Pure Patterns Type Systems

Gilles Barthe, Horatiu Cirstea, Claude Kirchner, Luigi Liquori

Pure Patterns Type Systems

Gilles Barthe Horatiu Cirstea Claude Kirchner Luigi Liquori

LORIA & INRIA 54506 Vandoeuvre-lès-Nancy BP 239 Cedex France

INRIA 06902 Sophia Antipolis BP 93 Cedex France

gbarthe@sophia.inria.fr, cirstea@loria.fr, ckirchner@loria.fr, lliquori@loria.fr

Abstract

We introduce a new framework of algebraic pure type systems in which we consider rewrite rules as lambda terms with patterns and rewrite rule application as abstraction application with built-in matching facilities.

This framework, that we call *"Pure Pattern Type Systems"*, is particularly well-suited for the foundations of programming (meta)languages and proof assistants since it provides in a fully unified setting higher-order capabilities and pattern matching ability together with powerful type systems.

We prove some standard properties like confluence and subject reduction for the case of a syntactic theory and under a classical (syntactical) restriction over the shape of patterns. We also conjecture the strong normalization of typable terms.

This work should be seen as a contribution to a formal connection between logics and rewriting, and a step towards new proof engines based on the Curry-Howard isomorphism.

Keywords

Lambda-calculus, Matching, Rewriting, Pure Type Systems, Curry-Howard, Logics.

1 Introduction

Pattern abstractions generalize λ-abstractions by binding structured expressions instead of variables, and are commonly used to compile case-expressions in functional programming languages [22] and to provide term calculi for sequent calculi [18]. For example, the pattern abstractions $\lambda 0.0$ and $\lambda \mathsf{succ}(N).N$ are used to compile the predecessor function $\lambda X.\ \mathsf{case}\ X\ \mathsf{of}\{0 \Rightarrow 0 \mid \mathsf{succ}(N) \Rightarrow N\}$, whereas the pattern abstraction $\lambda\langle X, Y\rangle$. X is used to encode the sequent derivation

$$\dfrac{\dfrac{A, B \vdash A}{A \wedge B \vdash A}}{\vdash (A \wedge B) \to A}$$

Extended abstractions generalize pattern-abstractions by binding arbitrary expressions instead of patterns, and are used in the rewriting-calculus [11] to provide a first-class account of rewrite rules and rewriting strategies. For example, generalized abstractions can be used to encode innermost rewriting strategies for term-rewriting systems. Furthermore, generalized abstractions correspond to a form of higher-order natural deduction, where (parts of) proof trees are discharged instead of assumptions.

Although such extended abstractions are a firmly grounded artifact both in logic and in programming language design and implementation, they lack established foundations. As a first step towards these foundations, we provide a framework for studying (type systems for) extended abstractions. Concretely, we start from a term calculus with generalized abstractions, which provides a very rich computational model that goes far beyond λ-calculus, and endorse it with a parameterized type system as in Pure Type Systems.

Pure Type Systems (**PTS**) [3, 4, 14, 15], provide a concise and unifying view of many typed λ-calculi and logics that occur in the literature, and offer a generic framework for the study of typed λ-calculi à la Church. Pure Type Systems have a well-understood theory and enjoy good meta-theoretical properties, and hence offer an appealing setting in which to specify the kernel of functional programming languages, see *e.g.* [23], and proof assistants based on the Curry-Howard isomorphism, see *e.g.* [4, 24].

In *Pure Pattern Type Systems* ($\mathbf{P}^2\,\mathbf{T}\,\mathbf{S}$), the framework presented here, the usual λ-abstraction of Pure Type Systems $\lambda X{:}A.B$ of type $\Pi X{:}A.C$ is replaced by a generalized abstraction $\lambda A{:}\Gamma.B$ of type $\Pi A{:}\Gamma.C$, where A is an arbitrary term (in jargon a *pattern*) and Γ is a context in which are declared the free variables of A, which are bound in $\lambda A{:}\Gamma.B$. Further, the application of an abstraction $\lambda A{:}\Gamma.B$ to a term C always "fires" and produces as result the term $[A \ll_\Gamma C].B$ which represents a *delayed matching abstraction*, *i.e.* an abstraction where the matching equation is "put on the stack". The matching abstraction will be evaluated (in case a matching solution exists) or delayed (in case no solution exists). If a solution σ exists, the delayed matching abstraction self-evaluates to $\sigma(B)$.

The main technical contribution of this paper is to show that, provided we restrict suitably the body of generalized abstractions, Pure Pattern Type Systems inherit all elementary properties of Pure Type Systems, and hence could serve as a basis for proof engines and (dependently) typed functional programming languages. It should be stressed however that, like Pure Type Systems, Pure Pattern Type Systems are conceived as core calculi, and that several extensions are required to obtain a realistic language.

We only mention a few extensions where generalized abstractions and matching play a prominent role:

- *Uniform failure.* Pure Pattern Type Systems feature failure of a matching equation as first-class citizens, via "stuck" matching abstractions, e.g. $[\mathsf{succ}(N) \ll_\Gamma 0].A$. In some circumstances however, it is useful to introduce a notion of uniform failure in which the failure is recorded, but the explanation of the failure (in the above case the attempt to match a successor against zero) is discarded. Such a uniform failure is captured by a new constant Null and by a rewrite rule

$$[A \ll_\Gamma B].C \to \mathsf{Null} \quad \text{if } A \checkmark B$$

  where $\checkmark$ is a suitably defined condition. All the results of this paper extend to Pure Pattern Type Systems, provided the relation $\checkmark$ is stable under substitution and reduction.

- *Matching based subtyping.* Matching yields a natural subtyping relation on types. For example, the function type $\Pi\mathsf{succ}(N){:}\Gamma.nat$ is a subtype of the function type $nat \to nat$, since its inhabitants are functions from $nat \to nat$ that get stuck when applied to 0. It is therefore natural to extend Pure Pattern Type Systems with a theory of subtyping derived from matching (see *e.g.* [27] for an extension of Pure Type Systems with subtyping). In the resulting formalism, the following typing rule for generalized abstractions is derivable

$$\frac{\Gamma, \Delta \vdash A : C \quad \Gamma, \Delta \vdash B : D\{A/X\} \quad \Gamma \vdash \Pi X{:}C.D : s}{\Gamma \vdash \lambda A{:}\Delta.B : \Pi X{:}C.D}$$

  Such a rule enhances the expressive power of Pure Pattern Type Systems. For example, it allows to encode case-expressions using the "," notation that allows to merge generalized abstractions.

- *Structures.* Pure Pattern Type Systems need to be extended with a form of dependent sum types for encoding structures. The traditional formalization of Σ-types can be generalized to expressions of the form $\Sigma P{:}\Gamma.B$, which represent pairs $\langle M, N\rangle$ such that M matches P and N is of type $\sigma(B)$ where σ is the solution of matching P against M. Such an extension bears some similarities and generalizes the notion of manifest field of modules and dependent record types, in which P is required to be a closed expression. In this setting also, matching-based subtyping seems relevant since it corresponds to the usual notion of subtyping between dependent record types with manifest fields [12, 19].

- *Matching modulo an equational theory.* The rewriting-calculus, upon which Pure Pattern Type Systems are built, takes as a parameter an equational theory modulo which matching is performed. We conjecture that all the results of this paper scale up to an extension of Pure Pattern Type Systems with matching theories, provided the matching theory only involves algebraic terms, i.e. terms build from variables and function symbols.

Beyond these extensions, Pure Pattern Type Systems raise two challenges for future work:

- *Extending the Curry-Howard Isomorphism.* The extension can be considered from the point of view of sequent calculi, deduction modulo, and natural deduction respectively. From the point of view of sequent calculi, it remains to investigate how Pure Pattern Type Systems can be used to extend previous results on term calculi for sequent calculi, and how their extension with matching theories can be used to provide suitable term calculi for deduction modulo. From the point of view of natural deduction, Pure Pattern Type Systems correspond to an extension of natural deduction where parts of proof trees are discharged instead of assumptions. To our best knowledge, such an extended form of natural deduction has not been considered previously, but it seems interesting to investigate whether such an extended natural deduction could find some applications in proof assistants, *e.g.* for transforming and optimizing proofs.

- *Extending Algebraic Type Systems.* One original motivation of the rewriting-calculus is to provide a setting in which not only term-rewriting systems, but also rewriting strategies for these systems can be encoded. It seems interesting to design a well-behaved extension of Pure Pattern Type Systems that has enough expressive power for encoding *Algebraic Type Systems*, which combine (first-order or higher-order) rewriting and λ-calculus, see *e.g.* [1, 5, 6, 7, 17], and rewriting strategies for Algebraic Type Systems. We believe that, if successful, such an endeavor could bear direct impact on the design of proof assistants based on type theory and rapid prototyping systems based on rewriting calculi.

The paper presents $\mathbf{P^2TS}$ and the main results without their proofs. The full version of the paper can be found at `www.loria.fr/~lliquori/PPTS.ps.gz` and contain all the proofs and extensive examples.

2 Syntax

The syntax of the $\mathbf{P^2TS}$ extends that of the classical Pure Type Systems [4] by considering structured terms and abstraction on patterns more elaborated than simple variables [11]. The contexts defining the types of the free variables of these patterns are given explicitly as part of the abstraction.

Notational Conventions. In this paper, we consider the meta-symbols λ (function abstraction), Π (product, or type abstraction), and "," (structure operator). The symbols $A, B, C, \ldots$ range over the set $\mathcal{T}$ of typed terms, the symbols $X, Y, Z, \ldots$ range over the infinite set $\mathcal{V}$ of variables, the symbols $a, b, c, f, \ldots$ range over the infinite set $\mathcal{K}$ of constants, the symbol s ranges over the set $\mathcal{S}$ of sorts. We assume $\mathcal{S} \subseteq \mathcal{K}$. The symbols $\alpha, \beta, \ldots$ range over $\mathcal{K}$ and $\mathcal{V}$. The symbols Γ, Δ range over the set $\mathcal{C}$ of type contexts.

All symbols can be indexed. The symbol $\equiv$ denotes syntactic identity of objects like terms or substitutions. The application of a constant function, say f, to a term A will be usually denoted by $f(A)$, following the algebraic "folklore"; this convention can be "curryfied" in order to denote a function taking multiple arguments. Constants can be also parameterized over a given theory.

As usual, we work modulo the "α-*convention*" of Church [9]; and modulo the "*hygiene-convention*" of Barendregt [2], *i.e.* free and bound variables have different names.

Syntax. The syntax of contexts and pseudo-terms of the $\mathbf{P^2\,T\,S}$ is defined as follows:

$$\mathcal{C} \quad ::= \quad \emptyset \mid \mathcal{C}, \mathcal{V}{:}\mathcal{T} \mid \mathcal{C}, \mathcal{K}{:}\mathcal{T}$$

$$\mathcal{T} \quad ::= \quad \mathcal{V} \mid \mathcal{K} \mid \lambda\mathcal{T}{:}\mathcal{C}.\mathcal{T} \mid \Pi\mathcal{T}{:}\mathcal{C}.\mathcal{T} \mid [\mathcal{T} \ll_{\mathcal{C}} \mathcal{T}].\mathcal{T} \mid \mathcal{T}\ \mathcal{T} \mid \mathcal{T}, \mathcal{T}$$

The main intuition behind this syntax is that the term $\lambda A{:}\Gamma.B$ (or $\Pi A{:}\Gamma.B$, or $[A \ll_\Gamma C].B$) is an *abstraction* with pattern A, body B and context Γ (and possibly an argument C). The free variables of the pattern A bind the corresponding variables in the body B and the context Γ. The types of the free variables of the pattern A are declared in Γ and, by abuse of notation, we will sometime write $\lambda\alpha{:}D.B$ (resp. $\Pi\alpha{:}D.B$, or $[\alpha \ll_D C].B$) instead of a more redundant $\lambda\alpha{:}(\alpha{:}D).B$ (resp. $\Pi\alpha{:}(\alpha{:}D).B$, or $[\alpha \ll_{\alpha:D} C].B$).

Just to support intuition, we mention here that the application of an abstraction $\lambda A{:}\Gamma.B$ to a term C always "fires" and produces as result the term $[A \ll_\Gamma C].B$ which represents a *delayed matching abstraction*, *i.e.* an abstraction where the matching equation is "put on the stack". The matching abstraction will be evaluated (in case a matching solution exists) or delayed (in case no solution exists at this stage of the evaluation). If a solution exists, the delayed matching abstraction self-evaluates to $\sigma(B)$, where σ is the solution of the matching between A and C.

Finally, as a sort of syntactic sugar, and directly inspired from previous works on the Rho-calculus [10], all the terms can be grouped together into *structures* built using the meta-symbol ",".

To resume, there are three abstractors in $\mathbf{P^2\,T\,S}$:

- the λ abstractor, used in λ-calculus to bind ordinary variables, that binds here over patterns;
- the Π abstractor that binds over patterns too;
- the new $[\_ \ll \_]$ abstractor, that binds over (delayed) matching equations (in fact, over the patters from the left-hand sides of matching equations).

