A Categorical Model for the Lambda Calculus with Constructors

Barbara Petit

Focus - INRIA - Università di Bologna
barbara.petit@ens-lyon.org

Abstract:

The lambda calculus with constructors is an extension of the lambda calculus with variadic constructors. It decomposes the pattern-matching à la ML into a case analysis on constants and a commutation rule between case and application constructs. Although this commutation rule does not match with the usual computing intuitions, it makes the calculus expressive and confluent, with a rather simple syntax. In this paper we define a sound notion of categorical model for the lambda calculus with constructors. We then prove that this definition is complete for the fragment of the calculus with no match-failure, using the model of partial equivalence relations.

Keywords: Lambda calculus, Pattern matching, Semantics, Categorical model, PER model.

Introduction

Pattern matching is now a key feature in most functional programming languages. Inherited from the simple constants recognition mechanism that appeared in the late 60’s (in Snobol or in Pascal for instance), it is now a elaborated feature in main programming languages (ML, Haskell etc.) and some proof assistants (such as Coq or Agda), able to decompose complex data-structures.

Its theoretical aspects are being intensively studied since the 90’s [3, 11]. In particular, several lambda calculi with pattern matching have been proposed [19, 4, 8]. Among them, the lambda calculus with constructors [1] (or λc-calculus) offers the advantage of having simple computation rules. Indeed, the pattern matching à la ML is there decomposed into two atomic rules (a constants analysis rule, and a commutation rule). The rather simple syntax of this calculus together with the decomposition of its powerful computational behaviour into elementary steps stimulate a semantic study of the the λc-calculus from a categorical point of view.

As far as we know, no categorical model had been proposed so far for a calculus with pattern matching. Yet category theory allows to express some generic semantic properties on a calculus, and to factorise many of its different concrete models. Furthermore, when the categorical model is complete, it synthesises exactly the extensional properties of the calculus. Since the description of the models for the pure lambda calculus as Cartesian closed categories with a reflexive object [16], some complete categorical models have been defined for variants of the lambda calculus [7, 17, 6].

In this paper, after a brief presentation of the λc-calculus (Sec. ), we establish a categorical definition of models for it (Sec. 2). We then prove that it is to some extent complete for the
\(\lambda_c\)-calculus, using the standard PER model and some rewriting techniques (Sec. 3). Notice that we only use very basic notions of category theory (knowledge of the first two chapters of [3] is sufficient).

1 The lambda calculus with constructors

The lambda calculus with constructors extends the pure lambda calculus with pattern matching features: a set of constants (that we consider here to be finite of cardinality \(n\)) called \textit{constructors} and denoted by \(c, d\) etc. is added, with a simple mechanism of case analysis on these constants (similar to the \texttt{case} instruction of Pascal):

\[ \{c_1 \mapsto t_1; \ldots; c_k \mapsto t_k\} \cdot c_i \rightarrow t_i \quad \text{(CaseCons)} \]

Although only constant constructors can be analysed, a matching on variant constructors can be performed via a commutation rule between case construction and application:

\[ \{\theta\} \cdot (tu) \rightarrow (\{\theta\} \cdot t) u \quad \text{(CaseApp)} \]

This commutation rule enables simulating any pattern matching \texttt{à la} ML, by generalising the following example: in the \(\lambda_c\)-calculus, the predecessor function on unary integers (represented with the constructors \(0\) and \(S\)) is implemented as\(\text{pred} = \lambda x.\{0 \mapsto 0; S \mapsto \lambda y.y\} \cdot x\). Applying this function to a non zero integer \(Sn\) actually produces the expected result:

\[
\text{pred} (S m) \rightarrow \{0 \mapsto 0; S \mapsto \lambda y.y\} \cdot (S m) \\
\rightarrow (\{0 \mapsto 0; S \mapsto \lambda y.y\} \cdot S) m \rightarrow (\lambda y.y) m \rightarrow m
\]

Formally, the syntax of the \(\lambda_c\)-calculus is defined by the following grammar:

\[
t, u, v := x \mid tu \mid \lambda x. t \mid c \mid \{\theta\} \cdot t
\]

\[
\theta, \phi := \{c_1 \mapsto u_1; \ldots; c_k \mapsto u_k\} \quad \text{(with } k \geq 0 \text{ and } c_i \neq c_j \text{ for } i \neq j)\]

In the terms (denoted by \(t, u\) etc.) the application takes precedence over lambda abstraction and case construct. Notice that constructors, like any terms, can be applied to any number of arguments and thereby are \textit{variadic} (they have no fix arity). We call \textit{data-structure} a term on the form \(c. t_1 \cdots t_k\).

A \textit{case-binding} \(\theta\) is just a (partial) function from constructors to terms, whose domain is written \(\text{dom}(\theta)\). By analogy with sequential notation, we may write \(\theta_u\) for \(u \in \theta\). In order to ease the reading, we may write \(\{c_1 \mapsto u_1; \ldots; c_n \mapsto u_n\} \cdot t\) instead of \(\{c_1 \mapsto u_1; \ldots; c_n \mapsto u_n\}\) \cdot t. The usual definition of the free variables of a term is naturally extended to the new constructions of the calculus, taking care that constructors are not variables (and therefore not subject to substitution nor \(\alpha\)-conversion).

In this calculus, a \textit{match failure} is a term \(\{\theta\} \cdot c\) where \(c \notin \text{dom}(\theta)\). We say that a term is \textit{defined} when none of its subterm is a match failure, and that it is \textit{hereditarily defined} when all this reducts (in any number of steps, including zero) are defined.

Reduction rules are given in Fig. 1. In addition to the usual \(\beta\)-reduction (called \texttt{APPLAM}) and to the two rules presented earlier, there is a rule of commutation between case construct and lambda abstraction (\texttt{CASELAM}) to ensure confluence [1, Cor. 1], and the usual \(\eta\)-reduction (called \texttt{LAMAPP}) as well as a rule of composition of case-bindings (\texttt{CASECASE}) so that the calculus enjoys the \textit{separation property} [1, Theo. 2]. More explanations and examples about this calculus can be found in [2, 12].
2 The categorical model

In this section we may define a notion of a categorical model for the λε-calculus, that we prove to be sound. No deep knowledge in category theory is assumed from the reader, he might just need the identity morphism on A is written \( \text{Id}_A \). Also the correction of the new constructions and the new rules of the calculus. In particular, writing \( \text{App} \) for the typed lambda calculus. To cope with the problem of self application of terms, such a category must be provided with a reflexive object \( 1 \) and the identity morphism on \( A \) by \( \text{Id}_A \) (or simply \( \text{Id} \) if it raises no ambiguity). The \( j \)-th projection morphism of a \( k \)-ary product is written \( \pi^k_j \), or \( \pi^k_i \) if \( k = 2 \). Given some morphisms \( f : A \rightarrow B \), \( g : A \rightarrow C \) and \( h : A \rightarrow C \), \( (f;g) \) denotes the pairing of \( f \) and \( g \), and \( f;h \) the composition of \( f \) and \( h \). The evaluation map of \( A \) and \( B \) is \( \text{ev} : B^A \times A \rightarrow B \) and the curried form of a morphism \( f \) is written \( \Lambda(f) \).

2.1 \( \lambda_\varepsilon \)-models

It is well known [10] that Cartesian closed categories have exactly the good structure to interpret the typed lambda calculus. To cope with the problem of self application of terms, such a category must be provided with a reflexive object \( D \) in order to interpret the untyped lambda calculus [10]. Terms are then interpreted by points of \( D \). The denotation of applications is constructed with a morphism \( \text{app} : D \rightarrow D^D \), and the one of lambda abstractions with a morphism \( \text{lam} : D^D \rightarrow D \). Also the correction of the \( \beta \)-reduction is ensured by the equality \( \text{lam;app} = \text{Id}_{D^D} \) (if moreover \( \text{app;lam} = \text{Id}_D \), then the model satisfies the \( \eta \)-equivalence).

Building a model for the \( \lambda_\varepsilon \)-calculus requires some extra morphisms and equalities for the new constructions and the new rules of the calculus. In particular, writing \( \{ c_1, \ldots, c_n \} \) the set of constructors, a special point \( c^*_i \) of \( D \) is needed for each \( i \leq n \) to interpret them. The denotations of case-bindings are then points of \( D^n \). A case binding \( \theta \) is interpreted by the \( n \)-tuple \( (d_1; \ldots; d_n) \) where \( d_i \) is the denotation of \( \text{dom}(\theta) \), and is a special point \( \xi \) representing match failure otherwise. In order to interpret case constructs, we need a morphism \( \text{case} : D^n \times D \rightarrow D \), that transforms the denotation of \( \theta \) and \( t \) into the one of \( \{ \theta \} \cdot t \).

Let us informally confuse terms and their denotations, and write a case-binding \( \{ c_i \mapsto u_i \mid 1 \leq i \leq n \} \) as \( \{ \xi \mapsto \xi \} \) and its denotation as \( \xi \). Then the rule \( \text{CASECONS} \) is valid if \( \{ \xi \mapsto \xi \} \cdot c_i \) and \( u_i \) have the same denotation, i.e. intuitively if \( \text{case}(\xi, c_i) = \pi^n_i(\xi) \). This is formally expressed by the commutation of the diagram (D2) in Fig. 2.

