spaces and $f : \mathcal{Cl}(X) \longrightarrow \mathcal{Cl}(Y)$ be a monotone function. Then f is continuous if and only if $f(\cup D) = \cup\{f(x) \mid x \in D\}$, for each $D \subseteq \mathcal{Cl}(X)$ directed.
- (ii) Let X and Y be coherence spaces and $f : \mathcal{Cl}(X) \longrightarrow \mathcal{Cl}(Y)$ be a continuous function. Then f is stable if and only if $\forall x, x' \in \mathcal{Cl}(X)$, $x \cup x' \in \mathcal{Cl}(X)$ implies $f(x \cap x') = f(x) \cap f(x')$.

Stable functions can be represented as cliques of a coherence space..

Definition 4.5 Let X and Y be coherence spaces.

The *trace* $\text{tr}(f)$ of the stable function $f : \mathcal{Cl}(X) \longrightarrow \mathcal{Cl}(Y)$ is the set of pairs $(x_0, b) \in \mathcal{Cl}_{fin}(X) \times |Y|$ such that $b \in f(x_0)$ and $\forall x \subseteq x_0$, $b \in f(x)$ implies $x = x_0$.

Definition 4.6 Let X and Y be coherence spaces.

$X \Rightarrow Y$ is the coherence space having $|X \Rightarrow Y| = \mathcal{Cl}_{fin}(X) \times |Y|$ as web, while if $(x_0, b_0), (x_1, b_1) \in |X \Rightarrow Y|$, then $(x_0, b_0) \supset_{X \Rightarrow Y} (x_1, b_1)$ under the following conditions:

- (i) $x_0 \cup x_1 \in \mathcal{Cl}(X)$ implies $b_0 \supset_Y b_1$;
- (ii) $x_0 \cup x_1 \in \mathcal{Cl}(X)$ and $b_0 = b_1$ imply $x_0 = x_1$.

The previous definition can be reformulated in term of strict coherence as follows.

Proposition 4.7 $(x_0, b_0) \frown_{X \Rightarrow Y} (x_1, b_1)$ iff $x_0 \cup x_1 \in \mathcal{Cl}(X)$ implies $b_0 \frown_Y b_1$.

The bridge between stable functions and cliques follows.

Lemma 4.8 If $f : \mathcal{Cl}(X) \longrightarrow \mathcal{Cl}(Y)$ is a stable function then $\text{tr}(f) \in \mathcal{Cl}(X \Rightarrow Y)$.

Let X, Y be coherence spaces and $t \in \mathcal{Cl}(X \Rightarrow Y)$ and $x \in \mathcal{Cl}(X)$. Let us define $\mathcal{F}(t) : \mathcal{Cl}(X) \longrightarrow \mathcal{Cl}(Y)$ be the function such that

$$\mathcal{F}(t)(x) = \{b \in |Y| \mid \exists x_0 \in \mathcal{Cl}(X) \ (x_0, b) \in t \wedge x_0 \subseteq x\}.$$

Lemma 4.9 *If $t \in \mathcal{Cl}(X \Rightarrow Y)$ then $\mathcal{F}(t) : \mathcal{Cl}(X) \rightarrow \mathcal{Cl}(Y)$ is a stable function.*

Coherence spaces and stable functions form a cartesian closed category, which will be denoted with \mathcal{Coh} . It is a full subcategory of the categories of qualitative domains and dI-domains endowed with stable functions. All these categories contain objects and morphisms in the range of the standard interpretation of PCF, so without ambiguity they will be called stable models.

We conclude with an interesting sub-class of stable function, given by the following definition

Definition 4.10 A stable function $f : \mathcal{Cl}(X) \rightarrow \mathcal{Cl}(Y)$ is *linear* whenever, $f(\emptyset) = \emptyset$ and for all $x_1, x_2 \in \mathcal{Cl}(X)$ such that $x_1 \cup x_2 \in \mathcal{Cl}(X)$ we have that $f(x_1 \cup x_2) = f(x_1) \cup f(x_2)$.

It is easy to see that if $f : \mathcal{Cl}(X) \rightarrow \mathcal{Cl}(Y)$ is a linear function, then its trace $\text{tr}(f)$ can be expressed as a subset of $|X| \times |Y|$.