Example 2.1 *Some simple examples of terms are:*

- *the term* $(\lambda f(X){:}\Gamma.X)f(a)$ *where* $\Gamma \triangleq X{:}i$ *"simulates" the application of a rewrite rule* $f(X) \to X$ *to the term* $f(a)$*, returning* a*;*
- *the term* $(\lambda(\lambda Z{:}i.(ZX)Y){:}\Gamma.X)$*, where* Γ *defines the corresponding types of* X *and* Y*, represents the projection on the first element of a pair and its application to the term* $(\lambda Z{:}i.(ZA)B))$ *evaluates to* A*.*
- *the term* $(1, 2, 3)$ *denotes a simple triple of integers;*
- *the term* $(\lambda(X, Y, Z){:}\Gamma.X)$*, where* $\Gamma \triangleq X{:}i, Y{:}i, Z{:}i$ *is a simple projection function on triples of integers: if we apply this function to* $(1, 2, 3)$*, then we obtain as result* 1*;*
- *the term* $(\lambda a{:}A.b, \lambda c{:}C.d)$ *denotes a record with two fields,* a *and* c*.*

3 $\mathbf{P^2\,T\,S}$: Matching and Substitutions

As already mentioned, in $\mathbf{P^2\,T\,S}$ we deal with abstractions on patterns and their application and thus, computing the substitutions which solves the matching from a pattern A to a subject B is a fundamental element of the framework. We focus on syntactic matching and we revisit the corresponding notions and algorithms in the context of $\mathbf{P^2\,T\,S}$.

Substitutions. We adapt the classical notion of simultaneous substitution application to deal with the new forms of abstraction introduced in the $\mathbf{P^2\,T\,S}$.

Definition 3.1 (Substitutions) *A (non-empty) substitution* σ *is of the form*

$$\{A_1/X_1 \ldots A_m/X_m\}$$

and the empty substitution is denoted by σ_{ID}*. The application of a substitution* σ *to a term* B*, denoted by* $\sigma(B)$ *(or* $B\sigma$*), is defined as follows:*

$$\begin{array}{lcl}
\sigma_{ID}(A) & \triangleq & A \\
\sigma(s) & \triangleq & s \\
\sigma(a) & \triangleq & a \\
\sigma(X_i) & \triangleq & \begin{cases} A_i & \text{if } X_i \in Dom(\sigma) \\ X_i & \text{otherwise} \end{cases} \\
\sigma(\lambda A{:}\Gamma.B) & \triangleq & \lambda A{:}\sigma(\Gamma).\sigma(B) \\
\sigma(\Pi A{:}\Gamma.B) & \triangleq & \Pi A{:}\sigma(\Gamma).\sigma(B) \\
\sigma([A \ll_\Gamma B].C) & \triangleq & [A \ll_{\sigma(\Gamma)} \sigma(B)].\sigma(C) \\
\sigma(AB) & \triangleq & \sigma(A)\ \sigma(B) \\
\sigma(A, B) & \triangleq & \sigma(A), \sigma(B) \\
\sigma(\emptyset) & \triangleq & \emptyset \\
\sigma(\Gamma, \alpha{:}A) & \triangleq & \sigma(\Gamma), \alpha{:}\sigma(A)
\end{array}$$

Recall that we work modulo the α*-convention* and thus, when applying a substitution to an abstraction we know that the free variables of the corresponding abstracted pattern do not belong to the domain of the substitution.

Matching Equations and Solutions. In general, syntactic matching is formally defined as follows:

Definition 3.2 (Matching Equations and Solutions)

1. *A match equation is a formula of the form* $A \ll_\Gamma B$*;*
2. *A matching system* $\mathsf{T} \triangleq \bigwedge_{j=0\ldots n} A_j \ll_{\Gamma_j} B_j$ *is a conjunction of match equations, where* $\wedge$ *is associative, commutative and idempotent;*
3. *A matching system* T *is "successful" if it is empty or*
   (a) *has the shape* $\bigwedge_{j=0\ldots n} X_j \ll_{\Gamma_j} A_j \ \bigwedge_{i=0\ldots m} a_i \ll_{\Gamma_i} a_i$*;*
   (b) *for all* $i \neq j$*, we have* $X_i \neq X_j$*;*
   (c) *for all* j *such that* $X_j \in Dom(\Gamma_j)$*, we have* $A_j \equiv X_j$*;*
4. *A substitution* $\sigma_{\mathsf{T}} = \{A_1/X_1 \cdots A_n/X_n\}$ *is the solution of a successful matching system* T*;*

$$
\begin{array}{lll}
(Lambda) & (\lambda A{:}\Delta.B_1) \ll_\Gamma (\lambda A{:}\Delta.B_2) & \leadsto \quad B_1 \ll_{\Gamma,\Delta} B_2 \\
(Prod) & (\Pi A{:}\Delta.B_1) \ll_\Gamma (\Pi A{:}\Delta.B_2) & \leadsto \quad B_1 \ll_{\Gamma,\Delta} B_2 \\
(Delay) & [A \ll_\Delta C_1].B_1 \ll_\Gamma [A \ll_\Delta C_2].B_2 & \leadsto \quad B_1 \ll_{\Gamma,\Delta} B_2 \wedge C_1 \ll_\Gamma C_2 \\
(Appl) & (A_1 B_1) \ll_\Gamma (A_2 B_2) & \leadsto \quad A_1 \ll_\Gamma A_2 \wedge B_1 \ll_\Gamma B_2 \\
(Struct) & (A_1, B_1) \ll_\Gamma (A_2, B_2) & \leadsto \quad A_1 \ll_\Gamma A_2 \wedge B_1 \ll_\Gamma B_2
\end{array}
$$

Figure 1: Matching Reduction System

$$
\begin{array}{llll}
(\rho) & (\lambda A{:}\Gamma.B)C & \to_\rho & [A \ll_\Gamma C].B \\
(\sigma) & [A \ll_\Gamma C].B & \to_\sigma & \sigma_{(A \ll_\Gamma C)}(B) \\
(\delta) & (A, B)\, C & \to_\delta & A\, C, B\, C
\end{array}
$$

Figure 2: Top-level Rules of the $\mathbf{P^2 T S}$

5. *Let* T *be such that* $\mathsf{T} \leadsto^* \mathsf{T}'$ *with* T' *successful, we define the substitution* σ_{T}*, solution of* T*, as* $\sigma_{\mathsf{T}'}$.

The matching substitution solving a matching system T can be computed by the *matching reduction system* of Figure 1.

Starting from a matching equation $A \ll_\Gamma B$, the application of this rule set obviously terminates and either leads to an unsuccessful matching system in which case we say that the matching has failed or a substitution $\sigma_{(A \ll_\Gamma B)}$ such that $\sigma_{(A \ll_\Gamma B)}(A) \equiv B$ is exhibited.

This set of rules could be easily extended to matching modulo commutativity (it is decidable too but the number of matches can then be exponential in the size of the initial problem). It is also decidable for associativity-commutativity [16] but in the general case matching could be as difficult as unification [8].

4 Dynamic Semantics

Top-level Rules. The top-level rules are presented in Figure 2. The central idea of the (ρ) rule of the calculus is that the application of a term $\lambda A{:}\Gamma.B$ to a term C, reduces to the delayed matching abstraction $[A \ll_\Gamma C].B$, while the application of the (σ) rule consists in computing the matching equation $A \ll_\Gamma C$, and applying the obtained result to the the term B. The rule (δ) deals with the distributivity of the application on the structures built with the "," constructor.

It is important to remark that if A is a variable, then the subsequent combination of (ρ) and (σ) rules corresponds exactly to the (β) rule of the λ-calculus, and variable manipulations in substitutions are handled externally, using α-conversion and Barendregt's hygiene convention if necessary. The top-level rules can be point-wise extended to contexts.

One-step, Many-steps, Congruence. The next definition introduces the classical notions of one-step, many-steps, and congruence relation of $\to_{\rho\sigma\delta}$.

Definition 4.1 (One-step, Multi-steps, Congruence) *Let* $\mathsf{Ctx}[-]$ *be any context in* $\mathcal{T}$ *with a single hole, and let* $\mathsf{Ctx}[A]$ *be the result of filling the hole with the term* A*;*

1. *the one-step evaluation* $\mapsto_{\rho\sigma\delta}$ *is defined by the following inference rules:*

$$\frac{A \to_{\rho\sigma\delta} B}{\mathsf{Ctx}[A] \mapsto_{\rho\sigma\delta} \mathsf{Ctx}[B]} \quad (\mathsf{Ctx}[-])$$

   where $\to_{\rho\sigma\delta}$ *denotes one of the top-level rules of the* $\mathbf{P^2 T S}$*;*

2. *the multi-step evaluation* $\mapsto\!\!\!\!\to_{\rho\sigma\delta}$ *is defined as the reflexive and transitive closure of* $\mapsto_{\rho\sigma\delta}$*;*

3. *the congruence relation* $=_{\rho\sigma\delta}$ *is the symmetric closure of* $\mapsto\!\!\!\!\to_{\rho\sigma\delta}$.

The above rules can be point-wise extended to contexts and substitutions.

Two Simple (Untyped) Examples In order to illustrate the behavior of the top-level rules, we present in Figure 3 the reduction of two terms using different evaluation strategies (outermost *vs.* innermost) and yielding in the first case a "successful" result (*i.e.* containing no delayed matching abstraction) and in the second one an "unsuccessful" one. We underline redexes to be reduced.

It is worth noticing that the term $[3 \ll 4].3$ represents "de facto" a computation failure, which can be read as follows: "an exception occurred due to a matching failure". The capability of $\mathbf{P^2 T S}$ to record failures is directly inherited from previous versions of the rewriting-calculus, where a special symbol Null was explicitly denoting computation failures. This kind of "exception" could be catch by a suitable handler, *e.g.* $(\lambda([3 \ll 4].X).\texttt{exc\_handler})(\texttt{protected\_body})$. An exception handling mechanism for the rewriting-calculus has been already studied [13] and we plan to analyze this topic in a typed framework in a future paper.

The Rigid Pattern Condition. When no restrictions are imposed neither on the term formation nor on the reduction strategy, the classical properties, and in particular the confluence, are not valid. Therefore, a suitable condition Cond should be used in order to ensure that $\mathbf{P^2 T S}$ are well-behaved extensions of plain **PTS**.

The shape of patterns in $\mathbf{P^2 T S}$ abstractions is essential in order to avoid bizarre non-confluent reductions and thus, in order to recover the good properties, Cond should ensure that these patterns satisfy certain properties. For the scope of this paper we use for Cond the *Rigid Pattern Condition* (RPC) that was firstly formalized in [26] but we conjecture that less restrictive conditions can lead to similar results.

A successful computation

- $\underline{(\lambda f(X).(\lambda 3.3)X)\ f(3)} \mapsto_\rho \underline{[f(X) \ll f(3)].((\lambda 3.3)\ X)} \mapsto_\sigma ((\lambda 3.3)\ X)\{3/X\} \triangleq \underline{(\lambda 3.3)\ 3} \mapsto_\rho \underline{[3 \ll 3].3} \mapsto_\sigma 3\{\ \} \triangleq 3$
- $(\lambda f(X).\underline{(\lambda 3.3)X})\ f(3) \mapsto_\rho \underline{(\lambda f(X).[3 \ll X].X)\ f(3)} \mapsto_\rho \underline{[f(X) \ll f(3)].([3 \ll X].X)} \mapsto_\sigma ([3 \ll X].X)\{3/X\} \triangleq \underline{[3 \ll 3].3} \mapsto_\sigma 3\{\ \} \triangleq 3$

An unsuccessful computation

- $\underline{(\lambda f(X).(\lambda 3.3)X)\ f(4)} \mapsto_\rho \underline{[f(X) \ll f(4)].((\lambda 3.3)\ X)} \mapsto_\sigma ((\lambda 3.3)\ X)\{4/X\} \triangleq \underline{(\lambda 3.3)\ 4} \mapsto_\rho [3 \ll 4].3$
- $(\lambda f(X).\underline{(\lambda 3.3)X})\ f(4) \mapsto_\rho \underline{(\lambda f(X).[3 \ll X].3)\ f(4)} \mapsto_\rho \underline{[f(X) \ll f(4)].([3 \ll X].3)} \mapsto_\sigma ([3 \ll X].3)\{4/X\} \triangleq [3 \ll 4].3$

Figure 3: Two Simple Evaluations in $\mathbf{P^2TS}$

The *Rigid Pattern Condition* is sufficient for obtaining the "diamond property" of the (parallel) reduction and hence the confluence of the reduction in $\mathbf{P^2TS}$. As well explained in [26], the rigid pattern condition does not characterize the shape of patterns, but syntactic characterizations of the terms satisfying this condition can be given.