In the same way, the rule \( \text{CASEAPP} \) is valid if the diagram (D3) commutes, i.e. if
\[
D^n \times D \times D \xrightarrow{\text{app}} D^n \times D^D \times D \xrightarrow{\text{ev}} D^n \times D \xrightarrow{\text{case}} D
\]

\[(\vec{u}, t, t') \mapsto (\tilde{x}, \tilde{x}.t, t') \mapsto (\tilde{x}.(\tilde{\xi} \mapsto \tilde{u}).t, t') \mapsto (\tilde{\xi} \mapsto \tilde{u}) \cdot t \cdot t'
\]

(\text{where } \tilde{x}.v \text{ represents the function mapping } v_0 \text{ to } v[x := v_0])

is equal to

\[
D^n \times D \times D \xrightarrow{\text{case} \times} D \times D \xrightarrow{\text{app} \times} D^D \times D \xrightarrow{\text{ev}} D
\]

\[(\tilde{u}, t, t') \mapsto (\tilde{\xi} \mapsto \tilde{u}) \cdot t \cdot t' \mapsto (\tilde{x}.(\tilde{\xi} \mapsto \tilde{u}).t, t') \mapsto (\tilde{\xi} \mapsto \tilde{u}) \cdot t \cdot t'
\]

To express the rule \texttt{CaseLam} we need a morphism that abstracts the case construct \textit{w.r.t.} a variable:

\[
\text{case}^\circ = \Lambda (f_{\text{case}}) : D^n \times D^D \to D^D
\]

\[(\tilde{u}, \tilde{x}.t) \mapsto (\tilde{x}.(\tilde{\xi} \mapsto \tilde{u}).t, t)
\]

where \(f_{\text{case}} = (D^n \times D^D) \times D \xrightarrow{\cong} D^n \times (D^D \times D) \xrightarrow{\text{Id}_{D^n} \times \text{ev}} D^n \times D \xrightarrow{\text{case}} D.
\]

Then the rule \texttt{CaseLam} is valid if (D4) commutes:

\[
D^n \times D^D \xrightarrow{\text{case}^\circ} D^D \xrightarrow{\text{lam}} D = D^n \times (D^D \times D) \xrightarrow{\text{Id}_{D^n} \times \text{ev} \times \text{lam}} D^n \times D \xrightarrow{\text{case}} D
\]

\[(\tilde{u}, \tilde{x}.t) \mapsto \tilde{x}.(\tilde{\xi} \mapsto \tilde{u}).t \cdot \tilde{x}.(\tilde{\xi} \mapsto \tilde{u}).t \cdot\lambda x.(\tilde{\xi} \mapsto \tilde{u}).t \cdot \tilde{u}, \tilde{x}.(\tilde{\xi} \mapsto \tilde{u}).t \mapsto \tilde{\xi} \mapsto \tilde{u} \cdot \lambda x.t
\]

Also the rule \texttt{CaseCase} requires a morphism to compose case-bindings:

\[
\bullet : D^n \times D^n \to D^n
\]

\[(\tilde{u}, (t_i)_{i=1}^n) \mapsto (\tilde{\xi} \mapsto \tilde{u}) \cdot t_i
\]

It is defined as the pairing of the morphisms \((\text{Id}_{D^n} \times \pi_i^n) \cdot \text{case}, 1 \leq i \leq n\). So it is the unique morphism that makes the diagram on the following commute.

Then the commutation of the diagram (D5) validates the rule \texttt{CaseCase}.

This leads to the following definition.

\textbf{Definition 2.1 (\lambda_V-model)} A categorical model for the untyped \lambda_V-calculus is \(\mathcal{M} = (\mathbb{C}, D, \text{app}, \text{lam}, (c_i^*)_{i=1}^n, \xi, \text{case})\) where

- \(\mathbb{C}\) is a Cartesian closed category,
- \(D\) is an object of \(\mathbb{C}\),
- All the \(c_i^*\)s and \(\xi\) are points of \(D\),
- \text{app} is a morphism of \(D \to D^D\), \text{lam} is a morphism of \(D^D \to D\) and \text{case} a morphism of \(D^n \times D \to D\),
- The six diagrams of Fig. 2 commute (the diagram (D2) must commute for every \(i \in [1..n]\)).
Equivalent definition. In fact we can simplify the definition of a $\lambda_\varepsilon$-model, since the isomorphism $D \cong D^D$ entails the equivalence of the diagrams (D3) and (D4). This can be understood from a syntactical point of view, given that the commutation of the diagram (D3) validates the rule CaseApp and the one of (D4) validates CaseLam. Indeed, the only role of CaseLam in the calculus is to close a critical pair created by the rule CaseApp [1, Theo. 1, (CC3)].

Proposition 2.1 If $\text{lam}$ and $\text{app}$ form an isomorphism between $D$ and $D^D$, then the diagram (D3) commutes if and only if the diagram (D4) commutes.

Proof:
Since (D1) commutes, (D4) commutes iff the diagram on the right commutes.
Write $f = \text{Id}_{D^n} \times \text{lam}; \text{case}; \text{app}$.
Since $\text{case}^o = \Lambda(\cong; \text{Id}_{D^n} \times \text{ev}; \text{case})$, and by uniqueness of the exponential, $f = \text{case}^o$ if and only if the following diagram commutes:

$D^n \times D^D \xrightarrow{\text{case}^0} D^D$

$\text{Id} \times \text{lam}$

$D^n \times D \xrightarrow{\text{case}} D$

Figure 2: Commuting diagrams in a $\lambda_\varepsilon$-model
We can detail this diagram as follows:

\[
\begin{align*}
(D^n \times D^D) \times D & \xrightarrow[\cong]{	ext{Id} \times \text{app} \times \text{ld}} D^n \times \text{app} \times (D^D \times D) \xrightarrow[\cong]{\text{ld} \times \text{ev}} D^n \times D \\
(D^n \times D) \times D & \xrightarrow[\cong]{(\text{ld} \times \text{ev})} D^n \times (D \times D) \\
(D^n \times D) & \xrightarrow[\cong]{\text{ld} \times \text{ev}} D^n \times D \\
(D^D \times D) \times D & \xrightarrow[\cong]{\text{ld} \times \text{ev}} D^D \times D \\
D \times D & \xrightarrow[\cong]{\text{app} \times \text{ld}} D^D \times D \\
& \xrightarrow[\cong]{\text{ev}} D 
\end{align*}
\]

Since the sub-diagram in the upper-left corner commutes, then \((D4)\) commutes if and only if \((D3)\) commutes. □

Thus we can omit the commutation of \((D3)\) or the one of \((D4)\) in the definition of a \(\lambda_{\ell}\)-model.

### 2.2 Soundness

In the previous section we gave some intuitions on how to interpret \(\lambda_{\ell}\)-terms in a \(\lambda_{\ell}\)-model. Formally, the denotation \([t]_\Gamma\) of a term \(t\) in such a category is defined by structural induction (in Fig. 3). It depends on a list of variables \(\Gamma = x_1, \ldots, x_k\) that must contain all the free variables of \(t\), and its a morphism of \(D^k \rightarrow D\). Similarly, the denotation \([\theta]_\Gamma\) of a case-binding \(\theta\) with free variables in \(\Gamma\) is a morphism of \(D^k \rightarrow D^n\). We show that this definition provides a correct model of the \(\lambda_{\ell}\)-calculus (we write \(\simeq_{\lambda_{\ell}}\) for the reflexive symmetric transitive closure of its six rules).

\[
\begin{align*}
[x_i]_\Gamma &= \pi^k_i : D^k \rightarrow D \\
[tu]_\Gamma &= D^k \xrightarrow{(\theta)_\Gamma;[u]_\Gamma} D \times D \xrightarrow{\text{app} \times \text{ld} \times D} D^D \times D \xrightarrow{\text{ev}} D \\
[\lambda x_{k+1}.t]_\Gamma &= D^k \xrightarrow{\lambda(f_i)} D^D \xrightarrow{\text{ld}} D \\
\text{where } f_i &= D^k \times D \xrightarrow{\cong} D^{k+1} \xrightarrow{[\theta]_\Gamma; x_{k+1}} D \\
[c]_\Gamma &= D^k \xrightarrow{1} 1 \xrightarrow{e^*} D \\
[\{\theta\} \cdot t]_\Gamma &= D^k \xrightarrow{(\theta)_\Gamma;[t]_\Gamma} D^n \times D \xrightarrow{\text{case}} D \\
[\theta]_\Gamma &= \langle f_1; \ldots; f_n \rangle : D^k \rightarrow D^n, \quad \text{where } f_i = \left\{ \begin{array}{ll} [u_i]_\Gamma & \text{if } c_i \mapsto u_i \in \theta \\ !D^k; \xi & \text{if } c_i \notin \text{dom}(\theta) \end{array} \right. 
\end{align*}
\]

Figure 3: Interpretation of \(\lambda_{\ell}\)-terms in a categorical model

**Theorem 2.2 (Soundness)** If \(\mathcal{M} = (C, D, \text{lam}, \text{app}, (c^*_i)_{i=1}^n, \text{case}, \xi)\) is a \(\lambda_{\ell}\)-model, then for any \(\lambda_{\ell}\)-term \(t, t'\) whose free variables are in \(\Gamma\),

\[
t \simeq_{\lambda_{\ell}} t' \implies [t]_\Gamma = [t']_\Gamma
\]

To prove this theorem, we fix a \(\lambda_{\ell}\)-model \(\mathcal{M} = (C, D, \text{lam}, \text{app}, (c^*_i)_{i=1}^n, \text{case}, \xi)\) and use some preliminary lemmas. The first one expresses that the morphism \(\bullet\) actually corresponds to case-composition. This is where we technically need the diagram \((D6)\), even though its semantic meaning is not as intuitive as for the other one.
Lemma 2.3 (Categorical case-composition) If the diagram (D6) commutes, then for any case-bindings $\theta$ and $\phi$, whose free variables are in $\Gamma = \{x_1, \ldots, x_k\}$, the following diagram commute:

$$
\begin{array}{ccc}
D^k & \xrightarrow{([\theta] \cdot [\phi]_\Gamma)} & D^n \times D^n \\
\downarrow & & \downarrow \\
D^n
\end{array}
$$