5 Stable λ -models

Let X be a reflexive object in the category \mathcal{Coh} , namely a coherence space X such that $X \Rightarrow X$ is a retract of X (noted $X \triangleright X \Rightarrow X$) through a retraction pair (ϕ, ψ) , where ϕ and ψ are two stable functions such that $\phi : \mathcal{Cl}(X) \rightarrow \mathcal{Cl}(X \Rightarrow X)$, $\psi : \mathcal{Cl}(X \Rightarrow X) \rightarrow \mathcal{Cl}(X)$ and $\phi \cdot \psi = \text{id}_{X \Rightarrow X}$. We call ϕ an *immersion morphism* and ψ a *projection morphism*. The following categorical diagram summarizes the previous notions.

$$\begin{array}{ccc}
 & \xrightarrow{\phi} & \\
 X & \xrightarrow{\quad \triangleright \quad} & X \Rightarrow X \\
 & \xleftarrow{\psi} & \\
 & \text{id} &
 \end{array}$$

A retraction through (ϕ, ψ) on a coherence space X gives rise to the model for λ -calculus $\mathcal{M} = \langle \mathcal{Cl}(X), \circ, \llbracket \cdot \rrbracket \rangle$, where:

- $x \circ y = \mathcal{F}(\phi(x))(y)$
- $\llbracket \cdot \rrbracket : \Lambda \rightarrow \text{Env} \rightarrow \mathcal{Cl}(X)$ is defined as:
 - $\llbracket \mathbf{x} \rrbracket_\rho = \rho(\mathbf{x})$;
 - $\llbracket \lambda \mathbf{x}. \mathbf{M} \rrbracket_\rho = \psi(\text{tr}(\llbracket \mathbf{M} \rrbracket_{\rho[y/\mathbf{x}]})$
 - $\llbracket \mathbf{MN} \rrbracket_\rho = \llbracket \mathbf{M} \rrbracket_\rho \circ \llbracket \mathbf{N} \rrbracket_\rho$;

where $\text{Env} = \{\rho \mid \rho : \text{Var} \rightarrow \mathcal{Cl}(X)\}$ and $\llbracket \cdot \rrbracket$ denotes the meta-theoretic abstraction.

Here we will define two particular classes of such models, putting some restrictions on the retraction pair. If $X = \langle |X|, \circ_X \rangle$ and $Y = \langle |Y|, \circ_Y \rangle$ are two coherence spaces, then a function $i : |X| \rightarrow |Y|$ is *iso-coherent* when $a \circ_X b$ if and only if $i(a) \circ_Y i(b)$, for all $a, b \in |X|$. Note that an iso-coherent function is a map from tokens to tokens.

Definition 5.1 Let X be a coherence space and let $i : |X \Rightarrow X| \rightarrow |X|$ be a total injective iso-coherent function. The pair $\langle X, i \rangle$ is a *coherence lambda-structure*.

- (i) A *linear* λ -model is the λ -model \mathcal{M} induced by the retraction pair (ϕ_i, ψ_i) defined as

$$\phi_i(x) = \{a \mid i(a) \in x\} \quad \psi_i(y) = \{i(a) \mid a \in y\}$$

- (ii) Let X be such that $|X|$ contains a distinguished token j such that it is coherent with all other tokens and $j \notin i(|X \Rightarrow X|)$. A *lazy* λ -model is the λ -model \mathcal{M}^j induced by the retraction pair (ϕ_i^j, ψ_i^j) defined as

$$\phi_i^j(x) = \begin{cases} \{a \mid i(a) \in x\} & \text{if } j \in x \\ \emptyset & \text{otherwise} \end{cases} \quad \psi_i^j(y) = \{i(a) \mid a \in y\} \cup \{j\}$$

It is routine to check that linear λ -models and lazy λ -models defined in the previous definition are well-given λ -model in the sense of Definition 2.2, since ϕ_i, ψ_i, ϕ_i^j and ψ_i^j are stable functions and both (ϕ_i, ψ_i) and (ϕ_i^j, ψ_i^j) are retraction pairs. The class of linear λ -models (first defined in [13]) collects all models in the category *Coh* whose immersion-projection morphisms (ϕ, ψ) are linear functions, hence the adjective “linear”. The class of lazy λ -models collects instead a sub-class of those models whose projection morphism ψ is not strict, i.e. $\psi(\emptyset) \neq \emptyset$.