Our terms have now the following (restricted) new shape with the patterns $\mathcal{P} \subset \mathcal{T}$:

$$\mathcal{T} ::= \lambda \mathcal{P}{:}\Gamma.\mathcal{T} \mid \Pi \mathcal{P}{:}\Gamma.\mathcal{T} \mid [\mathcal{P} \ll_{\mathcal{C}} \mathcal{T}].\mathcal{T} \mid \ldots \text{ as before } \ldots$$

In what follows, and in order to give an intuition to the reader, we anticipate a formal definition of the RPC *w.r.t.* the patterns $\mathcal{P}$, and we also present a "simple" characterization of $\mathcal{P}$ which satisfies RPC. The RPC condition is intensively used in all metatheory.

We introduce the RPC which turns out to be sufficient to prove that the parallel reduction $\Rightarrow_{\rho\sigma\delta}$ (defined in Section 6) satisfies the diamond property.

Definition 4.2 (Rigid Pattern Condition (RPC) [26]) *Let $\mathcal{P} \subset \mathcal{T}$, $\mathcal{P}$ satisfies* RPC *if for all $A \in \mathcal{P}$ and for all $B, \vec{C} \in \mathcal{T}$, there exists $\vec{D} \in \mathcal{T}$ such that*

$$A\{\vec{C}/\vec{X}\} \Rightarrow_{\rho\sigma\delta} B \text{ implies } \vec{C} \Rightarrow_{\rho\sigma\delta} \vec{D} \text{ and } B \equiv A\{\vec{D}/\vec{X}\}.$$

Remark 4.1 *1. $\mathcal{V}$ obviously satisfies* RPC*, but $\mathcal{T}$ do not;*

2. *The operational semantics and the type systems remain unchanged when taking into account the (now restricted) shape of patterns;*
3. *From now on we deal only with terms containing patterns which satisfy* RPC*;*
4. *The* RPC *is only used to prove the diamond property of the parallel reduction.*

A Characterization of RPC. We now characterize an "honest" subset $\mathcal{P}$ of $\mathcal{T}$ which properly contains $\mathcal{V}$ and satisfies RPC.

Definition 4.3 (A Simple Characterization of $\mathcal{P} \subset \mathcal{T}$) *Let $NF(\rho\sigma\delta)$ be the set of terms that cannot be reduced by one of the rules of $\mathbf{P^2TS}$, we define*

$$\mathcal{P} \triangleq \{A \in NF(\rho\sigma\delta) \mid A \text{ is "linear" with no "active" variables}\}$$

One can easily verify that the above characterization of $\mathcal{P} \subset \mathcal{T}$ satisfies RPC *and properly contains $\mathcal{V}$. As well remarked (still) in [26] the set $\mathcal{P}$ is not the maximal set for which our $\mapsto_{\rho\sigma\delta}$ is confluent, e.g. the set*

$\mathcal{P}_\Omega \triangleq \mathcal{P} \cup \{\Omega\}$ *where* $\Omega \triangleq (\lambda X.XX)(\lambda X.XX)$ *also satisfies* RPC.

Remark 4.2 ([26]) *"... RPC does not prevent reduction from taking place inside patterns. However, one easily verifies that if $A \in \mathcal{P}$ and $A \mapsto_{\rho\sigma\delta} B$, then $A \equiv B$ ...". As usual, the type system is delegated to "block" (i.e. make not typable) those terms.*

5 Static Semantics

This section presents the type system underneath the $\mathbf{P^2TS}$. The system allows one to assign types to terms of the calculus.

Definition of $\mathbf{P^2TS}$. As for the **PTS**, the specification of a $\mathbf{P^2TS}$ consists of a quadruple

$$\mathsf{P^2TS} \triangleq (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$$

where:

- $\mathcal{S}$ is a subset of $\mathcal{C}$ and contains the *sorts*;
- $\mathcal{A}$ is a set of axioms of the form $s_1 : s_2$, with $s_1, s_2 \in \mathcal{S}$;
- $\mathcal{R}$ is a set of rules of the form (s_1, s_2, s_3) with $(s_1, s_2, s_3) \in \mathcal{S}$;
- $\mathcal{P}$ is a subset of $\mathcal{T}$, which satisfies RPC.

Type System. The notion of type derivation in $\mathbf{P^2TS}$ involves two judgments of the shape:

$$\Gamma \vdash_{\mathsf{P^2TS}} A : B \quad \text{and} \quad \vdash_{\mathsf{P^2TS}} A \ll_\Gamma B : ok$$

In the following, we use $\vdash$ instead of $\vdash_{\mathsf{P^2TS}}$. Intuitively, the first judgment type-checks $\mathbf{P^2TS}$ terms, while the second deals with the type-checking of matching equations. The latter judgment can be point-wise extended to contexts.

A "Guided Tour" on Type Rules. The type system is defined by rule schemas presented in Figure 4, while the typed matching equation system is defined in Figure 5. In the following, we briefly comment the typing rules and motivate by an example the need for the typed matching.

- The $(Axioms), (Start), (Weak)$ rules are standard and need no comments;
- The $(Conv)$ rule is relaxed *w.r.t.* the usual one can find in usual **PTS** (where normally we have $\Gamma \vdash C : s$);

$$\frac{(s_1, s_2) \in \mathcal{A}}{\emptyset \vdash s_1 : s_2} \quad (Axioms)$$

$$\frac{\Gamma \vdash A : C \quad \Gamma \vdash B : C}{\Gamma \vdash A, B : C} \quad (Struct)$$

$$\frac{\Gamma \vdash A : s \quad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha{:}A \vdash \alpha : A} \quad (Start)$$

$$\frac{\Gamma \vdash A : B \quad \Gamma \vdash C : s \quad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha{:}C \vdash A : B} \quad (Weak)$$

$$\frac{\Gamma \vdash A : B \quad \Gamma \vdash C : D \quad B =_{\rho\partial} C}{\Gamma \vdash A : C} \quad (Conv)$$

$$\frac{\Gamma, \Delta \vdash B : C \quad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash \lambda A{:}\Delta.B : \Pi A{:}\Delta.C} \quad (Abs)$$

$$\frac{\begin{array}{l}\Gamma, \Delta \vdash A : C \quad (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma, \Delta \vdash C : s_1 \quad \Gamma, \Delta \vdash B : s_2\end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3} \quad (Prod)$$

$$\frac{\begin{array}{ll}\Gamma \vdash B : D & \Gamma, \Delta \vdash C : D \\ \Gamma, \Delta \vdash A : E & \vdash C \ll_{\Gamma,\Delta} B : ok\end{array}}{\Gamma \vdash [C \ll_\Delta B].A : [C \ll_\Delta B].E} \quad (Subst)$$

$$\frac{\begin{array}{lll}\Gamma \vdash B : E & & \Gamma, \Delta \vdash C : E \\ \Gamma \vdash A : \Pi C{:}\Delta.D & & \vdash C \ll_{\Gamma,\Delta} B : ok\end{array}}{\Gamma \vdash A\ B : [C \ll_\Delta B].D} \quad (Appl)$$

Figure 4: The Type Rules for $\mathbf{P^2\,T\,S}$

- The (*Struct*) rule says that a structure A, B, where $A : C$ and $B : C$ can be typed with type C, hence forcing all subterms to be of the same type;
- The (*Abs*) rule deals with lambda abstractions in which we bind over (non trivial) patterns instead of variables; note that the context Δ is "charged" in the first premise in order to type-check the body of the function B;
- The (*Prod*) rule deals with product types in which we bind over (non trivial) patterns instead of variables; also in this rule, the context Δ is "charged" in all premises in order to type-check every subterm, hence "selecting" a suitable triple $(s_1, s_2, s_3) \in \mathcal{R}$;
- The (*Subst*) rule deals with terms in which a *delayed matching equation* occurs *hard-coded* into the term (and its type); this rule is essentially needed to ensure the well-typedness of terms leading to matching failures (*e.g.* $(\lambda 3{:}int.3)\ X$) and ensure the subject reduction property for the top-level rule (ρ), and (σ);
- The (*Appl*) rule deals with applications where the type of the function is a product type with (non-trivial) patterns; note that the context Δ is "charged" in order to type-check the pattern C of the function A. Moreover, the resulting type in the conclusion is a delayed matching abstraction "type" .

On Terms

$$\frac{\Gamma \vdash \alpha : B \quad \Gamma \vdash A : B}{\vdash \alpha \ll_\Gamma A : ok} \quad (\alpha)$$

$$\frac{\begin{array}{l}\vdash A_1 \ll_\Gamma A_2 : ok \\ \vdash B_1 \ll_\Gamma B_2 : ok \quad \diamond \in \{,\ @\}\end{array}}{\vdash A_1 \diamond B_1 \ll_\Gamma A_2 \diamond B_2 : ok} \quad (\diamond_1)$$

$$\frac{\vdash B_1 \ll_{\Gamma,\Delta} B_2 : ok \quad \diamond \in \{\lambda\ \Pi\}}{\vdash \diamond A{:}\Delta.B_1 \ll_\Gamma \diamond A{:}\Delta.B_2 : ok} \quad (\diamond_2)$$

$$\frac{\vdash B_1 \ll_{\Gamma,\Delta} B_2 : ok \quad \vdash C_1 \ll_\Gamma C_2 : ok}{\vdash [A \ll_\Delta C_1].B_1 \ll_\Gamma [A \ll_\Delta C_2].B_2 : ok} \quad (\diamond_3)$$

$$\frac{\vdash A \ll_\Gamma B_1 : ok \quad B_1 \mapsto\!\!\!\!\rightarrow_{\rho\partial} B_2 \quad A \in \mathcal{P}}{\vdash A \ll_\Gamma B_2 : ok} \quad (Red)$$

Figure 5: The Type Matching Rules for $\mathbf{P^2\,T\,S}$

The typed matching judgment in the rules (*Subst*) and (*Appl*) guarantees the well-typedness of the matching equation. The lack of this test could lead to the failure of subject reduction as shown in the following example.

Example 5.1 (Failure of Subject Reduction) *The term* $(\lambda(X_1 Y_1){:}\Delta.Y_1)(X_2 Y_2)$, *where* $\Delta \triangleq X_1{:}\Pi Z{:}a.b, Y_1{:}a$ *can be typed with* b *in the context* $\Gamma \triangleq a, b, c{:}*, X_2{:}\Pi Z{:}c.b, Y_2{:}c$ *when the type system does not check the well-typedness of the matching equations. This term can be reduced to* Y_2 *that has the type* c *in the previous context.*

The type matching rules are fairly easy: they guarantee the conformance of types of constants and variables when dealing with matching equations. The only intriguing rule is the rule (*Red*) which imposes in the side condition $A \in \mathcal{P}$ (hence RPC), hence guaranteeing the good behavior of patterns.

One can notice that the pattern used in the Example 5.1 does not satisfy RPC. Indeed, we conjecture that the typed matching judgment is not needed when RPC is imposed on patterns, but is useful for further extensions of the framework with less restrictive conditions on the patterns.

The Graph Dependency. In order to summarize all the congruence / theories / matching / type systems, Figure 6 shows all potential dependencies between them: in this figure, $\spadesuit \rightarrow \clubsuit$ means that $\spadesuit$ "make use" (or "depends") on $\clubsuit$.

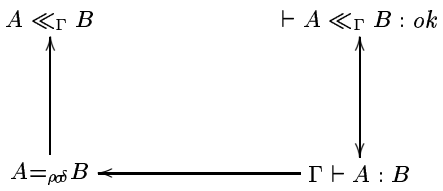


Figure 6: Dependency Graph

$\mathbf{P^2\,T\,S}$ and $\lambda^{\ll}$-cube. As it is well known in the literature [4], the type systems depicted in the famous λ-cube are a special case of **PTS** when $s_2 \equiv s_3$, *i.e.*

$$(*, \Box, \Box) \in \mathcal{R}$$

In perfect symmetry, we can depict 8 type systems which we call $\lambda^{\ll}$*-cube with pattern matching* ($\lambda^{\ll}$-cube for short). Figure 7 collects into a classical table the 8 type systems, and their underpinning logic systems, where $\mathcal{S} = \{*, \Box\}$, and $\mathcal{R} = \{(*, \Box, \Box)\}$. In the following, (s_1, s_2) stands for (s_1, s_2, s_2). As in the plain λ-cube, well-know logics underpinning the type systems drawn in the $\lambda^{\ll}$-cube are classified in Figure 7.

Representing Logics and Data-types. Our $\mathbf{P^2\,T\,S}$ are a conservative extension of ordinary **PTS**. Therefore, the Curry-Howard isomorphism on type systems and logics naturally applies, and we depict in Figure 7 the $\lambda^{\ll}$-cube, and the corresponding *Logic*-cube which conforms to the classical slogan:

Proof-as-($\lambda^{\ll}$)-terms and propositions-as-types

We terminate the presentation of our $\mathbf{P^2\,T\,S}$ with a typing derivation showed in Figure 8.

6 Metatheory

This section collects all the meta-theory of $\mathbf{P^2\,T\,S}$: more precisely, we study confluence, typability, and some conjectures about normalization.

We introduce the notion of free variables that will be used in the subsequent sections.