**Proof:** If $\phi = \{c_i \mapsto u_i/i \in J\}$ (with $J \subseteq \{1..n\}$), then

$$
[\theta \circ \phi]_\Gamma = \langle f_1, \ldots, f_n \rangle , \quad \text{with } f_i = \begin{cases} 
[\{\theta\} \cdot u_i]_\Gamma & \text{if } i \in J \\
!_{D^n}; \cdot & \text{if } i \notin J 
\end{cases}
$$

On the other hand, $\bullet = \langle((Id_{D^n} \times \pi^n_1); \text{case}), \ldots, ((Id_{D^n} \times \pi^n_1); \text{case})\rangle$. So

$$
\langle[\theta]_\Gamma, [\phi]_\Gamma\rangle ; \bullet = \langle g_1, \ldots, g_n \rangle , \quad \text{with } g_i = \langle [\theta]_\Gamma, ([\phi]_\Gamma; \pi^n_i)\rangle ; \text{case} \quad \text{which is } f_i.
$$

If $i \in J$, then $[\phi]_\Gamma; \pi^n_i = [u_i]_\Gamma$ and then $g_i = \langle [\theta]_\Gamma, [u_i]_\Gamma\rangle ; \text{case}$ which is $f_i$.

If $i \notin J$, then $[\phi]_\Gamma; \pi^n_i = !_{D^k} \times \cdot$. Hence

$$
g_i = D^k \xrightarrow{([\theta]_\Gamma;1!_{D^k})} D^n \times 1 \xrightarrow{!_{D^n} \times \cdot} D^n \times D \xrightarrow{\text{case}} D
$$

$$
= D^k \xrightarrow{([\theta]_\Gamma;!_{D^k})} D^n \times 1 \xrightarrow{\pi_2} 1 \xrightarrow{\cdot} D \quad \text{(by (D6))}
$$

$$
= D^k \xrightarrow{!_{D^k}} 1 \xrightarrow{\cdot} D
$$

So $g_i = f_i$ for any $i \leq n$, and $\langle[\theta]_\Gamma, [\phi]_\Gamma\rangle ; \bullet = [\theta \circ \phi]_\Gamma$. \qed

We also need the standard following lemmas.

Lemma 2.4 (Contextual rules) Exchange: Let $\Gamma = \{x_1, \ldots, x_k\}$ and $\sigma$ be a substitution over $[1..k]$. Write $\sigma(\Gamma) = \{\sigma(1), \ldots, \sigma(k)\}$. Then, for any term $t$ whose free variables are in $\Gamma$,

$$
[\sigma(t)]_\Gamma = \langle \pi^{k}_{\sigma(1)}, \ldots, \pi^{k}_{\sigma(k)} \rangle ; [t]_{\sigma(\Gamma)}.
$$

Weakening: Let $\Gamma = \{x_1, \ldots, x_k\}$ containing all free variables of a term $t$, and $y \notin \Gamma$. Then

$$
[t]_{\Gamma,y} = \langle \pi^{k+1}_{1}, \ldots, \pi^{k+1}_{k} \rangle ; [t]_\Gamma.
$$

Lemma 2.5 (Substitution) Given $\Gamma = \{x_1, \ldots, x_k\}$, and two terms $t$ and $u$ such that $\text{fv}(u) \subseteq \Gamma$ and $\text{fv}(t) \subseteq \Gamma \cup \{y\}$,

$$
[t[y := u]]_\Gamma = D^k \xrightarrow{([\text{id}, u]_\Gamma)} D^k \times D \xrightarrow{\sim} D^{k+1} \xrightarrow{[\cdot]_{\Gamma,y}} D
$$

The soundness theorem is then a direct corollary of the following proposition, that is proved (in appendix A) by structural induction:

**Proposition 2.6** If $\mathcal{M} = (\mathbb{C}, D, \text{lam}, \text{app}, (c^n)_{i=1}^n, \text{case}, \cdot)$ is a $\lambda\varepsilon$-model, then for any $\Gamma = \{x_1, \ldots, x_k\}$ and any terms $t_1, t_2$ such that $\text{fv}(t_1) \subseteq \Gamma$ and $t_1 \rightarrow t_2$, the interpretation given in Fig. 3 satisfies $[t_1]_\Gamma = [t_2]_\Gamma$.  

7
3 Completeness

In this part we shall prove that the converse of Theo. 2.2 holds in absence of match failure. Namely if two terms have the same interpretation in any \( \lambda \varepsilon \)-model then they are convertible using the rules of the calculus. It means that, without match failure, the diagrams of Fig. 2 are minimal.

**Theorem 3.1 (Completeness)** If \( t \) and \( t' \) are two hereditarily defined \( \lambda \varepsilon \)-terms such that in any categorical \( \lambda \varepsilon \)-model \([t] = [t']\), then

\[
t \simeq_{\lambda \varepsilon} t'.
\]

Notice that this theorem does not hold for undefined terms. Indeed, every match failure receives the same denotation \( \dashv \) in any \( \lambda \varepsilon \)-model, even though they are not \( \lambda \varepsilon \)-convertible. The completeness result is established using the same method as [6]:

1. We define \( \text{PER}_{\lambda \varepsilon} \), the Cartesian closed category of partial equivalence relation compatible with \( \simeq_{\lambda \varepsilon} \).
2. In this syntactic category, we construct a \( \lambda \varepsilon \)-model \( \mathcal{M}_{\text{synt}} \).
3. Then we show that if \([t] = [t']\) in \( \mathcal{M}_{\text{synt}} \), then \( t \simeq_{\lambda \varepsilon} t' \).

### 3.1 Partial equivalence relations

Partial equivalence relations (PER) are commonly used to transform a model of the untyped lambda calculus into a model of the typed lambda-calculus [9, 18]. Yet we use them here to instantiate the definition of \( \lambda \varepsilon \)-models in the category of PER on \( \lambda \varepsilon \)-terms. Thereby we construct a syntactic model of the untyped \( \lambda \varepsilon \)-calculus.

**Definition 3.1 (\( \lambda \varepsilon \)-per)** Given a set \( X \), a partial equivalence relation on \( X \) is a binary relation \( R \) that is symmetric and transitive. We may write \( t = t' : R \) instead of \((t, t') \in R\). A \( \lambda \varepsilon \)-per is a partial equivalence relation \( R \) on \( \Lambda \) (the set of all \( \lambda \varepsilon \)-terms) that is compatible with \( \lambda \varepsilon \)-equivalence, which means:

\[
\begin{align*}
&\{ t = t' : R \\
& t_0 \simeq_{\lambda \varepsilon} t' \implies t = t_0 : R
\}
\end{align*}
\]

We write \( \overline{\{ t \}} \) the equivalence class of an element \( e \) modulo \( R \) (or simply \( \overline{\{ t \}} \) when it raises no ambiguity), and if it is non empty we say that \( e \) is accessible by \( R \). This is denoted by \( e \in \text{dom}(R) \).

We call the domain of \( R \) (denoted by \( \text{dom}(R) \)) the set of all its accessible elements modulo \( R \):

\[
\text{dom}(R) = \{ \overline{\{ t \}} / e \in R \}.
\]

Notice that if a partial equivalence relation \( R \) is compatible with \( \lambda \varepsilon \)-equivalence, then by definition

\[
t \simeq_{\lambda \varepsilon} t' \implies \overline{\{ t \}} = \overline{\{ t' \}}.
\]

(1)

It is well known that the family of partial equivalence relations can be provided with the usual semantic operators (arrow, and product) and constitute a CCC [15, Theo 7.1] To this end, we use the well-known Church’s encoding for tuples:

\[
\begin{align*}
&\langle x_1, \ldots, x_k \rangle_k = \lambda f. \langle f \rangle x_1 \ldots x_k \\
&\pi_i^k = \lambda p. (\lambda x_1 \ldots x_k. x_i) (i \in [1..k])
\end{align*}
\]

(We may write \( \langle x, y \rangle \) for \( \langle x, y \rangle_2 \) and \( \pi_i \) for \( \pi_i^2 \)). It satisfies the expected equivalence:

\[
\pi_i^k \langle t_1, \ldots, t_k \rangle_k \simeq_{\lambda \varepsilon} t_i.
\]
Proposition 3.2 (Operations on \(\lambda_{\varepsilon}-\text{pers}\)) Let \((R_i)_{1 \leq i \leq n}\) be a family of PERs (with \(n \geq 2\)). Define \(R_1 \rightarrow R_2\) and \(R_1 \times \ldots \times R_n\) by

\[
t = t' : R \rightarrow R' \quad \text{when} \quad \text{for any} \ u, u' \ : R \implies tu = t'u' : R'
\]
\[
t = u : R_1 \times \ldots \times R_k \quad \text{when} \quad \text{for each} \ i \in [1..k], \ \pi_i^k t = \pi_i^k u : R_i
\]

Then if all the \(R_i\)'s are \(\lambda_{\varepsilon}-\text{pers}\), so are \(R_1 \rightarrow R_2\) and \(R_1 \times \ldots \times R_n\).