Let ∇ be a legal stable type system. In the following of this section, we denote respectively by $[\tau]_{\nabla}$ and $\mathbb{T}_{\nabla}/\simeq_{\nabla}$ the equivalence class $\{\sigma \in \mathbb{T}_{\nabla} \mid \sigma \simeq_{\nabla} \tau\}$ and the whole set of such classes, i.e., $\{[\tau]_{\nabla} \mid \tau \in \mathbb{T}_{\nabla}\}$. The next lemma proves that types are names for tokens. The type theory formalizes the homonymy under which tokens are represented. Yet, an arrow is a special name denoting that the subjecting token is in the image of the projection of an oportune retraction.

Lemma 5.2 *If ∇ be a legal stable type system then $X_{\nabla} = \langle \mathbb{T}_{\nabla}/\simeq_{\nabla}, \circ_{\nabla} \rangle$ is a coherence space.*

Proof. Easy, since the relation \circ_{∇} is reflexive and symmetric by construction. \square

So, by Definition 4.6, $X_{\nabla} \Rightarrow X_{\nabla} = \langle |X_{\nabla} \Rightarrow X_{\nabla}|, \circ_{X_{\nabla} \Rightarrow X_{\nabla}} \rangle$ is the coherence space, where:

- (i) $|X_{\nabla} \Rightarrow X_{\nabla}| = \left\{ \left(\{[\sigma_1]_{\nabla}, \dots, [\sigma_n]_{\nabla}\}, [\tau]_{\nabla} \right) \mid \circ_{\nabla} \{\sigma_1, \dots, \sigma_n\} \right\}$,
- (ii) $(\{[\sigma_1]_{\nabla}, \dots, [\sigma_n]_{\nabla}\}, [\sigma]_{\nabla}) \circ_{X_{\nabla} \Rightarrow X_{\nabla}} (\{[\tau_1]_{\nabla}, \dots, [\tau_m]_{\nabla}\}, [\tau]_{\nabla})$ whenever,
if $\circ_{\nabla} \{[\sigma_1]_{\nabla}, \dots, [\sigma_n]_{\nabla}, [\tau_1]_{\nabla}, \dots, [\tau_m]_{\nabla}\}$ then $[\sigma]_{\nabla} \circ_{\nabla} [\tau]_{\nabla}$ and $[\sigma]_{\nabla} = [\tau]_{\nabla}$
implies $\{[\sigma_1]_{\nabla}, \dots, [\sigma_n]_{\nabla}\} = \{[\tau_1]_{\nabla}, \dots, [\tau_m]_{\nabla}\}$.

Definition 5.3 Let ∇ be a legal stable type system. The *legal-structure* induced by ∇ is the pair $\langle X_{\nabla}, i_{\nabla} \rangle$ where $X_{\nabla} = \langle \mathbb{T}_{\nabla}/\simeq_{\nabla}, \circ_{\nabla} \rangle$ and $i_{\nabla} : |X_{\nabla} \Rightarrow X_{\nabla}| \rightarrow |X_{\nabla}|$ is the map such that:

$$i_{\nabla} \left(\{[\sigma_1]_{\nabla}, \dots, [\sigma_n]_{\nabla}\}, [\tau]_{\nabla} \right) = [[\sigma_1, \dots, \sigma_n] \rightarrow \tau]_{\nabla}.$$

Note that without denote sets by sequences of their elements without repetitions.

We can show that legal-structures induce coherence lambda-structures.

Lemma 5.4 *Let $\langle X_\nabla, i_\nabla \rangle$ be a legal-structure.*

- (i) $\langle X_\nabla, i_\nabla \rangle$ is a coherence λ -structure, so it induces a linear λ -model.
- (ii) If ∇ is lazy then $\langle X_\nabla, i_\nabla \rangle$ induces also a lazy λ -model by putting $j = [\nu_\nabla]_{\underline{\nabla}}$.

Proof. (i) Let $x = \{[\sigma_1]_{\underline{\nabla}}, \dots, [\sigma_n]_{\underline{\nabla}}\}$, $y = \{[\tau_1]_{\underline{\nabla}}, \dots, [\tau_m]_{\underline{\nabla}}\}$, $a = [\sigma]_{\underline{\nabla}}$ and $b = [\tau]_{\underline{\nabla}}$. We want to check that i_∇ is injective. First, suppose $i_\nabla((x, a)) = i_\nabla((y, b))$. By definition of i_∇ , this means that $[\sigma_1, \dots, \sigma_n] \rightarrow \sigma \simeq_\nabla [\tau_1, \dots, \tau_m] \rightarrow \tau$. Since the type system is legal then $\{\sigma_1, \dots, \sigma_n\} \simeq_\nabla \{\tau_1, \dots, \tau_m\}$, $m = n$ and $\sigma \simeq_\nabla \tau$. Thus $(x, a) = (y, b)$. Second, it is easy to check that $i_\nabla((x, a)) \frown_\nabla i_\nabla((y, b))$ if and only if $(a, x) \frown_{X_\nabla \Rightarrow X_\nabla} (y, b)$ by Definition 3.1.(iii).