Definition 6.1 (Free Variables FV) *The set of free variables is inductively defined on terms as follows*

$$\begin{array}{lcl}
FV(s) & \triangleq & \emptyset \\
FV(a) & \triangleq & \emptyset \\
FV(X) & \triangleq & \{X\} \\
FV(\lambda A{:}\Gamma.B) & \triangleq & FV(\Gamma) \cup (FV(B) \setminus FV(A)) \\
FV(\Pi A{:}\Gamma.B) & \triangleq & FV(\Gamma) \cup (FV(B) \setminus FV(A)) \\
FV([A \ll_\Gamma C].B) & \triangleq & FV(\Gamma) \cup (FV(B) \setminus FV(A)) \cup FV(C) \\
FV(AB) & \triangleq & FV(A) \cup FV(B) \\
FV(A, B) & \triangleq & FV(A) \cup FV(B)
\end{array}$$

6.1 $\mathbf{P^2\,T\,S}$: Confluence.

Let us define a "comfortable" notion of parallel reduction for our $\mathbf{P^2\,T\,S}$.

Definition 6.2 (Parallel Reduction) *The parallel reduction, denoted by $\Rightarrow_{\rho\sigma\delta}$, is inductively defined as follows:*

(Par_1) $\alpha \Rightarrow_{\rho\sigma\delta} \alpha$

(Par_2) $\lambda A{:}\Gamma.B_1 \Rightarrow_{\rho\sigma\delta} \lambda A{:}\Gamma.B_2$ *if* $B_1 \Rightarrow_{\rho\sigma\delta} B_2$

(Par_3) $\Pi A{:}\Gamma.B_1 \Rightarrow_{\rho\sigma\delta} \Pi A{:}\Gamma.B_2$ *if* $B_1 \Rightarrow_{\rho\sigma\delta} B_2$

(Par_4) $A_1B_1 \Rightarrow_{\rho\sigma\delta} A_2B_2$ *if* $A_1 \Rightarrow_{\rho\sigma\delta} A_2$ *and* $B_1 \Rightarrow_{\rho\sigma\delta} B_2$

(Par_5) $A_1, B_1 \Rightarrow_{\rho\sigma\delta} A_2, B_2$ *if* $A_1 \Rightarrow_{\rho\sigma\delta} A_2$ *and* $B_1 \Rightarrow_{\rho\sigma\delta} B_2$

(Par_6) $(\lambda A{:}\Gamma.B_1)C_1 \Rightarrow_{\rho\sigma\delta} [A \ll_\Gamma C_2].B_2$ *if* $B_1 \Rightarrow_{\rho\sigma\delta} B_2$ *and* $C_1 \Rightarrow_{\rho\sigma\delta} C_2$

(Par_7) $[A \ll_\Gamma B_1].C_1 \Rightarrow_{\rho\sigma\delta} [A \ll_\Gamma B_2].C_2$ *if* $B_1 \Rightarrow_{\rho\sigma\delta} B_2$ *and* $C_1 \Rightarrow_{\rho\sigma\delta} C_2$

(Par_8) $[A \ll_\Gamma B_1].C_1 \Rightarrow_{\rho\sigma\delta} \sigma_{(A \ll_\Gamma B_2)}(C_2)$ *if* $B_1 \Rightarrow_{\rho\sigma\delta} B_2$ *and* $C_1 \Rightarrow_{\rho\sigma\delta} C_2$

(Par_9) $(A_1, B_1)C_1 \Rightarrow_{\rho\sigma\delta} A_2C_2, B_2C_2$ *if* $A_1 \Rightarrow_{\rho\sigma\delta} A_2$ *and* $B_1 \Rightarrow_{\rho\sigma\delta} B_2$ *and* $C_1 \Rightarrow_{\rho\sigma\delta} C_2$

The rules $(Par_1), \ldots, (Par_5)$ indicate that the relation $\Rightarrow_{\rho\sigma\delta}$ includes the identity on $\lambda^{\ll}$-terms, *i.e.* $A \Rightarrow_{\rho\sigma\delta} A$ holds for all $A \in \mathcal{T}$. Rules $(Par_6), \ldots, (Par_9)$ deal with reductions. Intuitively, $A \Rightarrow_{\rho\sigma\delta} B$ means that B is obtained from A, by simultaneous contractions of some $(\rho\sigma\delta)$-redexes possibly overlapping one each other.

Lemma 6.1 (Permutation) *1. If $X \notin FV(C)$, then $A\{B/X\}\{C/Y\} \equiv A\{C/Y\}\{B\{C/Y\}/X\}$;*

2. If $A =_{\rho\sigma\delta} B$, then $A\{C/X\} =_{\rho\sigma\delta} B\{C/X\}$.

Lemma 6.2 (Relations) $\mapsto_{\rho\sigma\delta} \subseteq \Rightarrow_{\rho\sigma\delta} \subseteq \twoheadmapsto_{\rho\sigma\delta}$.

For the parallel reduction, the following lemma holds.

Lemma 6.3 (Parallel) *If $A \Rightarrow_{\rho\sigma\delta} B$ and $\vec{C} \Rightarrow_{\rho\sigma\delta} \vec{D}$, then $A\{\vec{C}/\vec{X}\} \Rightarrow_{\rho\sigma\delta} B\{\vec{D}/\vec{X}\}$.*

Now, to prove the Church-Rosser Theorem for $\twoheadmapsto_{\rho\sigma\delta}$, it is sufficient to show the *Diamond Property* $\diamond$ for $\Rightarrow_{\rho\sigma\delta}$, *i.e.*:

Theorem 6.1 (Diamond Property for $\Rightarrow_{\rho\sigma\delta}$) *If $A \Rightarrow_{\rho\sigma\delta} B$ and $A \Rightarrow_{\rho\sigma\delta} C$, then there exists D, such that $B \Rightarrow_{\rho\sigma\delta} D$, and $C \Rightarrow_{\rho\sigma\delta} D$.*

Indeed, we can adapt the stronger statement from [25] to our $\mathbf{P^2\,T\,S}$, as follows:

Theorem 6.2 (Strong Church-Rosser for $\Rightarrow_{\rho\sigma\delta}$) *If $A \Rightarrow_{\rho\sigma\delta} B$, then there exists a term $A^\diamond$, depending only on A, such that $B \Rightarrow_{\rho\sigma\delta} A^\diamond$. (Intuitively, the term $A^\diamond$ is obtained by contracting all the redexes existing in A simultaneously.)*

System	Axioms			
$\lambda^{\ll}{\to}$	$(*,*)$			
$\lambda^{\ll}2$	$(*,*)$	$(\Box,*)$		
$\lambda^{\ll}\underline{\omega}$	$(*,*)$			$(\Box,\Box)$
$\lambda^{\ll}\omega$	$(*,*)$		$(*,\Box)$	$(\Box,\Box)$
$\lambda^{\ll}P$	$(*,*)$		$(*,\Box)$	
$\lambda^{\ll}P2$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	
$\lambda^{\ll}P\underline{\omega}$	$(*,*)$		$(*,\Box)$	$(\Box,\Box)$
$\lambda^{\ll}P\omega$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	$(\Box,\Box)$

Name	Logics
PROP	proposition logic
PROP2	second-order PROP
PROP$\underline{\omega}$	weakly higher-order PROP
PROPω	higher-order PROP
PRED	predicate logic
PRED2	second-order PRED
PRED$\underline{\omega}$	weakly higher-order PRED
PREDω	higher-order PRED

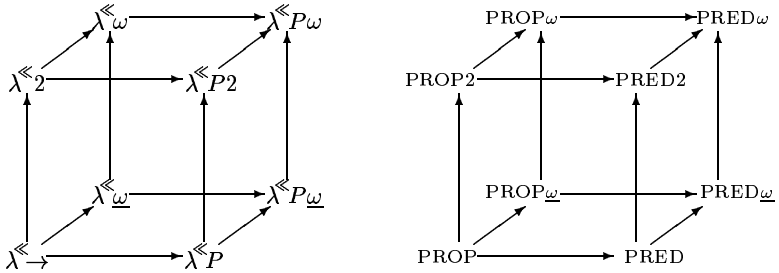


Figure 7: The Axiom Table for the $\lambda^{\ll}$-cube, their Logic Systems, and their cubes

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma,\Delta\vdash(\lambda 3{:}\emptyset.3):\Pi 3{:}\emptyset.i}\ \boxed{6}\ \boxed{7}\ \vdash 3\ll_{\Gamma,\Delta} X:ok \qquad \dfrac{\boxed{4}\quad(*,*,*)\in\mathcal{R}}{\Gamma,\Delta\vdash[Z\ll_\Delta X].i:*}}{\Gamma,\Delta\vdash(\lambda 3{:}\emptyset.3)\,X:[3\ll_{\Gamma,\Delta} X].i}\quad \dfrac{}{\Gamma\vdash\Pi f(X){:}\Delta.[3\ll_{\Gamma,\Delta} X].i:*}}{\Gamma\vdash\lambda f(X){:}\Delta.(\lambda 3{:}\emptyset.3)X:\Pi f(X){:}\Delta.[3\ll_{\Gamma,\Delta} X].i}\ \boxed{3}\ \boxed{4}\ \boxed{5}}{\Gamma\vdash(\lambda f(X){:}\Delta.(\lambda 3{:}\emptyset.3)X)\,f(3):[f(X)\ll_{\Gamma,\Delta} f(3)].[3\ll_{\Gamma,\Delta} X].i}\ \boxed{1}\ \boxed{2}}{\Gamma\vdash(\lambda f(X){:}\Delta.(\lambda 3{:}\emptyset.3)X)\,f(3):i}$$

Let $\Gamma\triangleq f{:}\Pi Z{:}i.i, 3{:}i, 4{:}i,\ \Delta\triangleq X{:}i$, $\boxed{1}\triangleq[f(X)\ll_{\Gamma,\Delta} f(3)].[3\ll_{\Gamma,\Delta} X].i=_{\rho\delta} i$, $\boxed{2}\triangleq\Gamma\vdash i:*$, $\boxed{3}\triangleq\Gamma\vdash f(3):[Z\ll_\Delta 3].i$, $\boxed{4}\triangleq\Gamma,\Delta\vdash f(X):[Z\ll_\Delta X].i$, $\boxed{5}\triangleq\vdash f(X)\ll_{\Gamma,\Delta} f(3):ok$, $\boxed{6}\triangleq\Gamma,\Delta\vdash X:i$, $\boxed{7}\triangleq\Gamma,\Delta\vdash 3:i$.

Figure 8: A Typing Derivation

Proof: *Define the mapping $\diamond$ by induction on terms (and pointwise extends to contexts) as follows:*

$$\begin{array}{rcl}
\alpha^\diamond & \triangleq & \alpha\\
(\lambda A{:}\Gamma.B)^\diamond & \triangleq & \lambda A^\diamond{:}\Gamma^\diamond.B^\diamond\\
(\Pi A{:}\Gamma.B)^\diamond & \triangleq & \Pi A^\diamond{:}\Gamma^\diamond.B^\diamond\\
(A,B)^\diamond & \triangleq & A^\diamond,B^\diamond\\
(AB)^\diamond & \triangleq & A^\diamond B^\diamond \text{ if } AB \text{ is not a } \Rightarrow_{\rho\delta}\text{-redex}\\
((\lambda A{:}\Gamma.B)C)^\diamond & \triangleq & B^\diamond[C^\diamond/A^\diamond]\\
((A,B)C)^\diamond & \triangleq & (AC)^\diamond,(BC)^\diamond\\
([A\ll_\Gamma B].C)^\diamond & \triangleq & \begin{cases}\sigma_{(A^\diamond\ll_\Gamma B^\diamond)}(C^\diamond) & \text{if defined}\\ [A^\diamond\ll_{\Gamma^\diamond}B^\diamond].C^\diamond & \text{otherwise}\end{cases}
\end{array}$$

Then the proof proceeds by induction on A. $\Box$

Theorem 6.3 ($\mathbf{P^2TS}$ are Confluent) *The relation $\mapsto_{\rho\delta}$ is confluent.*

6.2 $\mathbf{P^2TS}$: Subject Reduction

We denote $\vdash$ as the symbol of derivability in one of the type systems $\mathcal{S}$ of the $\mathbf{P^2TS}$. The following lemmas are the classical routine lemmas used to prove standard metatheory on **PTS**.

Definition 6.3 (Legal Context) *1. $\emptyset$ is legal;*

2. If Γ is legal and $\alpha\notin Dom(\Gamma)$ and $\Gamma\vdash A:s$, then $\Gamma,\alpha{:}A$ is legal.

Lemma 6.4 (Context) *If $\Gamma\vdash A:B$ then Γ is legal and $\Gamma\vdash B:s$.*

Lemma 6.5 (Free Variables) *If $\Gamma\vdash A:B$, then $FV(A)\cup FV(B)\subseteq Dom(\Gamma)$.*

Lemma 6.6 (Thinning) *If* $\Gamma \vdash A : B$ *and* $\Gamma \subseteq \Delta$ *and* Δ *legal, then* $\Delta \vdash A : B$.