The category \(\text{Per}_{\lambda_{\varepsilon}}\). The previous proposition enables providing the category of \(\lambda_{\varepsilon}-\text{pers}\) with the structure of a CCC. In the category \(\text{Per}_{\lambda_{\varepsilon}}\), objects are the PERs compatible with \(\lambda_{\varepsilon}\), and given two \(\lambda_{\varepsilon}-\text{pers}\) \(A\) and \(B\) the morphisms of \(A \rightarrow B\) are the equivalence classes in \(\text{dom}(A \rightarrow B)\). The identity morphism on \(A\) is \(\lambda x. x^{1\rightarrow A}\), and the composition of \(\overline{t} : A \rightarrow B\) and \(\overline{t}' : B \rightarrow C\) is \(\overline{t};\overline{t}' = \lambda z. t'(tz)^{A \rightarrow C}\). This defines correctly a category, as the composition is associative and has identity morphisms as neutral elements.

The categorical product of two \(\lambda_{\varepsilon}-\text{pers}\) \(A\) and \(B\) is \(\langle A \times B, \pi_1^1 A \times B \rightarrow A, \pi_2^1 A \times B \rightarrow B\rangle\), and for \(\overline{t} : C \rightarrow A\) and \(\overline{t}' : C \rightarrow B\), the pairing of \(\overline{t}_1\) and \(\overline{t}_2\) is \(\langle \overline{t}_1, \overline{t}_2 \rangle = \lambda x. \langle tl(x), tl'(x) \rangle^{C \rightarrow A \times B}\). It is well defined (in particular it does not depend on the representative that we chose in the equivalence classes \(\overline{t}\) and \(\overline{t}'\)) and is universal for the diagram on the right. The terminal object is the maximal \(\lambda_{\varepsilon}-\text{per} 1 = \Lambda \times \Lambda\).

The exponent of \(A\) and \(B\) is \(B^A = A \rightarrow B\), and the corresponding evaluation morphism is \(\text{ev} = \lambda x. \langle \pi_1 x, \pi_2 x \rangle^{B^A A \rightarrow B}\).

The curried form of a morphism \(\overline{t} : C \times A \rightarrow B\) is then \(\Lambda(\overline{t}) = \lambda x. \lambda y. t \langle x, y \rangle^{C \rightarrow B^A}\). It is well defined and is the unique morphism that makes the diagram on the right commute.

**Proposition 3.3** \(\text{Per}_{\lambda_{\varepsilon}}\) is a Cartesian closed category.

### 3.2 Syntactic model in \(\text{Per}_{\lambda_{\varepsilon}}\).

We will now define a \(\lambda_{\varepsilon}\)-model in the CCC \(\text{Per}_{\lambda_{\varepsilon}}\). In this category, there is a trivial reflexive object, that is actually equal to its object of functions (as proved in appendix B.1).

**Lemma 3.4** Let \(D\) be the object \(\simeq_{\lambda_{\varepsilon}}\) in \(\text{Per}_{\lambda_{\varepsilon}}\). Then \(D = D^D\).

Also \(\simeq_{\lambda_{\varepsilon}}\) is the object of \(\text{Per}_{\lambda_{\varepsilon}}\) that will be used to interpret untyped \(\lambda_{\varepsilon}\)-terms. We do not need to define \(\text{lam}\) and \(\text{app}\), and the morphisms \(c^1\)’s and \(\text{case}\) are quite intuitive: informally, \(c^*\) is the constant function returning \(c\), and \(\text{case}\) takes an argument \((\theta, t)\) in \(D^n \times D\) and return \(\{\theta\} \cdot t\). In the same way, \(\check{\cdot}\) is just a constant function returning a match failure (we arbitrarily choose one of the possible ones). This actually defines a \(\lambda_{\varepsilon}\)-model (appendix B.1).

**Definition 3.2 (Syntactic model)** The syntactic model (or PER model) of the \(\lambda_{\varepsilon}\)-calculus is \(\mathcal{M}_{\text{syn}} = (\text{Per}_{\lambda_{\varepsilon}}, D, \text{Id}_D, \text{Id}_D, (c^1_i)_{1 \leq i \leq n}, \text{case}, \check{\cdot})\), where:

- \(D\) is the relation \(\simeq_{\lambda_{\varepsilon}}\).
- given \(c\) a constructor, \(c^*\) is \(\lambda x. c^{1D}\).
• case is \( \lambda x. \{ (c_1 \mapsto \pi^n_1(x)) | 1 \leq i \leq n \} \cdot \pi^m_2x \)
  \((D^n \times D) \to D\).

• \( \xi \) is \( \lambda x. \{ \} \cdot c_1 \to D \).

**Proposition 3.5** \( \mathcal{M}_{synt} \) is a \( \lambda_\infty \)-model.

**Case-binding completion.** Remember that \( \lambda_\infty \)-models do not distinguish different match failures (as a matter of fact, all of them are interpreted by \( \xi \)). That is because the interpretation of a term first “completes” each case-binding with branches \( c_j \mapsto \xi \) if \( c_j \) is not in its domain (cf. the description of the denotation of a case-binding page 3). Also in the PER model, undefined terms are “unblocked” and the rule CaseCons can be performed (and give \( \{ \} \cdot c_1 \)).

Now we formalise the idea of case-binding completion. This enables an explicit definition of the interpretation of a term in the PER model, so that we can prove the completeness theorem.

**Definition 3.3 (Case-completion)** The case-completion \( \tilde{t} \) of a term \( t \) is defined by induction:

- \( \tilde{x} = x \)
- \( \tilde{c} = c \)
- \( \tilde{\theta} = \{ c_i \mapsto u'_i/1 \leq i \leq n \} \) with \( u'_i = \begin{cases} \tilde{u}_i & \text{if } c_i \mapsto u_i \in \theta \\ \{ \} \cdot c_1 & \text{if } c_i \notin \text{dom(\theta)} \end{cases} \)

**Fact 3.4** This case-completion does not unify different defined terms: if two defined terms have the same case-completion, then they are equal.

**Proposition 3.6** In the model \( \mathcal{M}_{synt} \), the interpretation of a term \( t \) in a context \( \Gamma = \pi^1 \cdot \ldots \cdot \pi^k \) is

\[
[t]_\Gamma = \lambda x.\tilde{t}[x_1 := \pi^k x]^{D^k \to D}
\]

**(with x fresh in t).**

**3.3 Completeness result.**

The proposition 3.6 ensures that if two \( \lambda_\infty \)-terms have the same denotation in the PER model, then they have the same case-completion modulo \( D \) (i.e. they are \( \lambda_\infty \)-convertible). It does not necessarily mean that the two terms are \( \lambda_\infty \)-equivalent themselves, as it is not true for match failure:

\[
\{ c_1 \mapsto \tilde{\lambda}y.y \cdot \pi^2 \} \cdot c_2 = \{ c_1 \mapsto \lambda y.y \cdot \pi^2 \} \cdot c_1 \cdot c_2 \simeq_{\lambda_\infty} \{ \} \cdot \pi^2 \cdot c_1
\]

Nevertheless, \( \{ c_1 \mapsto \lambda y.y \} \cdot c_2 \not\simeq_{\lambda_\infty} \{ c_2 \mapsto \lambda y.y \} \cdot c_1 \). This explains why match failure all have the same interpretation in \( \mathcal{M}_{synt} \). However, this defect is restricted to undefined terms. Now we show that the case-completion does not modify the \( \lambda_\infty \)-equivalence on defined terms.

**Proposition 3.7** Let \( t_1 \) and \( t_2 \) be two hereditarily defined terms. Then

\[
\tilde{t}_1 \simeq_{\lambda_\infty} \tilde{t}_2 \implies t_1 \simeq_{\lambda_\infty} t_2
\]

The proof of this proposition uses rewriting techniques, and relies on several lemmas (whose proofs are given in appendix B.2). For technical reasons, we need to separate the rule CaseCase from the other ones. Also we write \( \lambda_\infty \) the calculus with all the rules except CaseCase, and cc the rule CaseCase.

**Fact 3.5** The definition of case-completion (Def. 3.3) preserves all \( \lambda_\infty \)-redexes. Also if \( t \to u \) then \( \tilde{t} \to \tilde{u} \), and if \( \tilde{t} \) is a normal form then so is \( t \).
Lemma 3.8 (Reduction on completed terms)  1. Let $t$ be a defined term.
   Then, for any term $t'$,
   
   $$\tilde{t} \xrightarrow{\lambda} t' \implies t' = \tilde{t}_0 \text{ for some } t_0 \text{ such that } t \rightarrow t_0.$$
   
   2. For any terms $t, t'$,
   
   $$\tilde{t} \xrightarrow{cc} t' \implies t' \xrightarrow{cc} \tilde{t}_0 \text{ for some } t_0 \text{ such that } t \rightarrow cc t_0.$$
   
   The rule CASE does not have the same behaviour as the other rules w.r.t. case-completion, and requires a special attention. It has been proved that the reduction rule CASE forms a confluent [1, Theo. 1] and strongly normalising [1, Prop. 2] rewriting system. So every $\lambda$-term $t$ has a unique normal form $\downarrow t$ for the rule CASE. It is characterised by the following equations:
   
   $$\downarrow x = x, \quad \downarrow \{c_i \mapsto u_i / i \in I\} = \{c_i \mapsto \downarrow u_i / i \in I\}$$
   $$\downarrow c = c, \quad \text{if } t = x \mid c \mid \lambda x. u \mid t_1 t_2, \quad \text{then}$$
   $$\downarrow \lambda x. t = \lambda x. \downarrow t, \quad \downarrow \{\theta\} \cdot t = \downarrow \{\theta\} \cdot \downarrow t$$
   $$\downarrow (tu) = \downarrow t \downarrow u, \quad \downarrow (\{\mathbb{I}\} \cdot \{\phi\} \cdot t) = \downarrow (\{\theta \circ \phi\} \cdot t)$$
   