- (ii) The proof follows easily, since $j = [\nu_\nabla]_{\underline{\nabla}}$ trivially enjoys the required coherence constraints. □

We are now able to state a soundness result proving that all our clique models correspond to linear and lazy models. Let us say that two λ -models $\mathcal{M} = \langle \mathbb{D}, \circ, [\cdot]_{\underline{\nabla}}^{\mathcal{M}} \rangle$ and $\mathcal{M}' = \langle \mathbb{D}', \circ', [\cdot]_{\underline{\nabla}}^{\mathcal{M}'} \rangle$ coincide if \mathbb{D} and \mathbb{D}' are isomorphic through a bijection $h : \mathbb{D}' \rightarrow \mathbb{D}$, and $[\mathbb{M}]_{\rho}^{\mathcal{M}} = h([\mathbb{M}]_{\rho'}^{\mathcal{M}'})$, where $\rho(\mathbf{x}) = h(\rho'(\mathbf{x}))$.

Theorem 5.5 (Soundness) *Let ∇ be a legal stable type system.*

- (i) The clique model \mathcal{M}^∇ and the linear λ -model induced by $\langle X_\nabla, i_\nabla \rangle$ coincide.
- (ii) If ∇ is lazy with pivot ν_∇ then the lazy clique model \mathcal{M}^∇ and the lazy λ -model induced by $\langle X_\nabla, i_\nabla \rangle$, with $j = [\nu_\nabla]_{\underline{\nabla}}$, coincide.

Proof. (i) Let us recall that the clique model induced by ∇ is $\mathcal{M}^\nabla = \langle \mathcal{S}(\nabla), \circ_\nabla, [\cdot]_{\underline{\nabla}}^\nabla \rangle$. The model induced by $\langle X_\nabla, i_\nabla \rangle$, is $\mathcal{X}^\nabla = \langle \mathcal{Cl}(X_\nabla), \circ, [\cdot]_{\underline{\nabla}}^{\mathcal{X}^\nabla} \rangle$, where $x \circ y = \mathcal{F}(\phi_{i_\nabla}(x))(y)$, and the interpretation function is defined as at the beginning of this section. The isomorphism between $\mathcal{S}(\nabla)$ and $\mathcal{Cl}(X_\nabla)$ follows from the definition of X_∇ , namely, for each $s \in \mathcal{S}(\nabla)$ the corresponding element of $\mathcal{Cl}(X_\nabla)$ is $\{[\tau]_{\underline{\nabla}} \mid \tau \in s\}$. Then it is boring but easy to check, by cases, that $[\mathbb{M}]_{\rho}^{\mathcal{X}^\nabla} = [\mathbb{M}]_{\rho^*}^{\nabla} / \simeq_\nabla$ where $\rho^*(\mathbf{x}) = \bigcup_{[\tau]_{\underline{\nabla}} \in \rho(\mathbf{x})} [\tau]_{\underline{\nabla}}$, for all \mathbb{M} .

- (ii) The proof is similar to the previous point. □

In order to prove completeness we need to recall some basic property of coherence spaces.

Proposition 5.6 *Let X and Y be object of Coh , \mathcal{I} be the set of all (possible) isomorphisms between X and Y and \mathcal{T} be the set of (possible) iso-coherent bijective maps between $|X|$ and $|Y|$. A biunivocal correspondence can be established between \mathcal{I} and \mathcal{T} .*

Proof. Isomorphisms of Coh between X and Y are all and only bijective linear functions [14] whose trace univocally determines an iso-coherent bijective maps between $|X|$ and $|Y|$. For the other direction, an iso-coherent bijective map between $|X|$ and

$|Y|$ defines univocally a trace of a bijective linear function, i.e. an isomorphism (the hypothesis of iso-coherence is crucial in order to have a well-defined linear function). So there exists a biunivocal correspondence between \mathcal{I} and \mathcal{T} . \square

Now we will prove that, given a (lazy) linear λ -model induced by a coherence space we can define a type system inducing a (lazy) clique model coinciding with it. Then the completeness follows.