Lemma 6.7 (Replacement) *If* $\Gamma, X{:}C, \Delta \vdash A : B$ *and* $Y \notin BV(A) \cup BV(B)$, *then* $\Gamma, Y{:}C, \Delta\{Y/X\} \vdash A\{Y/X\} : B\{Y/X\}$

Lemma 6.8 (Substitution) *If* $\Gamma, \Delta \vdash A : B$ *and* $\Delta \equiv \vec{X_i}{:}\vec{D_i}$ *and* $\sigma \equiv \{\vec{C_i}/\vec{X_i}\}$ *and for all* $i = 1 \ldots n$ *we have* $\Gamma \vdash C_i : D_i$, *then* $\Gamma, \sigma(\Sigma) \vdash \sigma(A) : \sigma(B)$.

Lemma 6.9 (Generation)

1. *If* $\Gamma \vdash s_1 : A$, *then* $A =_{\rho\delta} s_2$ *and* $s_1 : s_2 \in \mathcal{A}$;
2. *If* $\Gamma \vdash \alpha : A$, *then* $\Gamma \equiv \Gamma, \alpha{:}B, \Delta$, *and* $A =_{\rho\delta} B$;
3. *If* $\Gamma \vdash \lambda A{:}\Delta.B : D$, *then* $\Gamma, \Delta \vdash B : C$, *and* $\Gamma \vdash \Pi A{:}\Delta.C : s$, *and* $D =_{\rho\delta} \Pi A{:}\Delta.C$;
4. *If* $\Gamma \vdash \Pi A{:}\Delta.B : D$, *then* $\Gamma, \Delta \vdash A : C$, *and* $\Gamma, \Delta \vdash C : s_1$, *and* $\Gamma, \Delta \vdash B : s_2$, *and* $(s_1, s_2, s_3) \in \mathcal{R}$, *and* $D =_{\rho\delta} s_3$;
5. *If* $\Gamma \vdash AB : F$, *then* $\Gamma \vdash A : \Pi C{:}\Delta.D$, *and* $\Gamma, \Delta \vdash C : E$, *and* $\Gamma \vdash B : E$ *and* $\vdash C \ll_{\Gamma,\Delta} B : ok$, *and* $F =_{\rho\delta} D[B/C]$;
6. *If* $\Gamma \vdash A, B : C$, *then* $\Gamma \vdash A : D$ *and* $\Gamma \vdash B : D$ *and* $C =_{\rho\delta} D$;
7. *If* $\Gamma \vdash [C \ll_\Delta B].A : F$, *then* $\Gamma \vdash B : E$ *and* $\Gamma, \Delta \vdash C : E$ *and* $\Gamma \vdash A : E$ *and* $\vdash C \ll_{\Gamma,\Delta} B : ok$ *and* $F =_{\rho\delta} [C \ll_\Delta B].D$.

Proof: *Each item can be easily proved by induction on the structure of derivations. Intuitively, if* $\Gamma \vdash A : B$, *then we apply a number of times some non-syntax directed rules, namely* $(Weak)$ *and* $(Conv)$, *before applying a structural rule, say* (R), *where the term* A *is formed and derived in a context* $\Gamma' \subseteq \Gamma$. *Therefore,* $(\mathsf{R}) \in \{(Axioms), (Struct), (Start), (Abs), (Prod), (Appl), (Subst)\}$. *At that point, we distinguish between the above rules and we conclude with an application of the Thinning Lemma 6.6. Graphically:*

$$\dfrac{\dfrac{\cdots}{\Gamma' \vdash A : B'}\ (\mathsf{R})}{\begin{array}{c}\vdots \\ (Weak)+(Conv) \\ \vdots \\ \Gamma \vdash A : B\end{array}}$$

□

Lemma 6.10 (Correctness of Types) *If* $\Gamma \vdash A : B$ *then* $B \equiv \Box$, *or* $\Gamma \vdash B : C$.

Lemma 6.11 (Strengthening) *For functional* $\mathbf{P^2\,T\,S}$ *[4], if* $\Gamma, \Delta, \Sigma \vdash A : B$ *and* $FV(A) \cup FV(B) \cup FV(\Sigma) \notin Dom(\Delta)$, *then* $\Gamma, \Sigma \vdash A : B$.

Theorem 6.4 (Subject Reduction) *If* $\Gamma \vdash A : B$, *and* $A \twoheadrightarrow_{\rho\delta} C$, *then* $\Gamma \vdash C : B$.

Proof: *By induction on the number of reduction steps and, then by induction on the structure of* A *and finally by cases on the top-level reduction, the latter by induction on the structure of the typed derivation.* □

Lemma 6.12 (Unicity of Typing) *For functional* $\mathbf{P^2\,T\,S}$, *if* $\Gamma \vdash A : B$ *and* $\Gamma \vdash A : C$, *then* $B =_{\rho\delta} C$.

Conjecture 6.1 (Consistency in $\mathbf{P^2\,T\,S}$ [4]) $\mathbf{P^2\,T\,S}$ *extending* $\lambda^{\ll}2$ *are logically consistent, i.e. given a closed term* A, *we have* $\not\vdash A : \bot$ *where* $\bot \triangleq \Pi X{:}*.X$.

6.3 $\mathbf{P^2\,T\,S}$: Normalization.

We conjecture also the strong normalization of $\mathbf{P^2\,T\,S}$. We do not have yet a formal proof: intuitively we would like to reduce the normalization of $\mathbf{P^2\,T\,S}$ to the normalization of plain **PTS** [4]. To do this, we need, as usual, an erasing function E_Γ (context depending) which maps legal terms into the $\mathbf{P^2\,T\,S}$ to corresponding legal terms into **PTS**. The main purpose of the function E_Γ is to eliminate structures and patterns and intuitively, it works as follows:

1. Structures (*i.e.* pairs) are compiled as usually in plain λ-calculus;
2. Constant terms are compiled to fresh variables;
3. Application compilation distinguishes between function-structures (hence modeling (δ)-reductions), and function-abstractions (hence modeling $(\rho\sigma)$-reductions).

Lemma 6.13 (Erasing) *Let* $\mapsto_\beta^+$ *be the transitive closure of* $\mapsto_\beta$ *in* **PTS**.

1. $\mathsf{E}_\Gamma(A\{B/X\}) \equiv \mathsf{E}_\Delta(A)\{\mathsf{E}_\Gamma(B)/X\}$ *with* $\Delta \equiv \Gamma, X{:}C$;
2. *If* $A \mapsto_{\rho\delta} B$, *then* $\mathsf{E}_\Gamma(A) \mapsto_\beta^+ \mathsf{E}_\Gamma(B)$;
3. *If* $\Gamma \vdash A : B$, *then* $\mathsf{E}_\emptyset(\Gamma) \vdash_{PTS} \mathsf{E}_\Gamma(A) : \mathsf{E}_\Gamma(B)$.

Conjecture 6.2 (Strong Normalization of $\mathbf{P^2\,T\,S}$) *Given* $\Gamma \vdash A : B$, *then*

1. A *is strongly normalizing;*
2. B *is strongly normalizing;*
3. *All predicates in* Γ *are strongly normalizing.*

7 Conclusion

We have shown that Pure Pattern Type Systems provide a well-behaved extension of Pure Type Systems. Our results open the road for the study of new powerful proof engines and (meta)languages based on sophisticated type theories.

We are also considering conditions less restrictive than RPC that allow larger class of patterns in abstractions.

One future direction is the introduction of the Null failure as a first-class citizen together with the corresponding evaluation and typing rules. This would allow one to explicitly describe an exception catching mechanism.

Finally we plan to study an algorithmic presentation of the type system, and explore a limited form of decidable higher-order unification, in the style of λ-Prolog [20, 21]

Acknowledgments. Luigi would like to thanks Simonetta Ronchi della Rocca and Furio Honsell, for the time spent to teach to him the fundamentals and the insight of the "cubes-stuff", Vincent van Oostrom for its seminal paper *"Lambda Calculus with Patterns"*, Philippe de Groote, for its fruitful discussions on higher-order matching and unification, and Jöelle Despeyroux for her constructive suggestions on the syntax and the early version of type system.

A Metatheory

This appendix collects all the meta-theory of $\mathbf{P}^2\,\mathbf{T}\,\mathbf{S}$: more precisely, we study confluence, typability, and normalization properties. The full version of the paper contains all the proofs and extensive examples and typing derivations[1].

A.1 General Notations

Free and Bound Variables. In the following, we revisit the notions of free and bound variables.

Definition A.1 (FV, BV) *The set of free and bound variables are inductively defined on terms and contexts as follows:*

$$
\begin{array}{lcl}
FV(s) & \triangleq & \emptyset \\
FV(a) & \triangleq & \emptyset \\
FV(X) & \triangleq & \{X\} \\
FV(\emptyset) & \triangleq & \emptyset \\
FV(\lambda A{:}\Gamma.B) & \triangleq & FV(\Gamma) \cup (FV(B) \setminus FV(A)) \\
FV(\Pi A{:}\Gamma.B) & \triangleq & FV(\Gamma) \cup (FV(B) \setminus FV(A)) \\
FV([A \ll_\Gamma C].B) & \triangleq & FV(\Gamma) \cup (FV(B) \setminus FV(A)) \cup FV(C) \\
FV(AB) & \triangleq & FV(A) \cup FV(B) \\
FV(A, B) & \triangleq & FV(A) \cup FV(B) \\
FV(\Gamma, \alpha{:}A) & \triangleq & FV(\Gamma) \cup (FV(A) \setminus Dom(\Gamma))
\end{array}
$$

$$
\begin{array}{lcl}
BV(s) & \triangleq & \emptyset \\
BV(a) & \triangleq & \emptyset \\
BV(X) & \triangleq & \emptyset \\
BV(\emptyset) & \triangleq & \emptyset \\
BV(\lambda A{:}\Gamma.B) & \triangleq & BV(\Gamma) \cup BV(A) \cup BV(B) \cup FV(A) \\
BV(\Pi A{:}\Gamma.B) & \triangleq & BV(\Gamma) \cup BV(A) \cup BV(B) \cup FV(A) \\
BV([A \ll_\Gamma C].B) & \triangleq & BV(\Gamma) \cup BV(A) \cup BV(B) \cup FV(A) \\
 & & \cup BV(C) \\
BV(AB) & \triangleq & BV(A) \cup BV(B) \\
BV(A, B) & \triangleq & BV(A) \cup BV(B) \\
BV(\Gamma, \alpha{:}A) & \triangleq & BV(\Gamma) \cup BV(A) \cup Dom(\Gamma, \alpha{:}A)
\end{array}
$$

As in ordinary systems dealing with dependent types, we suppose that in $\Gamma, \alpha{:}A$, the α does not appear free in Γ, and A.

Proposition A.1 (Free & Bound) *For a term A we have:*

$$FV(A) \cup BV(A) = Var(A), \text{ and } FV(A) \cap BV(A) = \emptyset$$

[1] This version is permanently updated as the conjectures stated in the paper are proved.

A.2 Confluence

The first property we are interested in is the confluence and it is to see that this does not hold when the general syntax of terms is considered. When the RPC introduced in Section 4 is used for restricting the form of patterns in abstraction, the confluence is recovered. For proving this we use a parallel reduction in the style of *Tait & Martin-Löf* and a technique inspired from [25].

Let us define a "comfortable" notion of parallel reduction for our $\mathbf{P}^2\,\mathbf{T}\,\mathbf{S}$.