   Lemma 3.9 Commutation case-completion/cc-normal form
   For any term $t$,
   
   $$\downarrow \tilde{t} = \downarrow t.$$
   
   Lemma 3.10 For any terms $t, t'$, if $t \rightarrow^{* \lambda}_u t'$ then there exists a term $u$ such that
   
   $$\downarrow t \rightarrow^{* \lambda} u \rightarrow^{* cc} \downarrow t'.$$
   
   Corollary 3.11 If $t$ is hereditarily defined, then for any $t'$,
   
   $$\tilde{t} \rightarrow^{* cc} t' \implies \downarrow t' = \tilde{t}_0 \text{ for some } t_0 \text{ such that } t \rightarrow^{* cc} t_0.$$
   
   **Proof:** By induction on the reduction $\tilde{t} \rightarrow^{* cc} t'$.
   
   If $\tilde{t} = t'$, take $t_0 = \downarrow t$. Now assume $\tilde{t} \rightarrow^{* cc} u \rightarrow^{R} t'$. By induction hypothesis, there is some $u_0$ such that $\downarrow u = \tilde{u}_0$ and $t \rightarrow^{* cc} u_0$. If $u$ reduces on $t'$ with the rule $R = \text{CASE}$, then $\downarrow t' = \downarrow u = \tilde{u}_0$, and $t_0 = u_0$ does the job. Otherwise, $\tilde{t} \rightarrow^{* cc} u \rightarrow^{\lambda \gamma \delta} t'$.
   
   First of all, $u \rightarrow^{\lambda \gamma \delta} t'$ implies $\downarrow u \rightarrow^{\lambda \gamma \delta} u' \rightarrow^{* cc} \downarrow t'$ for some $u'$ (Lem. 3.10). Also $\tilde{u}_0 \rightarrow^{* cc} u'$, and thus $u' = \tilde{u}_1$ for some term $u_1$ such that $u_0 \rightarrow^{* cc} u_1$ (Lem. 3.8.1, since $u_0$ is defined). Moreover, $\tilde{u}_1 \rightarrow^{cc} \downarrow t'$ implies that $\downarrow t'$ is the CASE-normal form of $\tilde{u}_1$. Hence $\downarrow t' = \downarrow \tilde{u}_1 = \downarrow u_1$ (by Lem. 3.9). Also we can chose $t_0 = \downarrow u_1$. 

   Now we have all the ingredients we need to prove that the case-completion preserves the $\lambda\gamma\delta$-equivalence on hereditarily defined terms.
Let $t_1, t_2$ hereditarily defined such that $\tilde{t}_1 \simeq_{\lambda e} \tilde{t}_2$. Since the $\lambda e$-calculus satisfies the Church-Rosser property, there is a term $u$ such that $\tilde{t}_1 \to^* u$ and $\tilde{t}_2 \to^* u$.

Hence Cor. 3.11 provides a term $u'$ such that $\downarrow u = \tilde{u}'$, and $t_i \to^* u'$ for each $i \in \{1, 2\}$. Thus $t_1 \simeq_{\lambda e} u' \simeq_{\lambda e} t_2$. □

Together with the explicit definition of the interpretation of a term in the PER-model, this gives the result of completeness of $\lambda e$-models for terms with no match failure.

**Corollary 3.12 (Completeness)** Let $t_1$ and $t_2$ be two hereditarily defined terms whose free variables are in $\Gamma = \{x_1, \ldots, x_k\}$ such that $[t_1]_\Gamma = [t_2]_\Gamma$ in the syntactic model $\mathcal{M}_{\text{synt}}$, then $t_1 \simeq_{\lambda e} t_2$.

**Proof:** By Prop. 3.6, if $t_1$ and $t_2$ have the same interpretation in $\mathcal{M}_{\text{synt}}$, it means that

$$\lambda x. t_1[x_i := \pi^k_i x]^{D^k \to D} = \lambda x. t_2[x_i := \pi^k_i x]^{D^k \to D}.$$

Hence $(\lambda x. \tilde{t}_1[x_i := \pi^k_i x]) \langle x_1, \ldots, x_k \rangle_k = (\lambda x. \tilde{t}_2[x_i := \pi^k_i x]) \langle x_1, \ldots, x_k \rangle_k : D$. Since $D$ is the $\lambda e$-equivalence relation on terms, it means that $t_1 \simeq_{\lambda e} t_2$, which entails $t_1 \simeq_{\lambda e} t_2$ by Prop. 3.7. □

_A fortiori_ if two hereditarily defined terms have the same interpretation in _any_ $\lambda e$-model then they are $\lambda e$-equivalent, since $\mathcal{M}_{\text{synt}}$ is a $\lambda e$-model. This achieves the proof of Completeness theorem (Theo. 3.1).

Notice that the separation theorem for the lambda calculus with constructors [1, Theo. 2] specifies that two hereditarily defined terms are either $\lambda e$-equivalent or (weakly) separable. So any terms that can be separated by this syntactic lemma are also semantically distinguished by our definition of model. However a slight modification of this definition could allow to semantically separate more terms. If, instead of having one fail constant $\xi$ we had one for each constructor (say $\xi_1, \text{fail}_2$ etc.), we could “complete” a case binding with the corresponding fail constant in each undefined branch. This would enable keeping track of the constructor that raises the match failure. For instance, $\llbracket c_1 \mapsto \lambda x.x \rrbracket \cdot c_2$ would be denoted by $\xi_2$ and $\llbracket c_1 \mapsto \lambda x.x \rrbracket \cdot c_3$ by $\xi_3$. Only terms like $\llbracket c_1 \mapsto \lambda x.x \rrbracket \cdot c_2$ and $\llbracket c_3 \mapsto \lambda x.x \rrbracket \cdot c_2$ would not be semantically separated.

**Conclusion**

We have defined a notion of categorical model for the lambda calculus with constructors that is reasonably complex: in addition to the usual axioms of a CCC, it involves three morphisms (or family of morphisms) and the commutation of six simple diagrams. We have also proved that this categorical model is complete for terms with no match failure.

Still, completeness does not hold for match failures. This is due to the way we interpret the case-bindings. Since the denotation we give to them is a point of $D^n$, it requires to “fill” artificially every undefined branch of a case-binding. A way to cope with this problem could be to first identify the domain $\mathcal{I} \subseteq \llbracket 1..n \rrbracket$ of a case-binding $\theta = \{c_i \mapsto u_i | i \in I\}$, and interpret it by the point $(u_i)_{i \in \mathcal{I}}$ of $D^n \mathcal{I}$ (where $n_\mathcal{I}$ is the cardinal of $\mathcal{I}$). The object that represents case-bindings...
would then be the sum (the dual notion of product) \( \sum_{I \subseteq \{1..n\}} D^I \). However, the definition loses its relative simplicity and some difficulties arise to define the case composition.

**Future work** A natural question is now to find some concrete instances of the categorical model. The PER model is one, but it would be of great interest to have some non syntactic models. We could try to adapt the historically first model of the pure lambda calculus [14]. However there is no reason for the usual Scott’s \( D_\infty \) domain to satisfy the commutation of our diagrams. A first step could be to find out a domain equation to characterise the lambda calculus with constructors, and then solve it with Scott’s technique.

An other issue is to define a categorical model for the typed \( \lambda \eta \)-calculus [13]. This type system is rather complex, basically because of the reduction rule \textsc{CaseApp} that transforms a sub-term that is \textit{a priori} a function into a sub-term that is \textit{a priori} a data-structure. To deal with this difficulty (and also to enable the typing of variadic constructors), the type syntax includes an application construct and the type system uses sub-typing. Also defining a typed categorical model for the lambda calculus with constructors probably requires a categorical definition of this type application, and a way to express categorically this sub-typing relation.

**References**

[1] Ariel Arbiser, Alexandre Miquel, and Alejandro Ríos. A lambda-calculus with constructors. In *RTA*, pages 181–196, 2006.

[2] Ariel Arbiser, Alexandre Miquel, and Alejandro Ríos. The lambda-calculus with constructors: Syntax, confluence and separation. *Journal of Functional Programming*, 19(5):581–631, 2009.

[3] Andrea Asperti and Giuseppe Longo. *Categories, Types and Structures*. M.I.T. Press, 1991.

[4] Horatiu Cirstea and Claude Kirchner. The rewriting calculus as a semantics of elan. In *ASIAN*, pages 84–85, 1998.

[5] Thierry Coquand. Pattern matching with dependent types. In *Workshop on Types for Proofs and Programs*, pages 66–79, 1992.

[6] Germain Faure and Alexandre Miquel. A categorical semantics for the parallel lambda-calculus. Technical Report 7063, INRIA, October 2009.

[7] Martin Hofmann and Thomas Streicher. Continuation models are universal for lambda-mu-calculus. In *LICS*, pages 387–395, 1997.

[8] C. Barry Jay and Delia Kesner. Pure pattern calculus. In *ESOP*, pages 100–114, 2006.

[9] John C. Mitchell. A type-inference approach to reduction properties and semantics of polymorphic expressions (summary). In *LISP and Functional Programming*, pages 308–319, 1986.

[10] John C. Mitchell. *Foundations for Programming Languages*. The MIT Press, 1996.

[11] C.-H. Luke Ong and Steven James Ramsay. Verifying higher-order functional programs with pattern-matching algebraic data types. In *Principles of Programming Languages*, pages 587–598, 2011.
[12] Barbara Petit. *On the lambda calculus with constructors*. PhD thesis, ENS Lyon, 2011.