Lemma 5.7 *Let $\langle X, i \rangle$ be a coherence lambda-structure.*

- (i) *There is a legal stable type system ∇ inducing the legal-structure $\langle X_\nabla, i_\nabla \rangle$ and there are two iso-coherent bijections $(\cdot)^b : |X| \longrightarrow |X_\nabla|$ and $(\cdot)^\# : |X \Rightarrow X| \longrightarrow |X_\nabla \Rightarrow X_\nabla|$ such that the following diagram commutes*

$$\begin{array}{ccc} |X \Rightarrow X| & \xrightarrow{i} & |X| \\ (\cdot)^\# \downarrow & & \downarrow (\cdot)^b \\ |X_\nabla \Rightarrow X_\nabla| & \xrightarrow{i_\nabla} & |X_\nabla| \end{array}$$

- (ii) *Let X be such that $|X|$ contains a distinguished token j coherent with all its other tokens and $j \notin i(|X \Rightarrow X|)$. Then there is a legal lazy stable type system ∇ such that the previous diagram commutes.*

Proof. Let C_∇ be a set in bijection with the set $|X|$ through the map $(\cdot)^b : |X| \rightarrow C_\nabla$. Let \simeq_∇ be the least stable type theory on $\text{Row}(C_\nabla)$ such that, if $(x, a) \in |X \Rightarrow X|$, $x = \{a_1, \dots, a_n\}$ and $i((x, a)) = b \in |X|$ then $[a_1^b, \dots, a_n^b] \rightarrow a^b \simeq_\nabla (b)^b$. Since i is injective, $(\cdot)^b$ induces an injection from $|X|$ to $\text{Row}(C_\nabla)/\simeq_\nabla$, i.e. if $a, b \in |X|$ and $a^b \simeq_\nabla b^b$ then $a = b$. Let \wedge_∇ be the least binary relation on $\text{Row}(C_\nabla)/\simeq_\nabla$ satisfying Definition 3.1.(iii) and such that, if $a \wedge_X b$ then $[(a)^b]_{\simeq_\nabla} \wedge_\nabla [(b)^b]_{\simeq_\nabla}$, for all $a, b \in |X|$.

We will prove that $\nabla = \langle C_\nabla, \wedge_\nabla, \simeq_\nabla \rangle$ is the desired stable type system .

We already proved that $(\cdot)^b$ induces a bijection between $|X|$ and $\text{T}_\nabla/\simeq_\nabla$. Injectivity follows since $\text{T}_\nabla/\simeq_\nabla \subseteq \text{Row}(C_\nabla)/\simeq_\nabla$. We can easily prove by induction that if $\sigma \in \text{T}_\nabla$ then $\sigma \simeq_\nabla a^b$ for a $a \in |X|$. Thus surjectivity follows.

We show that the type system so obtained is legal. We begin by the point (i) of Definition 3.4. Suppose that $\{\sigma_1, \dots, \sigma_n\} \not\leq_\nabla \{\tau_1, \dots, \tau_m\}$ or $\sigma \not\leq_\nabla \tau$. Let x, y, a, b such that $x = \{a_1, \dots, a_n\}$ with $(a_j)^b \simeq_\nabla \sigma_j$ for all $j \in [1, n]$, $y = \{b_1, \dots, b_m\}$ with $(b_k)^b \simeq_\nabla \tau_k$ for all $k \in [1, m]$, $(a)^b \simeq_\nabla \sigma$, $(b)^b \simeq_\nabla \tau$. Since $(\cdot)^b$ induces a bijection between $|X|$ and $\text{T}_\nabla/\simeq_\nabla$, either $\sigma \not\leq_\nabla \tau$ implies $a \neq b$ or $\{\sigma_1, \dots, \sigma_n\} \not\leq_\nabla \{\tau_1, \dots, \tau_m\}$ implies that there is $a_j \neq b_k$. Then (x, a) and (y, b) are two different tokens of $|X \Rightarrow X|$. Since $i : |X \Rightarrow X| \rightarrow |X|$ is an injective function, we conclude $[\sigma_1, \dots, \sigma_n] \rightarrow \sigma \not\leq_\nabla [\tau_1, \dots, \tau_m] \rightarrow \tau$ as required. To prove the remaining point, we just observe that $(x, a) \wedge_{X \Rightarrow X} (y, b)$ if and only if either $a \wedge_X b$ or there exist $c_1 \in x, c_2 \in y$ such that $c_1 \smile_X c_2$. Thus point (ii) can be proved by using a reasoning similar to that for point (i).