Definition A.2 (Parallel Reduction) *The parallel reduction, denoted by $\Rightarrow_{\rho\sigma\delta}$, is inductively defined as follows:*

$$
\begin{array}{ll}
(Par_1) & \alpha \Rightarrow_{\rho\sigma\delta} \alpha \\
(Par_2) & \lambda A_1{:}\Gamma.B_1 \Rightarrow_{\rho\sigma\delta} \lambda A_2{:}\Delta.B_2 \\
 & \quad \textit{if } A_1 \Rightarrow_{\rho\sigma\delta} A_2, B_1 \Rightarrow_{\rho\sigma\delta} B_2, \Gamma \Rightarrow_{\rho\sigma\delta} \Delta \\
(Par_3) & \Pi A_1{:}\Gamma.B_1 \Rightarrow_{\rho\sigma\delta} \Pi A_2{:}\Delta.B_2 \\
 & \quad \textit{if } A_1 \Rightarrow_{\rho\sigma\delta} A_2, B_1 \Rightarrow_{\rho\sigma\delta} B_2, \Gamma \Rightarrow_{\rho\sigma\delta} \Delta \\
(Par_4) & [A_1 \ll_\Gamma B_1].C_1 \Rightarrow_{\rho\sigma\delta} [A_2 \ll_\Delta B_2].C_2 \\
 & \quad \textit{if } A_1 \Rightarrow_{\rho\sigma\delta} A_2, B_1 \Rightarrow_{\rho\sigma\delta} B_2, C_1 \Rightarrow_{\rho\sigma\delta} C_2, \Gamma \Rightarrow_{\rho\sigma\delta} \Delta \\
(Par_5) & A_1 B_1 \Rightarrow_{\rho\sigma\delta} A_2 B_2 \\
 & \quad \textit{if } A_1 \Rightarrow_{\rho\sigma\delta} A_2, B_1 \Rightarrow_{\rho\sigma\delta} B_2 \\
(Par_6) & A_1, B_1 \Rightarrow_{\rho\sigma\delta} A_2, B_2 \\
 & \quad \textit{if } A_1 \Rightarrow_{\rho\sigma\delta} A_2, B_1 \Rightarrow_{\rho\sigma\delta} B_2 \\
(Par_7) & (\lambda A_1{:}\Gamma.B_1)C_1 \Rightarrow_{\rho\sigma\delta} [A_2 \ll_\Delta C_2].B_2 \\
 & \quad \textit{if } A_1 \Rightarrow_{\rho\sigma\delta} A_2, B_1 \Rightarrow_{\rho\sigma\delta} B_2, C_1 \Rightarrow_{\rho\sigma\delta} C_2, \Gamma \Rightarrow_{\rho\sigma\delta} \Delta \\
(Par_8) & [A_1 \ll_\Gamma B_1].C_1 \Rightarrow_{\rho\sigma\delta} \sigma_{(A_2 \ll_\Delta B_2)}(C_2) \\
 & \quad \textit{if } A_1 \Rightarrow_{\rho\sigma\delta} A_2, B_1 \Rightarrow_{\rho\sigma\delta} B_2, C_1 \Rightarrow_{\rho\sigma\delta} C_2, \Gamma \Rightarrow_{\rho\sigma\delta} \Delta \\
(Par_9) & (A_1, B_1)C_1 \Rightarrow_{\rho\sigma\delta} A_2 C_2, B_2 C_2 \\
 & \quad \textit{if } A_1 \Rightarrow_{\rho\sigma\delta} A_2, B_1 \Rightarrow_{\rho\sigma\delta} B_2, C_1 \Rightarrow_{\rho\sigma\delta} C_2
\end{array}
$$

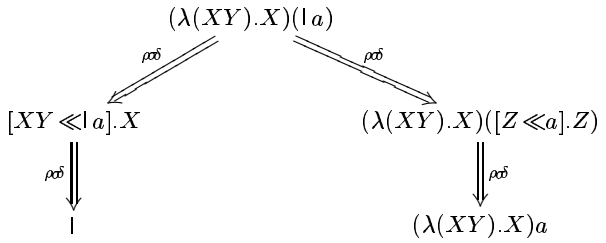
The rules $(Par_1), \ldots, (Par_6)$ indicate that the relation $\Rightarrow_{\rho\sigma\delta}$ includes the identity on $\lambda^{\ll}$-terms, *i.e.* $A \Rightarrow_{\rho\sigma\delta} A$ holds for all $A \in \mathcal{T}$. In rule (Par_1) α can be $\emptyset$ (*i.e.* the empty context) and the rule (Par_6) is overloaded and describes the reduction for structures as well as for contexts. Rules (Par_7), (Par_8) and (Par_9) correspond to the rules (ρ), (σ) and (δ) respectively. Intuitively, $A \Rightarrow_{\rho\sigma\delta} B$ means that B is obtained from A, by simultaneous contractions of some $(\rho\sigma\delta)$-redexes possibly overlapping one each other.

One can notice that the definition of the parallel reduction can be simplified (as in Section 6.1) if only patterns satisfying the RPC are considered (and thus reducing only to themselves) and if the contexts are considered to be in normal form.

We want to prove that this parallel reduction satisfies the diamond property and that is the transitive closure of the relation $\mapsto_{\rho\sigma\delta}$. We obtain as an immediate consequence the confluence of this latter relation. Nevertheless, the $\Rightarrow_{\rho\sigma\delta}$ does not satisfy the diamond property when arbitrary patterns are used in abstractions and we give in what follows a few examples showing the main problems.

Non-confluent reductions As well explained in [26], adding pattern matching facilities can cause the failure of the Church-Rosser property. The extra complications we have with $\mathbf{P}^2\,\mathbf{T}\,\mathbf{S}$ are mainly due to patterns that contain "active" variables.

Example A.1 (Confluence Failure in Plain $\mathbf{P^2TS}$) *We give an example for plain $\mathbf{P^2TS}$ inspired from [26] where, for the sake of readability, all type decorations are omitted. We consider $\mathsf{I} \triangleq \lambda Z.Z$ and an abstraction containing an application pattern with an "active" variable and we obtain the following non-confluent reduction:*



A sufficient condition to recover the Church-Rosser property for the top-level rules is essentially based on the well-known notion of *Rigid Pattern Condition* of Vincent van Oostrom [26].

The Rigid Pattern Condition. The main objective of this condition is to ensure that $\mathbf{P^2TS}$ are well-behaved extension of plain **PTS**. Rigid pattern condition is sufficient for obtaining the "diamond property" of the (parallel) reduction and hence the confluence of the reduction in $\mathbf{P^2TS}$. As well explained in [26], the rigid pattern condition does not characterize the shape of patterns. We will show one subset of $\mathbf{P^2TS}$ that can be used as patterns in reductions.

As showed in Example A.1, it turns out that the shape of patterns in lambda and product abstractions is essential in order to avoid bizarre non-confluent reductions. Therefore, we need to restrict to $\mathbf{P^2TS}$ whose patterns in abstractions satisfy some properties. This was firstly formalized in [26]. Our terms have now the following (restricted) new shape with $\mathcal{P} \subset \mathcal{T}$:

$$\mathcal{T} ::= \lambda \mathcal{P}{:}\Gamma.\mathcal{T} \mid \Pi \mathcal{P}{:}\Gamma.\mathcal{T} \mid [\mathcal{P} \ll_{\mathcal{C}} \mathcal{T}].\mathcal{T} \mid \ldots \text{ as before } \ldots$$

The following condition on the set of patterns $\mathcal{P}$ was presented by Vincent van Oostrom and it turns out to be sufficient to prove that the parallel reduction $\Rightarrow_{\rho\delta}$ satisfies the diamond property.

Definition A.3 (Rigid Pattern Condition (RPC) [26]) *Let $\mathcal{P} \subset \mathcal{T}$.*
$\mathcal{P}$ is RPC *if for all $A \in \mathcal{P}$ and for all $B, \vec{C} \in \mathcal{T}$, there exists $\vec{D} \in \mathcal{T}$ such that*

$$A\{\vec{C}/\vec{X}\} \Rightarrow_{\rho\delta} B \text{ implies } \vec{C} \Rightarrow_{\rho\delta} \vec{D} \text{ and } B \equiv A\{\vec{D}/\vec{X}\}.$$

Remark A.1 *1. $\mathcal{V}$ obviously satisfy* RPC*, but $\mathcal{T}$ do not.*

2. *The operational semantics and the type systems remain unchanged taking into account the (now restricted) shape of patterns.*

3. *From now on we deal only with terms which satisfy* RPC*.*

4. *The* RPC *will be only used to prove the diamond property of the parallel reduction.*

We first need some "routine" lemmas.

Lemma A.1 (Permutation) *1. If $X \notin FV(C)$, then $A\{B/X\}\{C/Y\} \equiv A\{C/Y\}\{B\{C/Y\}/X\}$;*

2. If $A =_{\rho\delta} B$, then $A\{C/X\} =_{\rho\delta} B\{C/X\}$.

Proof: *1. By induction on the definition of substitution.*
2. By induction on the definition of $=_{\rho\delta}$.
□

Lemma A.2 (Relations) $\mapsto_{\rho\delta} \subseteq \Rightarrow_{\rho\delta} \subseteq \twoheadrightarrow_{\rho\delta}$.

For the parallel reduction, the following lemma holds.

Lemma A.3 (Parallel) *If $A \Rightarrow_{\rho\delta} B$ and $\vec{C} \Rightarrow_{\rho\delta} \vec{D}$, then $A\{\vec{C}/\vec{X}\} \Rightarrow_{\rho\delta} B\{\vec{D}/\vec{X}\}$.*

Proof: *By induction on the definition of $\Rightarrow_{\rho\delta}$.*

(Par_1)**.** *Trivial with a little care in case $A \equiv Y$ and $Y \notin \vec{X}$;*

(Par_2) **and** (Par_3) **and** (Par_4)**.** *We consider the first case, the others being similar. Then $(\lambda A_1{:}\Gamma.B_1) \Rightarrow_{\rho\delta} (\lambda A_2{:}\Delta.B_2)$ if $A_1 \Rightarrow_{\rho\delta} A_2$ and $B_1 \Rightarrow_{\rho\delta} B_2$ and $\Gamma \Rightarrow_{\rho\delta} \Delta$ and we have:*

$$\begin{array}{rcl}(\lambda A_1{:}\Gamma.B_1)\{\vec{C}/\vec{X}\} & \triangleq & (\lambda A_1{:}\Gamma\{\vec{C}/\vec{X}\}.B_1\{\vec{C}/\vec{X}\}) \\ & \overset{ih}{\Rightarrow}_{\rho\delta} & (\lambda A_2{:}\Delta\{\vec{D}/\vec{X}\}.B_2\{\vec{D}/\vec{X}\}) \\ & \triangleq & (\lambda A_2{:}\Delta.B_2)\{\vec{D}/\vec{X}\}\end{array}$$

using α-conversion (if necessary) and the corresponding (Par) rule.

(Par_5) **and** (Par_6)**.** *We consider the first case, the second one being similar. Then $A_1B_1 \Rightarrow_{\rho\delta} A_2B_2$ if $A_1 \Rightarrow_{\rho\delta} A_2$ and $B_1 \Rightarrow_{\rho\delta} B_2$ and we have:*

$$\begin{array}{rcl}(A_1B_1)\{\vec{C}/\vec{X}\} & \triangleq & (A_1\{\vec{C}/\vec{X}\}B_1\{\vec{C}/\vec{X}\}) \\ & \overset{ih}{\Rightarrow}_{\rho\delta} & (A_2\{\vec{D}/\vec{X}\}B_2\{\vec{D}/\vec{X}\}) \\ & \triangleq & (A_2B_2)\{\vec{D}/\vec{X}\})\end{array}$$

(Par_7)**.** *Then $A \equiv (\lambda A_1{:}\Gamma.B_1)C_1 \Rightarrow_{\rho\delta} [A_2 \ll_{\Delta} C_2].B_2$ if $A_1 \Rightarrow_{\rho\delta} A_2$ and $B_1 \Rightarrow_{\rho\delta} B_2$ and $C_1 \Rightarrow_{\rho\delta} C_2$ and $\Gamma \Rightarrow_{\rho\delta} \Delta$ and we have:*

$$\begin{array}{rcl}A\{\vec{C}/\vec{X}\} & \triangleq & (\lambda A_1{:}\Gamma\{\vec{C}/\vec{X}\}.B_1\{\vec{C}/\vec{X}\})C_1\{\vec{C}/\vec{X}\} \\ & \overset{ih}{\Rightarrow}_{\rho\delta} & [A_2 \ll_{\Delta\{\vec{D}/\vec{X}\}} C_2\{\vec{D}/\vec{X}\}].B_2\{\vec{D}/\vec{X}\} \\ & \equiv & ([A_2 \ll_{\Delta} C_2].B_2)\{\vec{D}/\vec{X}\}\end{array}$$

using α-conversion (if necessary).

(Par_8)**.** *Then $A \equiv [A_1 \ll_{\Gamma} B_1].C_1 \Rightarrow_{\rho\delta} \sigma_{(A_2 \ll_{\Delta} B_2)}(C_2)$ if $A_1 \Rightarrow_{\rho\delta} A_2$ and $B_1 \Rightarrow_{\rho\delta} B_2$ and $C_1 \Rightarrow_{\rho\delta} C_2$ and $\Gamma \Rightarrow_{\rho\delta} \Delta$ and we have:*

$$\begin{array}{rcl}A\{\vec{C}/\vec{X}\} & \triangleq & ([A_1 \ll_{\Gamma\{\vec{C}/\vec{X}\}} B_1\{\vec{C}/\vec{X}\}].C_1\{\vec{C}/\vec{X}\} \\ & \overset{ih}{\Rightarrow}_{\rho\delta} & \sigma_{(A_2 \ll_{\Delta\{\vec{D}/\vec{X}\}} B_2\{\vec{D}/\vec{X}\})}(C_2\{\vec{D}/\vec{X}\}) \\ & \equiv & (\sigma_{(A_2 \ll_{\Delta} B_2)}(C_2))\{\vec{D}/\vec{X}\}\end{array}$$

using α-conversion (if necessary) and Lemma A.1(1).