[13] Barbara Petit. Semantics of typed lambda-calculus with constructors. *Logical Methods in Computer Science*, 7(1:2), 2011.

[14] Dana S. Scott. Outline of a mathematical theory of computation. Technical report, Princeton University, 1970.

[15] Dana S. Scott. Data types as lattices. *Siam Journal on Computing*, 5(3):522–587, 1976.

[16] Dana S. Scott. Relating theories of the lambda-calculus. In Roger Hindley and Jonathan P. Seldin, editors, *To H.B. Curry: essays in Combinatory Logic, lambda calculus and Formalisms*. Academic Press, 1980.

[17] Peter Selinger. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, 2001.

[18] Val Tannen and Thierry Coquand. Extensional models for polymorphism. In *TAPSOFT, Vol.2*, pages 291–307, 1987.

[19] Vincent van Oostrom. Lambda calculus with patterns. Technical report, Vrije Universiteit, Amsterdam, 1990.
A Proof of Soundness

Proposition 2.6. If $\mathcal{M} = (\mathcal{C}, D, \text{lam}, \text{app}, (c_\pi)_i, \text{case}, \xi)$ is a $\lambda_\xi$-model, then for any $\Gamma = \{x_1, \ldots, x_k\}$ and any terms $t_1, t_2$ such that $\text{fv}(t_1) \subseteq \Gamma$ and $t_1 \rightarrow t_2$, the interpretation given in Fig. 3 satisfies $[t_1]_\Gamma = [t_2]_\Gamma$.

Proof: Let $t_1, t_2$ be two $\lambda_\xi$-terms such that $t_1 \rightarrow t_2$. We prove by induction on the structure of $t_1$ that for any $\Gamma$ containing all free variables of $t_1$, $[t_1]_\Gamma = [t_2]_\Gamma$. If the reduction does not involve a head redex, we immediately conclude with induction hypothesis. So we consider all possible reductions in head position:

- $t_1 = (\lambda x.t) u$ and $t_2 = t[x := u]$.
  
  $[t_1]_\Gamma = D^k \xrightarrow{((\lambda(f_1) : \text{lam}) : [u]_\Gamma)} D \times D \xrightarrow{\text{app} \times Id_D} D^D \times D \xrightarrow{\text{ev}} D$
  
  with $f_1 = D^k \xrightarrow{D \times D} D^{k+1} \xrightarrow{[u]_{t,x}} D$. Thus
  
  $[t_1]_\Gamma = (Id_D, [u]_\Gamma) : (\lambda(f_1) : \text{lam}) \times Id_D ; \text{ev}$
  
  where $x / f_1$ and $[t_1]_\Gamma = [t_2]_\Gamma$ (Def. of exponential) (Lem. 2.5)

- $t_1 = \lambda x.t x$ (with $x \notin \text{fv}(t)$) and $t_2 = t$. Then $[t_1]_\Gamma = \Lambda(f_1) ; \text{lam}$
  
  where $f_1 = D^k \xrightarrow{D \times D} D^{k+1} \xrightarrow{[u]_{t,x}} D \times D \xrightarrow{\text{app} \times Id_D} D^D \times D \xrightarrow{\text{ev}} D$.
  
  But $x \notin \text{fv}(t)$ implies $[t]_{t,x} = (x_{k+1}^1, \ldots, x_{k+1}^n)$; $[t]_\Gamma$ by weakening property (Lem. 2.4), and $[x]_{t,x} = x_{k+1}^i$.
  
  So $f_1 = D^k \xrightarrow{D \times D} D^{k+1} \xrightarrow{[u]_{t,x}} D^D \times D \xrightarrow{\text{ev}} D$.

  By uniqueness of the exponential, $\Lambda(f_1x) = [t]_\Gamma; \text{app}$, and $[t_1]_\Gamma = [t_2]_\Gamma; \text{app}; \text{lam} = [t]_\Gamma$ by (D1).

- $t_1 = \{\theta\} \cdot c_i$ and $t_2 = u_i$, where $\theta = \{c_i \mapsto u_j / j \in J\}$, with $J \subseteq \{1..n\}$.
  
  Then $[t_1]_\Gamma = \{\langle f_1, \ldots, f_n \rangle, [c_i]_\Gamma \} : \text{case}$ with $f_j = \{ [u_j]_\Gamma \text{ if } j \in J \} \text{ and } [c_i]_\Gamma = !D^k : c_i$.
  
  The following diagram commutes: $D^k \xrightarrow{\langle f_1, \ldots, f_n \rangle, !D^k} D^n \times 1 \xrightarrow{\text{Id}_{D^n} \times c_i^*} D^n \times D$.

  so $[t_1]_\Gamma = \langle f_1, \ldots, f_n \rangle ; c_i = f_i = [u_i]_\Gamma$.

- $t_1 = \{\theta\} \cdot (tu)$ and $t_2 = (\{\theta\} \cdot t) u$.
  
  $[t_1]_\Gamma = \{\langle \theta \rangle_\Gamma, [tu]_\Gamma \} : \text{case}$ with $[tu]_\Gamma = \{[t]_\Gamma, [u]_\Gamma \} : (\text{app} \times Id_D) ; \text{ev}$
  
  $[t_2]_\Gamma = \{([\theta]_\Gamma, [t]_\Gamma) \times \text{case} \} : [u]_\Gamma \times (\text{app} \times Id_D) ; \text{ev}$
  
  So $[t_1]_\Gamma = [t_2]_\Gamma$ because the following diagram commutes:
On the other hand, \( \pi \) and \( \beta \) are both \( \pi \)-equivalent to \( \lambda \).

\[
\begin{align*}
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
&\vdash \lambda \cdot \lambda \cdot \lambda \cdot \lambda \cdot \lambda \\
\end{align*}
\]

\end{document}
The upper triangle commutes by uniqueness of the product, the triangle below commutes if (D6) commutes (consequence of Lem. 2.3), and the right part of the diagram is exactly (D5). Also the interpretation is correct w.r.t. CASECASE if (D5) and (D6) commute. □

B Proofs for Completeness

B.1 Some properties of $\mathbb{P}_{\mathit{ER}_{\lambda\varepsilon}}$.

Lemma 3.4. Let $D$ be the object $\simeq_{\lambda\varepsilon}$ in $\mathbb{P}_{\mathit{ER}_{\lambda\varepsilon}}$. Then $D = D^D$.

Proof:

$\subseteq$: If $t = t’ : D$, then $u = u’ : D$ implies $tu = t’u’ : D$ by definition of $D$. This means $t = t’ : D^D$

$\supseteq$: Assume $t = t’ : D^D$, and choose $x$ not free in $t$ nor $t’$. Since $x = x : D$, then $tx = t’x : D$.

Hence $\lambda x. tx = \lambda x. t’x : D$ by contextual closure, and $t = t’ : D$ by LAMAPP. □

Proposition 3.5. Let $\mathcal{M}_{\mathit{synt}} = (\mathbb{P}_{\mathit{ER}_{\lambda\varepsilon}}, D, Id_D, Id_D, (c^*)_{1 \leq i \leq n}, \mathsf{case}, \xi)$, where:

- $D$ is the relation $\simeq_{\lambda\varepsilon}$.
- given $c$ a constructor, $c^*$ is $\lambda x.c^{1D}$.
- case is $\lambda x.\{(c_i : \pi^n_i(\pi_1 x))_{1 \leq i \leq n}\} : \pi^2x^{(D^n \times D) \to D}$.
- $\xi$ is $\lambda x.\{\} : c^1 \to D$.

$\mathcal{M}_{\mathit{synt}}$ is a $\lambda\varepsilon$-model.

Proof: $\mathbb{P}_{\mathit{ER}_{\lambda\varepsilon}}$ is a Cartesian closed category by Prop. 3.3, and $Id_D$ is an isomorphism from $D$ to $D^D$ by Lem. 3.4. We first check that the morphisms are well-defined:

- $c^* \in \text{dom}(1 \to D)$ for each constructor $c$. Indeed, for any terms $u, u’$, $(\lambda x.c) u \simeq_{\lambda\varepsilon} c \simeq_{\lambda\varepsilon} (\lambda x.c) u’$. Hence $\lambda x.c = \lambda x.c : 1 \to D$. In the same way, $\xi \in \text{dom}(1 \to D)$.

- case $\in \text{dom}(D^n \times D \to D)$ since $\lambda x.\{(c_i : \pi^n_i(\pi_1 x))_{1 \leq i \leq n}\} : \pi^2x \in (D^n \times D) \to D$. Indeed, let $t = u : (D^n \times D)$. By definition, $\pi^n_i(\pi_1 t) = \pi^n_i(\pi_1 u) : D$, and $\pi^2t = \pi^2u : D$. Thus $\lambda x.\{(c_i : \pi^n_i(\pi_1 x))_{1 \leq i \leq n}\} : \pi^2x \simeq_{\lambda\varepsilon} \lambda x.\{(c_i : \pi^n_i(\pi_1 u))_{1 \leq i \leq n}\} : \pi^2t \simeq_{\lambda\varepsilon} \lambda x.\{(c_i : \pi^n_i(\pi_1 u))_{1 \leq i \leq n}\} : \pi^2u \simeq_{\lambda\varepsilon} (\lambda x.\{(c_i : \pi^n_i(\pi_1 x))_{1 \leq i \leq n}\} : \pi^2x) u$.

Finally by Prop. 2.1 it is sufficient to show that the diagrams (D1), (D2), (D3), (D5) and (D6) of Fig. 2 commute. For (D1) it is obvious with 1am = app = Id_D. We show the commutation property for the other diagram.