Legality implies that $X_\nabla = \langle \text{T}_\nabla/\simeq_\nabla, \circ_\nabla \rangle$ is a coherence space. Hence, the map $(\cdot)^\# : |X \Rightarrow X| \longrightarrow |X_\nabla \Rightarrow X_\nabla|$ is induced by the bijection between $|X|$ and $|X_\nabla|$. If $(\{a_1, \dots, a_n\}, b) \in |X \Rightarrow X|$, then $\wedge_\nabla \{[(a_1)^b]_{\simeq_\nabla}, \dots, [(a_n)^b]_{\simeq_\nabla}\}$ and we define

$(\{a_1, \dots, a_n\}, b)^\sharp = (\{[(a_1)^\flat]_{\nabla}, \dots, [(a_n)^\flat]_{\nabla}\}, [(a)^\flat]_{\nabla})$. It is easy to see that such a map is bijective, since $(\cdot)^\flat$ is. Last, we choose as i_∇ exactly that of Definition 5.3. It is easy to check that $(i((x, a)))^\flat = i_\nabla((x, a)^\sharp)$, for all $(x, a) \in |X \Rightarrow X|$. So, the proof is done.

The proof of point (ii) is similar, taking $(j)^\sharp = [\nu_\nabla]_{\nabla}$. □

Corollary 5.8 (Completeness)

Let \mathcal{M} and \mathcal{M}^j be respectively a linear λ -model and a lazy λ -model.

- (i) *There is a legal type system ∇ inducing a clique model that concides with \mathcal{M} .*
- (ii) *There is a legal lazy type system ∇ inducing a lazy clique model that concides with \mathcal{M}^j .*

Proof. By Lemma 5.7 we can build a legal type system ∇ inducing a linear model coinciding with \mathcal{M} . Thus, the proof follows by Theorem 5.5. The proof of point (ii) is similar. □

6 Conclusions and comparison with related works

Logical semantics in the setting of coherence spaces and stable functions has been previous studied in [6] and [17]. In [17] a class of stable λ -models is considered, containing as proper subclass the linear stable models. Let R be any subset of $X \Rightarrow X$, called the set of *representable functions* in [17]; they consider stable models based on a space X containing as retract R where the retract is realized by linear functions. Thus our linear models are a proper subset of them. Starting from a λ -model \mathcal{M} as that defined in [17], a formal system is designed, assigning to λ -terms tokens of the coherence space on which \mathcal{M} is based. This formal system provides a logical description of the interpretation function of the model \mathcal{M} : the fact that the token a belongs to the interpretation of a term M is logically equivalent to the fact that the formal system is able to assign a to M . From this formal system, three type assignment systems are designed, in order to study three particular stable λ -models. All these models are built as an inverse limit solution of a domain equation, which is $X \approx X \Rightarrow X$ for two of them and $X \approx N \otimes (X \Rightarrow X)$ for the other, where N is the coherence space of natural numbers.

The rules of our type assignment system \vdash are in some sense inspired by this work. But in [17] the formal system is guided by the semantics, since it manipulates directly tokens of a coherence space, while we define the type assignment system in a completely syntactical manner, and then the legality condition assure us that the resulting clique model is isomorphic to a stable model. So we prove an isomorphism between clique models and stable models, while in [17] an isomorphism is proved between denotational interpretation and “token” assignment.

In [6] a general definition of a model of the lazy λ -calculus (in the sense of [1,3,21]) is given, and two particular stable models for it are built, through two type assignment systems. Both these models are built as an inverse limit solution of the domain equation $X \approx I \& (X \Rightarrow X)$, where I denotes the coherence space with just one token. One of the system defined in [6] is an instance of our system \vdash_{∇}^{ℓ} .

In conclusion we want to point out that (lazy) clique models have a very easy syntactical definition, and they can be used for studying different denotational semantics of λ -calculus, without any acquaintance with coherence spaces and stable functions. The results in Section 5 assure that reasoning in a clique model corresponds to reasoning in a well established mathematical category.