(Par_9). *Then* $A \equiv (A_1B_1)C_1 \Rightarrow_{\rho\sigma\delta} (A_2C_2),(B_2C_2)$ *if* $A_1 \Rightarrow_{\rho\sigma\delta} A_2$ *and* $B_1 \Rightarrow_{\rho\sigma\delta} B_2$ *and* $C_1 \Rightarrow_{\rho\sigma\delta} C_2$ *and we have:*

$$\begin{array}{rcl} A\{\vec{C}/\vec{X}\} & \triangleq & (A_1\{\vec{C}/\vec{X}\}B_1\{\vec{C}/\vec{X}\})C_1\{\vec{C}/\vec{X}\} \\ & \stackrel{ih}{\Rightarrow}_{\rho\sigma\delta} & (A_2\{\vec{D}/\vec{X}\}B_2\{\vec{D}/\vec{X}\})C_2\{\vec{D}/\vec{X}\} \\ & \triangleq & ((A_2B_2)C_2)\{\vec{D}/\vec{X}\}) \end{array}$$

□

Now, to prove the Church-Rosser Theorem for $\mapsto\!\!\!\!\rightarrow_{\rho\sigma\delta}$, it is sufficient to show the *Diamond Property* $\diamond$ for $\Rightarrow_{\rho\sigma\delta}$, *i.e.*:

Theorem A.1 (Diamond Property for $\Rightarrow_{\rho\sigma\delta}$) $\Rightarrow_{\rho\sigma\delta} \Vdash \diamond$, *i.e.*
if $A \Rightarrow_{\rho\sigma\delta} B$ *and* $A \Rightarrow_{\rho\sigma\delta} C$, *then there exists* D, *such that* $B \Rightarrow_{\rho\sigma\delta} D$ *and* $C \Rightarrow_{\rho\sigma\delta} D$.

Indeed, we can adapt the stronger statement from [25] to our $\mathbf{P^2TS}$, as follows:

Theorem A.2 (Strong Church-Rosser for $\Rightarrow_{\rho\sigma\delta}$) *If* $A \Rightarrow_{\rho\sigma\delta} B$, *then there exists a term* $A^\diamond$, *depending only on* A, *such that* $B \Rightarrow_{\rho\sigma\delta} A^\diamond$. *(Intuitively, the term* $A^\diamond$ *is obtained by contracting all the redexes existing in* A *simultaneously.)*

Proof: *Define the mapping* $\diamond$ *by induction on terms (and pointwise extends to contexts) as follows:*

$$\begin{array}{lcl} \alpha^\diamond & \triangleq & \alpha \\ (\lambda A{:}\Gamma.B)^\diamond & \triangleq & \lambda A^\diamond{:}\Gamma^\diamond.B^\diamond \\ (\Pi A{:}\Gamma.B)^\diamond & \triangleq & \Pi A^\diamond{:}\Gamma^\diamond.B^\diamond \\ ([A \ll_\Gamma B].C)^\diamond & \triangleq & [A^\diamond \ll_{\Gamma^\diamond} B^\diamond].C^\diamond \\ & & \textit{if } [A \ll_\Gamma B].C \textit{ is not a } \Rightarrow_{\rho\sigma\delta}\textit{-redex} \\ (A,B)^\diamond & \triangleq & A^\diamond, B^\diamond \\ (AB)^\diamond & \triangleq & A^\diamond B^\diamond \\ & & \textit{if } AB \textit{ is not a } \Rightarrow_{\rho\sigma\delta}\textit{-redex} \\ ((\lambda A{:}\Gamma.B)C)^\diamond & \triangleq & [A^\diamond \ll_{\Gamma^\diamond} C^\diamond].B^\diamond \\ ([A \ll_\Gamma B].C)^\diamond & \triangleq & \sigma_{(A^\diamond \ll_{\Gamma^\diamond} B^\diamond)}C^\diamond \\ ((A,B)C)^\diamond & \triangleq & (AC)^\diamond, (BC)^\diamond \end{array}$$

The term $[A \ll_\Gamma B].C$ *is not a redex if* $\sigma_{(A^\diamond \ll B^\diamond)}$ *is not defined while the term* AB *is not a redex if it does not have one of the forms* $(\lambda A{:}\Gamma.B)C$ *or* $(A,B)C$.

Now the proof proceeds by induction on A:

($A \equiv \alpha$) *If* $\alpha \Rightarrow_{\rho\sigma\delta} B$, *then* $B \equiv \alpha \Rightarrow_{\rho\sigma\delta} \alpha \equiv \alpha^\diamond$.

($A \equiv \lambda C_1{:}\Gamma_1.D_1$) *Then,* $\lambda C_1{:}\Gamma_1.D_1 \Rightarrow_{\rho\sigma\delta} \lambda C_2{:}\Gamma_2.D_2$, *with* $C_1 \Rightarrow_{\rho\sigma\delta} C_2$ *and* $\Gamma_1 \Rightarrow_{\rho\sigma\delta} \Gamma_2$ *and* $D_1 \Rightarrow_{\rho\sigma\delta} D_2$. *By induction hypothesis, we get* $C_2 \Rightarrow_{\rho\sigma\delta} C_1^\diamond$ *and* $\Gamma_2 \Rightarrow_{\rho\sigma\delta} \Gamma_1^\diamond$ *and* $D_2 \Rightarrow_{\rho\sigma\delta} D_1^\diamond$. *Hence* $\lambda C_2{:}\Gamma_2.D_2 \Rightarrow_{\rho\sigma\delta} \lambda C_1^\diamond{:}\Gamma_1^\diamond.D_1^\diamond \equiv (\lambda C_1{:}\Gamma_1.D_1)^\diamond$.

($A \equiv \Pi C_1{:}\Gamma_1.D_1$) *As in the previous case.*

($A \equiv C_1D_1$ **and** A **is not a** $\Rightarrow_{\rho\sigma\delta}$**-redex.**) *Then* $C_1D_1 \Rightarrow_{\rho\sigma\delta} C_2D_2$, *with* $C_1 \Rightarrow_{\rho\sigma\delta} C_2$ *and* $D_1 \Rightarrow_{\rho\sigma\delta} D_2$. *By induction hypothesis, we get* $C_2 \Rightarrow_{\rho\sigma\delta} C_1^\diamond$ *and* $D_2 \Rightarrow_{\rho\sigma\delta} D_1^\diamond$. *Hence,* $C_2D_2 \Rightarrow_{\rho\sigma\delta} C_1^\diamond D_1^\diamond \equiv (C_1D_1)^\diamond$.

($A \equiv C_1, D_1$) *As in the previous case.*

($A \equiv [C_1 \ll_{\Gamma_1} D_1].E_1$ **is not a** $\Rightarrow_{\rho\sigma\delta}$**-redex.**) *As in the previous case.*

($A \equiv (\lambda C_1{:}\Gamma_1.D_1)E_1$ **is a redex.**) *Then* $((\lambda C_1{:}\Gamma_1.D_1)E_1) \Rightarrow_{\rho\sigma\delta} B$, *and either* $B \equiv (\lambda C_2{:}\Gamma_2.D_2)E_2$ *or* $B \equiv [C_2 \ll_{\Gamma_2} E_2].D_2$, *with* $C_1 \Rightarrow_{\rho\sigma\delta} C_2$ *and* $\Gamma_1 \Rightarrow_{\rho\sigma\delta} \Gamma_2$ *and* $D_1 \Rightarrow_{\rho\sigma\delta} D_2$ *and* $E_1 \Rightarrow_{\rho\sigma\delta} E_2$. *By induction hypothesis, we get* $C_2 \Rightarrow_{\rho\sigma\delta} C_1^\diamond$ *and* $\Gamma_2 \Rightarrow_{\rho\sigma\delta} \Gamma_1^\diamond$ *and* $D_2 \Rightarrow_{\rho\sigma\delta} D_1^\diamond$ *and* $E_2 \Rightarrow_{\rho\sigma\delta} E_1^\diamond$. *Since* $A^\diamond \equiv [C_1^\diamond \ll_{\Gamma_1^\diamond} E_1^\diamond].D_1^\diamond$ *the property holds for the two previously mentioned cases since:*

- ($B \equiv (\lambda C_2{:}\Gamma_2.D_2)E_2) \Rightarrow_{\rho\sigma\delta} A^\diamond$ *by applying* (Par_2),
- ($B \equiv [C_2 \ll_{\Gamma_2} E_2].D_2) \Rightarrow_{\rho\sigma\delta} A^\diamond$ *by applying* (Par_7);

($A \equiv (C_1, D_1)E_1$ **is a redex.**) *As in the previous case.*

($A \equiv [C_1 \ll_{\Gamma_1} D_1].E_1$ **is a redex.**) *Then* $[C_1 \ll_{\Gamma_1} D_1].E_1 \Rightarrow_{\rho\sigma\delta} B$, *and either* $B \equiv [C_2 \ll_{\Gamma_2} D_2].E_2$ *or* $B \equiv \sigma_{(C_2 \ll_{\Gamma_2} D_2)}(E_2)$, *with* $C_1 \Rightarrow_{\rho\sigma\delta} C_2$ *and* $\Gamma_1 \Rightarrow_{\rho\sigma\delta} \Gamma_2$ *and* $D_1 \Rightarrow_{\rho\sigma\delta} D_2$ *and* $E_1 \Rightarrow_{\rho\sigma\delta} E_2$. *By induction hypothesis, we get* $C_2 \Rightarrow_{\rho\sigma\delta} C_1^\diamond$ *and* $\Gamma_2 \Rightarrow_{\rho\sigma\delta} \Gamma_1^\diamond$ *and* $D_2 \Rightarrow_{\rho\sigma\delta} D_1^\diamond$ *and* $E_2 \Rightarrow_{\rho\sigma\delta} E_1^\diamond$. *Since* $A^\diamond \equiv \sigma_{(C_1^\diamond \ll_{\Gamma_1^\diamond} D_1^\diamond)}E_1^\diamond$ *the property holds for the two previously mentioned cases since:*

- ($B \equiv [C_2 \ll_{\Gamma_2} D_2].E_2) \Rightarrow_{\rho\sigma\delta} A^\diamond$ *by applying* (Par_8),
- ($B \equiv \sigma_{(C_2 \ll_{\Gamma_2} D_2)}(E_2)) \Rightarrow_{\rho\sigma\delta} A^\diamond$ *by using Lemma A.3 and the* RPC *condition, and applying* (Par_4).

□

Theorem A.3 ($\mathbf{P^2TS}$ which satisfy RPC are Confluent) *The relation* $\mapsto_{\rho\sigma\delta}$ *is confluent.*

Proof: *By Lemmas A.2 (or A.1) and A.2* □

A Characterization of RPC. We have extended **PTS** with patterns, and we proved that $\mathbf{P^2TS}$ which satisfy RPC are confluent. However, we have not characterized yet an "honest" subset $\mathcal{P}$ of $\mathcal{T}$ which properly contains $\mathcal{V}$. This can be easily done by considering the set of terms defined below.

Definition A.4 (A Simple Characterization of $\mathcal{P} \subset \mathcal{T}$) *Define*

$$\mathcal{P} \triangleq \{A \in NF(\rho\sigma\delta) \mid A \textit{ is "linear" with no "active" variables}\}$$

One can easily verify that the above characterization of $\mathcal{P} \subset \mathcal{T}$ *is* RPC *and properly contains* $\mathcal{V}$. *As pointed out in [26] the set* $\mathcal{P}$ *is not the maximal set for which our* $\mapsto_{\rho\sigma\delta}$ *is confluent, e.g. the set*

$\mathcal{P}_\Omega \triangleq \mathcal{P} \cup \{\Omega\}$ *where* $\Omega \triangleq (\lambda X.XX)(\lambda X.XX)$ *also satisfy* RPC.

Remark A.2 ([26]) *"...* RPC *does not prevent reduction from taking place inside patterns. However, one easily verifies that if* $A \in \mathcal{P}$ *and* $A \mapsto_{\rho\sigma\delta} B$, *then* $A \equiv B$ *...". As usual, the type system is delegated to "block" (i.e. make not typable) those terms thanks to the strong normalization theorem.*

A.3 Subject Reduction

We denote $\vdash$ as the symbol of derivability in one of the type systems $\mathcal{S}$ of the $\mathbf{P^2TS}$. The following lemmas are the classical routine lemmas used to prove standard metatheory on **PTS**.

Definition A.5 (Legal Context)

1. *$\emptyset$ is legal;*
2. *If Γ is legal and $\alpha \notin Dom(\Gamma)$ and $\Gamma \vdash A : s$, then $\Gamma, \alpha{:}A$ is legal.*

Lemma A.4 (Context) *If $\Gamma \vdash A : B$ then Γ is legal and $\Gamma \vdash B : s$.*

Proof: *By induction on the structure of the derivation.*

(*Axioms*). *By Definition A.5;*

(*Struct*). *By I.H.;*

(*Start*). *By I.H.;*

(*Weak*). *By Definition A.5 and I.H.;*

(*Conv*). *By I.H.;*

(*Abs*). *By I.H.;*

(*Prod*). *Idem (Weak);*

(*Subst*). *Idem (Weak);*

(*Appl*). *Idem (Weak);*

□

Lemma A.5 (Free Variables)
If $\Gamma \vdash A : B$, then $FV(A) \cup FV(B) \subseteq Dom(\Gamma)$.