17
(D2): We show that \( \text{rhs} = \pi^n_i \), where \( \text{rhs} = h_{\equiv} ; (Id_{D^n} \times c^n_i) ; \text{case} \) (with \( h_{\equiv} = \pi x.(x, x) \)). Notice that \((Id_{D^n} \times c^n_i) = \lambda x.(\pi x, (\lambda x.c_i)(\pi x))\). We simplify \( \text{rhs} \), considering terms up to \( \lambda y\)-equivalence (1).

\[
\begin{align*}
\text{rhs} & = \lambda z.t_{\text{case}}\left((\lambda x.(\pi x, (\lambda x.c_i)x))\ (\lambda x.(x, x))z\right) \\
& = \lambda z.t_{\text{case}}\left((\pi_1^i z, z)\ (\lambda x.c_i)(\pi_2^i z, z)\right) \\
& = \lambda z.t_{\text{case}}\left((\pi_1^i z)\ (\lambda x.c_i)(\pi_2^i z)\right) \\
& = \lambda z.\{((c_1 \mapsto \pi^n_1(\pi_1^i z, c_i))_{n=1}) \cdot \pi_2^i z\} \\
& = \lambda z.\{((c_1 \mapsto \pi^n_1(\pi_1^i z, c_i))_{n=1}) \cdot \pi_2^i z\} \\
& = \lambda z.\{((c_1 \mapsto \pi^n_1(\pi_1^i z, c_i))_{n=1}) \cdot c_i\} \\
& = \lambda z.\pi^n_i z \ (\text{by CaseCons}) \\
& = \pi^n_i \\
\end{align*}
\]

(D3): We show that \( \text{lhs} = \text{rhs} \), where \( \text{lhs} = (\text{case} \times Id_D) ; (\text{app} \times Id_D) ; \text{ev} \), and \( \text{rhs} = h_{\equiv} ; (Id_{D^n} \times (\text{app} \times Id_D)) ; (Id_{D^n} \times \text{ev}) ; \text{case} \), with

\[
\begin{align*}
\text{lhs} & = \lambda x.(\pi_1^i z, x) ; (\pi_2^i z, x) ; \text{ev} \\
& = \lambda z.(\text{case} (\pi_1^i z) \ (\pi_2^i z)) \\
& = \lambda z.\{((c_1 \mapsto \pi^n_1(\pi_1^i z, c_i))_{n=1}) \cdot \pi_2^i z\} \\
& = \lambda z.\{((c_1 \mapsto \pi^n_1(\pi_1^i z, c_i))_{n=1}) \cdot \pi_2^i z\} \\
& = \lambda z.\{((c_1 \mapsto \pi^n_1(\pi_1^i z, c_i))_{n=1}) \cdot \pi_2^i z\} \ (\text{by CaseApp}) \\
& = \lambda z.\pi^n_i z \\
\end{align*}
\]

(D5): Let \( \text{lhs} = (\bullet \times Id_D) ; \text{case} \), and \( \text{rhs} = h_{\equiv} ; (Id_{D^n} \times \text{case}) ; \text{case} \), with

\[
\begin{align*}
\text{lhs} & = \lambda x.(\pi_1^i x) ; (\pi_2^i x) \\
& = \lambda z.(\text{case} (\pi_1^i z) \ (\pi_2^i z)) \\
& = \lambda z.\{((c_1 \mapsto \pi^n_1(\pi_1^i z, c_i))_{n=1}) \cdot \pi_2^i z\} \\
& = \lambda z.\{((c_1 \mapsto \pi^n_1(\pi_1^i z, c_i))_{n=1}) \cdot \pi_2^i z\} \\
& = \lambda z.\pi^n_i z \ (\text{by CaseCase}) \\
& = \pi^n_i \\
\end{align*}
\]
(D6): This diagram commutes if lhs = rhs, with lhs \( = \pi_2 : \xi \), and rhs \( = (D^n \times 1) \); \textbf{case.}

\[
\begin{align*}
\text{lhs} &= \lambda z.(\lambda x.\{\} \cdot c_1) (\pi_2 z) D^n \times 1 \rightarrow D \\
&= \lambda z.\{\} \cdot c_1 \\
\text{rhs} &= \lambda z.t_{\text{case}} (\pi_1 z, (\lambda x.\{\} \cdot c_1) (\pi_2 z)) D^n \times 1 \rightarrow D \\
&= \lambda z.t_{\text{case}} (\pi_1 z, \{\} \cdot c_1) D^n \times 1 \rightarrow D \\
&= \lambda z.\{\} \cdot c_1 D^n \times 1 \rightarrow D \\
&= \lambda z.\{\} \cdot c_1 (\text{by CASECASE})
\end{align*}
\]

\[D \]

Since \( \text{B.2 Some rewriting properties} \)

\[\text{Proposition 3.6.} \ \text{In the model } M_{\text{synt}}, \text{the interpretation of a term } t \text{ in a context } \Gamma = x_1 : \cdots : x_k \text{ is}
\]

\[\begin{align*}
\langle t \rangle^\Gamma &= \lambda x.\tilde{t}[x_1 := \pi_k^x] D^k \rightarrow D \\
\text{(with } x \text{ fresh in } t).)
\end{align*}
\]

\[\text{Proof:} \ \text{The proof proceeds by structural induction on } t. \ \text{If } t = x_i \text{ or } t = c, \text{ we just have to write the definition of } \langle t \rangle^\Gamma. \ \text{If } t = \lambda x_{k+1}.t_0 \text{ or } t = t_1 t_2, \text{ the equation is straightforward from definition of } \langle t \rangle^\Gamma \text{ and induction hypothesis. We detail the proof when } t = (\langle \theta \rangle : u) \cdot w: \langle t \rangle^\Gamma = (\langle \theta \rangle ; \langle u \rangle ; \text{case}, \text{ with } \langle \theta \rangle^\Gamma = (f_1, \ldots, f_n) \text{ where } f_j = \langle u_j \rangle^\Gamma \text{ if } c_j \mapsto u_j \in \theta, \text{ and } f_j = !D^n \xi \xi (\lambda x.\{\} \cdot c_1 D^k \rightarrow D) \text{ if } c_j \notin \text{dom}(\theta). \ \text{So}
\]

\[\begin{align*}
\langle t \rangle^\Gamma &= \lambda x.\tilde{t}_{\text{case}} (\langle t_0 x, t_u x \rangle) D^k \rightarrow D \\
\text{with case } \tilde{t}_{\text{case}} &= \tilde{t}_{\text{case}} D^n \times D^k \rightarrow D, \langle \theta \rangle^\Gamma = \tilde{t}_{\text{case}} D^k \rightarrow D^n \text{, and } \langle u \rangle^\Gamma = \tilde{t}_{u} D^k \rightarrow D \text{. By induction hypothesis,}
\]

\[\text{we can chose } t_u = (\lambda x.\tilde{u}[x_i := \pi_k^x]), \text{ and } t_\theta = \lambda x.(\langle t_1 x, \ldots, t_n x \rangle)_n \text{ with } t_j = \lambda x.\tilde{u}[x_i := \pi_k^x] \text{ if } c_j \mapsto u_j \in \theta, \text{ and } t_j = \lambda x.\{\} \cdot c_1 \text{ if } c_j \notin \text{dom}(\theta). \ \text{Also}
\]

\[\begin{align*}
\lambda x.\tilde{t}_{\text{case}}, \{t_0 x, t_u x\} &\simeq_{\lambda x} \lambda x.\tilde{t}_{\text{case}}, \{\langle t_1 x, \ldots, t_n x \rangle_n, \tilde{u}[x_i := \pi_k^x]\} \\
&\simeq_{\lambda x} \lambda x.\tilde{u}[x_i := \pi_k^x] \\
&\simeq_{\lambda x} \lambda x.\{\} \cdot c_1 \text{ if } c_j \notin \text{dom}(\theta). \ \text{Indeed,}
\]

\[\begin{align*}
t_j x &\simeq_{\lambda x} \lambda x.\tilde{u}[x_i := \pi_k^x] \text{ if } c_j \mapsto u_j \in \theta, \text{ and } t_j \simeq_{\lambda x} \{\} \cdot c_1 \text{ if } c_j \notin \text{dom}(\theta). \ \text{Since}
\]

\[D^k \rightarrow D \text{ is compatible with } \simeq_{\lambda x}, \text{ } \langle t \rangle^\Gamma = \lambda x.\tilde{t}[x_i := \pi_k^x] D^k \rightarrow D. \ \text{□}
\]

\[B.2 \ \text{Some rewriting properties} \]

\[\text{Lemma 3.8.1 (} \lambda x \text{ reduction on completed terms).}
\]

\[\text{Let } t \text{ be a defined term. Then, for any term } t',
\]

\[\tilde{t} \rightarrow_{\lambda \xi} t' \ \text{ implies } t' = \tilde{t}_0 \text{ for some } t_0 \text{ such that } t \rightarrow t_0.
\]

\[\text{Proof:} \ \text{By structural induction on } t. \ \text{First notice that every CASECONS redex present in } \tilde{t} \text{ corresponds to a CASECONS redex in } t, \text{ as } t \text{ is defined. Moreover, } \{\} \cdot c_1 \text{ is not reducible so}
\]

\[\text{every redex in a sub-term of } \tilde{t} \text{ corresponds to a redex in a sub-term of } t \text{ Also if the reduction } \tilde{t} \rightarrow t' \text{ is performed in a (strict) sub-term of } \tilde{t}, \text{ we can immediately conclude with induction hypothesis.}
\]

\[\text{So it is sufficient to check the lemma for the five possible reductions in head position } \tilde{t} \rightarrow t', \text{ which is trivial.} \ \text{□}
\]

\[\text{Lemma 3.8.2 (CASECASE reduction on completed terms).}
\]

\[\text{For any term } t, t',
\]

\[\tilde{t} \rightarrow_{cc} t' \ \text{ implies } t' = \tilde{t}_0 \text{ for some } t_0 \text{ such that } t \rightarrow_{cc} t_0
\]

\[19\]
Proof: By structural induction on $t$. If the CASECASE reduction occurs in a strict subterm of $\tilde{t}$ then we conclude with induction hypothesis. Otherwise $t = \{θ\} \cdot \{φ\} \cdot u$, and $\tilde{t} = \{θ \circ φ\} \cdot \tilde{u}$. Then we take $t_0 = \{θ \circ φ\} \cdot u$, since $θ \circ φ \rightarrow^*_c θ \circ φ$. Indeed, if $φ = \{c_i \mapsto u_i / i \in I\}$ then
\[
\widetilde{θ \circ φ} = \{c_i \mapsto \{θ\} \cdot \tilde{u}_i / i \in I\} \cup \{c_i \mapsto \{θ\} \cdot \{\} \cdot c_i / i \notin I\}
\]
Also $t' \rightarrow^*_c \widetilde{t}_0$. □