Acknowledgement

The authors would like to thank the anonymous referees for their fruitful suggestions.

References

- [1] S. Abramsky. The lazy lambda-calculus. In D. N. Turner, editor, *Research Topics in Functional Programming*, pages 65–117. Addison-Wesley Publ. Co., 1990.
- [2] S. Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 51(1-2):1–77, 1991.
- [3] S. Abramsky and L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105(2):159–267, 1993.
- [4] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in logic and the foundation of mathematics*. North-Holland, Amsterdam, revised edition, 1984. (First edition, 1981).
- [5] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *The Journal of Symbolic Logic*, 48(4):931–940, Dec. 1983.
- [6] O. Bastonero, A. Pravato, and S. Ronchi Della Rocca. Structures for lazy semantics. In G. de Roever, editor, *Programming Concepts and Methods*, pages 30–48. Chaptman & Hall, 1998. International Conference on Programming Concepts and Methods - PROCOMET'98, 8-12 June 1998, Shelter Island, New York, USA.
- [7] G. Berry. Stable models of typed λ -calculi. In G. Ausiello and C. Böhm, editors, *Fifth International Colloquium on Automata, Languages and Programming - - ICALP'78, Udine, Italy, July 17-21, 1978*, volume 62 of *Lecture Notes in Computer Science*, pages 72–89. Springer-Verlag, 1978.
- [8] M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ -calculus. *Notre Dame Journal of Formal Logic*, 21(4):685–693, Oct. 1980.
- [9] M. Coppo, M. Dezani-Ciancaglini, F. Honsell, and G. Longo. Extended type structure and filter lambda models. In G. Lolli, G. Longo, and A. Marcja, editors, *Logic Colloquium'82*, pages 241–262. Elsevier Science Publishers, Amsterdam, 1984.
- [10] M. Coppo, M. Dezani-Ciancaglini, and M. Zacchi. Type theories, normal forms, and D_∞ -lambda-models. *Information and Computation*, 72(2):85–116, 1987.
- [11] M. Dezani-Ciancaglini, S. Ghilezan, and S. Likavec. Behavioural inverse limit lambda-models. *Theoretical Computer Science*, 316(1–3):49–74, 2004.
- [12] L. Egidi, F. Honsell, and S. Ronchi Della Rocca. Operational, denotational and logical descriptions: a case study. *Fundamenta Informaticæ*, 16(2):149–170, 1992.
- [13] J.-Y. Girard. The System F of variable types, fifteen years later. *Theoretical Computer Science*, 45(2):159–192, 1986.
- [14] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
- [15] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, 1989.
- [16] J. R. Hindley and G. Longo. Lambda calculus models and extensionality. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 26:289–310, 1980.
- [17] F. Honsell and S. Ronchi della Rocca. Reasoning about interpretation in qualitative lambda-models. In M. Broy and C. Jones, editors, *Proceedings of Working Conference on Programming Concepts and Methods - IFIP 2.2*, pages 505–521, Sea of Galilee, Israel, 1990. North-Holland.

- [18] F. Honsell and S. Ronchi Della Rocca. An approximation theorem for topological lambda models and the topological incompleteness of lambda calculus. *Journal of Computer and System Sciences*, 45(1):49–75, Aug. 1992.
- [19] C. P. J. Koymans. Models of the lambda calculus. *Information and Computation*, 52(3):306–323, 1982.
- [20] A. Meyer. What is a model of the lambda calculus? *Information and Computation*, 52(1):87–122, 1982.
- [21] C.-H. L. Ong. Fully abstract models of the lazy lambda calculus. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science - FOCS'88*, pages 368–376, White Plains, New York, 24–26 Oct. 1988. IEEE Computer Society Press.
- [22] G. D. Plotkin. Set-theoretical and other elementary models of the λ -calculus. In M. Dezani-Ciancaglini, S. Ronchi Della Rocca, and M. Venturini-Zilli, editors, *Theoretical Computer Science*, volume 121(1-2), pages 351–409. Logic, Semantics and Theory of Programming, 6 Dec. 1993. *A Collection of Contributions in Honour of Corrado Bhm on the Occasion of his 70th Birthday*.
- [23] S. Ronchi Della Rocca and L. Paolini. *The Parametric λ -Calculus: a Metamodel for Computation*. Texts in Theoretical Computer Science: An EATCS Series. Springer-Verlag, Berlin, 2004.