Lemma 3.9 (Commutation case-completion/cc-normal form). For any term $t$,
\[
\downarrow (t) = \widetilde{t}.
\]

Proof: By induction on the size of the maximal reduction $\widetilde{t} \rightarrow^*_c (\widetilde{t})$. If $\widetilde{t} = \downarrow (\widetilde{t})$, then $\widetilde{t}$ is CASECASE-normal, and so is $t$ (Fact 3.5). Thus $t = \downarrow t$ and $\widetilde{t} = \downarrow t$. Otherwise let $\widetilde{t} \rightarrow^*_c t' \rightarrow^*_c \downarrow (\widetilde{t})$. By Lem. 3.8.2, there is a term $t_0$ such that $t' \rightarrow^*_c \widetilde{t}_0$ and $t \rightarrow^*_c t_0$. Hence $\widetilde{t} \rightarrow^*_c t_0 \rightarrow^*_c \downarrow (\widetilde{t}) = \downarrow t_0$. By induction hypothesis, $\downarrow (t_0) = \downarrow t_0$. Moreover $\downarrow t_0 \equiv \downarrow t$, so $(\downarrow t) = (\downarrow t_0) = \downarrow t_0 = \downarrow (\widetilde{t})$. □

Lemma 3.10. For any terms $t, t'$, if $t \rightarrow^*_c t'$ then there exists a term $u$ such that
\[
\downarrow t \rightarrow^*_c u \rightarrow^*_c t'.
\]

Proof: The proof proceeds by induction on $s(t)$, the structural measure of $t$ defined by
\[
s(x) = 1 \quad s(λx.t) = s(t) + 1 \quad s(\{θ\} \cdot t) = s(t) \times (s(θ) + 2)
\]
\[
s(c) = 1 \quad s(tu) = s(t) + s(u) \quad s(θ) = \sum_{c \in \text{dom}(θ)} s(θ_c)
\]
Notice that this measure decreases with the subterm relation but also with CASECASE reduction $s(\{θ\} \cdot \{φ\} \cdot u) > s(\{θ \circ φ\} \cdot u)$ for any $θ, φ, t$. For any term $s$ (or any case-binding $θ$), $s'$ (resp. $θ'$) represents a term (resp. a case-binding) such that $s \rightarrow^*_c s'$ (resp. $θ \rightarrow^*_c θ'$) for some $c \in \text{dom}(θ)$, and $θ_c = θ'_c$ for $c' \neq c$)

- If $t$ is an application, either $t = t_1t_2$ and $t' = t'_1t'_2$ (or $t' = t'_1t'_2$) and we conclude with induction hypotheses, or $t = (λx.t_1)t_2$ and $t' = t_1[x := t_2]$. In that case, $\downarrow t = (λx. \downarrow t_1) \downarrow t_2 \rightarrow^*_c (\downarrow t_1)[x := \downarrow t_2] \rightarrow^*_c (\downarrow t_1)[x := \downarrow t_2]$. Moreover, $\downarrow (\downarrow t_1)[x := \downarrow t_2] = \downarrow (t_1[x := t_2])$. Thus $\downarrow t \rightarrow^*_c (\downarrow t_1)[x := \downarrow t_2] \rightarrow^*_c \downarrow t'$.

- If $t$ is an abstraction, either $t = λx.t_0$ and $t' = λx.t'_0$ and we conclude with induction hypothesis, or $t = λx.t'x$ with $x \notin \text{fv}(t')$. In that case, $\downarrow t = λx. \downarrow t'x \rightarrow^*_c \downarrow t'$.

- If $t = \{θ\} \cdot x$, then $t' = \{θ'\} \cdot x$ and we conclude with induction hypothesis.

- If $t = \{θ\} \cdot c$, then either $t' = \{θ'\} \cdot c$ and we conclude with induction hypothesis, or $t' = θ_c$ and $\downarrow t = (\downarrow θ) \cdot c \rightarrow^*_c θ_c$.

- If $t = \{θ\} \cdot t_1t_2$, then either $t' = \{θ'\} \cdot t_1t_2$ and we conclude with induction hypothesis, or $t' = \{θ\} \cdot t_0$ with $t_1t_2 \rightarrow^*_c t_0$ or $t' = (\{θ\} \cdot t_1)t_2$.

In the second case, by induction hypothesis there is some $u_0$ such that $\downarrow t_1t_2 \rightarrow^*_c u_0 \rightarrow^*_c t_0$. Hence
\[
\downarrow t = (\downarrow θ) \cdot (\downarrow t_1t_2) \rightarrow^*_c (\downarrow θ) \cdot u_0 \rightarrow^*_c (\downarrow θ) \cdot t_0 \rightarrow^*_c (\downarrow θ) \cdot \downarrow t_0.
\]
Moreover, every sub-term of $\downarrow t'$ is in CASECASE normal form, so $\downarrow t' = \uparrow \{ \downarrow \theta \} \downarrow u_0 \rightarrow c_c^* \downarrow t'$. In the last case, $\downarrow t = \uparrow \{ \downarrow \theta \} \cdot (\downarrow t_1 \downarrow t_2)$, so

$$\downarrow t \rightarrow \lambda_C^* (\downarrow \theta) \downarrow t_1 \downarrow t_2 \rightarrow c_c^* \downarrow (\downarrow \theta) \downarrow t_1 \downarrow t_2 = \uparrow \{ \downarrow \theta \} \cdot t_1 \downarrow t_2.$$

- If $t = \{ \theta \} \cdot \lambda x. t_0$, ideim as previous case.
- If $t = \{ \theta \} \cdot \{ \phi \} \cdot t_0$, then either $t' = \{ \theta \} \cdot \{ \phi' \} \cdot t_0$, or $t' = \{ \theta \} \cdot \{ \phi \} \cdot t'_0$, or $t' = \{ \theta' \} \cdot \{ \phi \} \cdot t_0$.

In the first case, write $t_1 = \{ \theta \circ \phi \} \cdot t_0$ and $t'_1 = \{ \theta \circ \phi' \} \cdot t_0$. Remark that $s(t_1) < s(t)$ (since the structural measure decreases by CASECASE-reduction), and that $t_1 \rightarrow \lambda_C^* t'_1$. By induction hypothesis, there is some $u$ such that $\downarrow t_1 \rightarrow \lambda_C^* u \rightarrow c_c^* \downarrow t'_1$. Since $\downarrow t = \uparrow t_1$ and $\downarrow t' = \uparrow t'_1$ we are done.

In the second case, same method but with $t'_1 = \{ \theta \circ \phi \} \cdot t'_0$.

In the last case, write $t = \{ \theta \} \cdot \{ \phi_1 \} \cdots \{ \phi_k \} \cdot u_0$, where $u_0$ is not a case construct (thus $k \geq 1$). Then $\downarrow t = \{ \downarrow (\theta \circ \psi) \} \cdot u_0$, with $\psi = \phi_1 \circ (\cdots \circ \phi_k)$, and $\downarrow t' = \{ \downarrow (\theta' \circ \psi) \} \cdot u_0$ (since $((\theta \circ \phi_1) \circ \cdots) \circ \phi_k \rightarrow c_c^* \theta \circ \psi$).

Let us explicit $\downarrow t$ and $\downarrow t'$: $\downarrow t = \{ c \mapsto \downarrow \{ \theta \} \cdot \psi_c / c \in \text{dom}(\psi) \} \cdot u_0$

$$\downarrow t' = \{ c \mapsto \downarrow \{ \theta' \} \cdot \psi_c / c \in \text{dom}(\psi) \} \cdot u_0$$

Remark that $s(\{ \theta \} \cdot \psi_c) \leq s(t)$ (the structural measure decreases by CASECASE-reduction, and preserves the order of sub-term relation), and that $\{ \theta \} \cdot \psi_c \rightarrow \lambda_C^* \{ \theta' \} \cdot \psi_c$. Hence, by induction hypothesis, for each $c \in \text{dom}(\psi)$ there is a term $u_c$ such that $\downarrow \{ \theta \} \cdot \psi_c \rightarrow \lambda_C^* u_c \rightarrow c_c^* \downarrow \{ \theta' \} \cdot \psi_c$. Thus

$$\downarrow t \rightarrow \lambda_C^* u \rightarrow c_c^* \downarrow t' \quad \text{for} \quad u = \{ c \mapsto u_c / c \in \text{dom}(\psi) \} \cdot u_0 . \quad \square$$