Proof: *By induction on the structure of the derivation.*

(*Axioms*). *Trivial;*

(*Struct*). *By I.H.*

(*Start*). *By I.H.;*

(*Weak*). *By I.H.;*

(*Conv*). *By I.H.;*

(*Abs*). *Then $\Gamma_1 \vdash \lambda A{:}\Gamma_2.B : \Pi A{:}\Gamma_2.B$. By I.H. Observe that $FV(B) \setminus FV(A) \subseteq Dom(\Gamma_1)$;*

(*Prod*). *Idem (Abs);*

(*Subst*). *Idem (Abs);*

(*Appl*). *Then $\Gamma_1 \vdash AB : [C \ll_{\Gamma_2} B].D$. By I.H. Observe that $FV([C \ll_{\Gamma_2} B].D) = (FV(D) \setminus FV(C)) \cup FV(B) \cup FV(\Gamma_2)$.*

□

Lemma A.6 (Thinning) *If $\Gamma_1 \vdash A : B$ and $\Gamma_1 \subseteq \Gamma_2$ and Γ_2 legal, then $\Gamma_2 \vdash A : B$.*

Proof: *By induction on the structure of the derivation.*

(*Axioms*). *Trivial*

(*Struct*). *By I.H.;*

(*Start*). *Apply (Weak) using legality of Γ_2;*

(*Weak*). *Idem (Start);*

(*Conv*). *By I.H.;*

(*Abs*). *Idem (Start);*

(*Prod*). *Idem (Start);*

(*Subst*). *Idem (Start);*

(*Appl*). *By I.H.;*

□

Lemma A.7 (Replacement) *If $\Gamma_1, X{:}C, \Gamma_2 \vdash A : B$ and $Y \notin BV(A) \cup BV(B)$, then $\Gamma_1, Y{:}C, \Gamma_2\{Y/X\} \vdash A\{Y/X\} : B\{Y/X\}$*

Proof: *Routine.* □

Lemma A.8 (Substitution) *If $\Gamma_1, \Gamma_2, \Gamma_3 \vdash A : B$ and $\Gamma_2 \equiv X_1{:}D_1, \ldots X_n{:}D_n$ and $\sigma \equiv \{C_1/X_1, \ldots, C_n/X_n\} \equiv \{\vec{C}/\vec{X}\}$ and for all $i = 1 \ldots n$ we have $\Gamma_1 \vdash C_i : D_i$, then $\Gamma_1, \sigma(\Gamma_3) \vdash \sigma(A) : \sigma(B)$.*

Proof: *By induction on the structure of the derivation.*

(*Axioms*). *Trivial;*

(*Struct*). *By I.H.;*

(*Start*). *If $\alpha \in \vec{X}$, then by I.H., otherwise direct using Lemma A.5;*

(*Weak*). *Idem (Start);*

(*Conv*). *By I.H. and Lemma A.1(2);*

(*Abs*). *By I.H.;*

(*Prod*). *By I.H.;*

(*Subst*). *By I.H.;*

(*Appl*). *By I.H.;*

□

Lemma A.9 (Generation)

1. *If $\Gamma \vdash s_1 : A$, then $A =_{\rho\delta} s_2$ and $s_1 : s_2 \in \mathcal{A}$;*
2. *If $\Gamma \vdash \alpha : A$, then $\Gamma \equiv \Gamma, \alpha{:}B, \Delta$, and $A =_{\rho\delta} B$;*
3. *If $\Gamma \vdash \lambda A{:}\Delta.B : D$, then $\Gamma, \Delta \vdash B : C$, and $\Gamma \vdash \Pi A{:}\Delta.C : s$, and $D =_{\rho\delta} \Pi A{:}\Delta.C$;*
4. *If $\Gamma \vdash \Pi A{:}\Delta.B : D$, then $\Gamma, \Delta \vdash A : C$, and $\Gamma, \Delta \vdash C : s_1$, and $\Gamma, \Delta \vdash B : s_2$, and $(s_1, s_2, s_3) \in \mathcal{R}$, and $D =_{\rho\delta} s_3$;*
5. *If $\Gamma \vdash AB : F$, then $\Gamma \vdash A : \Pi C{:}\Delta.D$, and $\Gamma, \Delta \vdash C : E$, and $\Gamma \vdash B : E$ and $\vdash C \ll_{\Gamma,\Delta} B : ok$, and $F =_{\rho\delta} [C \ll_{\Delta} B].D$;*
6. *If $\Gamma \vdash A, B : C$, then $\Gamma \vdash A : D$ and $\Gamma \vdash B : D$ and $C =_{\rho\delta} D$;*
7. *If $\Gamma \vdash [C \ll_{\Delta} B].A : F$, then $\Gamma \vdash B : E$ and $\Gamma, \Delta \vdash C : E$ and $\Gamma \vdash A : D$ and $\vdash C \ll_{\Gamma,\Delta} B : ok$ and $F =_{\rho\delta} [C \ll_{\Delta} B].D$.*

Proof: *Each item can be easily proved by induction on the structure of derivations. Intuitively, if $\Gamma \vdash A : B$, then we apply a number of times some non-syntax directed rules, namely (Weak) and (Conv), before applying a structural rule, say (R), where the term A is formed and derived in a context $\Gamma' \subseteq \Gamma$. Therefore, (R) $\in$ {(Axioms), (Struct), (Start), (Abs), (Prod), (Appl), (Subst)}. At that point, we distinguish between the*

above rules and we conclude with an application of the Thinning Lemma A.6. Graphically:

$$\dfrac{\dfrac{\cdots}{\Gamma' \vdash A : B'}\ (\mathsf{R})}{\begin{array}{c}\vdots\\ (Weak)+(Conv)\\ \vdots\end{array}}$$
$$\Gamma \vdash A : B$$

□

Lemma A.10 (Correctness of Types) *If $\Gamma \vdash A : B$ then $B \equiv \Box$, or $\Gamma \vdash B : C$.*

Proof: *By induction on the structure of the derivation.*

(*Axioms*). *Trivial;*

(*Struct*). *By I.H.;*

(*Start*). *Trivial;*

(*Weak*). *By I.H.;*

(*Conv*). *By I.H.;*

(*Abs*). *Trivial;*

(*Prod*). *Trivial;*

(*Subs*). *Trivial;*

(*Appl*). *Then $\Gamma_1 \vdash AB : [C \ll_{\Gamma_2} B].D$. By I.H. $\Gamma_1 \vdash \Pi C{:}\Gamma_2.D : C_1$ and by Generation Lemma A.9 $\Gamma_1, \Gamma_2 \vdash D : s$.*
Since $FV([C \ll_{\Gamma_2} B].D) = (FV(D) \setminus FV(C)) \cup FV(B) \cup FV(\Gamma_2)$, apply Substitution Lemma A.8 to get $\Gamma_1 \vdash [C \ll_{\Gamma_2} B].D : C_1$ using α-conversion (if necessary).

□

Lemma A.11 (Strengthening) *If $\Gamma_1, \Gamma_2, \Gamma_3 \vdash A : B$ and $FV(A) \cup FV(B) \cup FV(\Gamma_3) \notin Dom(\Gamma_2)$, then $\Gamma_1, \Gamma_3 \vdash A : B$.*

Proof: *By induction on the structure of the derivation.*

(*Axioms*). *Trivial;*

(*Struct*). *By I.H.;*

(*Start*). *By I.H*

(*Weak*). *Trivial;*

(*Conv*). *By I.H.;*

(*Abs*). *Then $\Gamma_1, \Gamma_2, \Gamma_3 \vdash \lambda A{:}\Gamma_4.B : \Pi A{:}\Gamma_4.C$. By I.H. using legality of $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$;*

(*Prod*). *Idem (Abs);*

(*Subs*). *Idem (Abs);*

(*Appl*). *By I.H. using legality of contexts.*

□

Theorem A.4 (Subject Reduction) *If $\Gamma_1 \vdash A : B$, and $A \twoheadrightarrow_{\rho\sigma\delta} C$, then $\Gamma_1 \vdash C : B$.*

Proof: *By induction on the number of reduction steps and, then by induction on the structure of A' and finally by cases on the top-level reduction, the latter by induction on the structure of the typed derivation.*

(Top-level)

(ρ) *Let $(\lambda A_1{:}\Gamma_2.B_1)C_1 \to [A_1 \ll_{\Gamma_2} C_1].B_1$. Then*

$$\Gamma_1 \vdash (\lambda A_1{:}\Gamma_2.B_1)C_1 : D_1$$

By Generation Lemma A.9,

$$\begin{array}{l}\Gamma_1 \vdash \lambda A_1{:}\Gamma_2.B_1 : \Pi A_1{:}\Gamma_2.E_1\\ \Gamma_1, \Gamma_2 \vdash A_1 : F_1\\ \Gamma_1 \vdash C_1 : F_1\\ \Gamma_1, \Gamma_2 \vdash A_1 \ll_{\Gamma_1,\Gamma_2} C_1 : ok\\ D_1 =_{\rho\sigma\delta} [A_1 \ll_{\Gamma_2} C_1].E_1\end{array}$$

By Generation Lemma A.9,

$$\begin{array}{l}\Gamma_1, \Gamma_2 \vdash B_1 : E_2\\ \Gamma_1 \vdash \Pi A_1{:}\Gamma_2.E_2 : s\\ \Pi A{:}\Gamma_2.E_1 =_{\rho\sigma\delta} \Pi A{:}\Gamma_2.E_2\end{array}$$

The latter equation implies

$$E_1 =_{\rho\sigma\delta} E_2$$

By (Subst) we have

$$\Gamma_1 \vdash [A_1 \ll_{\Gamma_2} C_1].B_1 : D_1$$

(σ) *Let $[A_1 \ll_{\Gamma_2} C_1].B_1 \to \sigma_{(A_1 \ll_{\Gamma_2} C_1)}(B_1)$. Then*

$$\Gamma_1 \vdash [A_1 \ll_{\Gamma_2} C_1].B_1 : D_1$$

By Generation Lemma A.9,

$$\begin{array}{l}\Gamma_1 \vdash B_1 : E_1\\ \Gamma_1, \Gamma_2 \vdash A_1 : F_1\\ \Gamma_1 \vdash C_1 : F_1\\ \Gamma_1, \Gamma_2 \vdash A_1 \ll_{\Gamma_1,\Gamma_2} C_1 : ok\\ D_1 =_{\rho\sigma\delta} [A_1 \ll_{\Gamma_2} C_1].E_1\end{array}$$

By Substitution Lemma A.1 we have

$$\Gamma_1 \vdash \sigma_{(A_1 \ll_{\Gamma_2} C_1)}(B_1) : \sigma_{(A_1 \ll_{\Gamma_2} C_1)}(E_1)$$

and finally by (Conv) we have

$$\Gamma_1 \vdash B_1[C_1/A_1] : D_1$$

(δ) *Let $(A_1, B_1)C_1 \to (A_1 C_1), (B_1 C_1)$. Then*

$$\Gamma_1 \vdash (A_1, B_1)C_1 : D_1$$

By Generation Lemma A.9

$$\begin{array}{l}\Gamma_1 \vdash (A_1, B_1) : \Pi A_2{:}\Gamma_2.E_2\\ \Gamma_1, \Gamma_2 \vdash A_2 : F_1\\ \Gamma_1 \vdash C_1 : F_1\\ \Gamma_1, \Gamma_2 \vdash A_2 \ll C_1 : ok\\ D_1 =_{\rho\sigma\delta} [A_2 \ll_{\Gamma_2} C_1].E_2\end{array}$$

By Generation Lemma A.9

$$\begin{array}{l}\Gamma_1 \vdash A_1 : \Pi A_3{:}\Gamma_3.E_3\\ \Gamma_1 \vdash B_1 : \Pi A_3{:}\Gamma_3.E_3\\ \Pi A_2{:}\Gamma_2.E_2 =_{\rho\sigma\delta} \Pi A_3{:}\Gamma_3.E_3\end{array}$$

By Correctness of Types Lemma A.10 and (Conv)

we have

$$\Gamma_1 \vdash A_1 : \Pi A_2{:}\Gamma_2.E_2$$

and by (Appl) we have

$$\Gamma_1 \vdash A_1C_1 : [A_2 \ll_{\Gamma_2} C_1].E_2$$

The same schema be applied to obtain

$$\Gamma_1 \vdash B_1C_1 : E[A_2 \ll_{\Gamma_2} C_1].E_2$$

By Correctness of Types Lemma A.10 and (Conv) we have

$$\Gamma_1 \vdash D_1 : s$$

Apply (Struct) and (Conv) to obtain

$$\Gamma_1 \vdash (A_1C_1), (B_1C_1) : D_1$$

(One-step)
Let $\mathsf{Ctx}[-]$ *be any context in* $\mathcal{T}$ *with a single hole, recall the one-step rule:*

$$\frac{A \to B}{\mathsf{Ctx}[A] \mapsto_{\rho\delta} \mathsf{Ctx}[B]} \quad (\mathsf{Ctx}[-])$$

We disregard arbitrary applications of (Weak) and (Conv) which can be treated in the standard way.

($\mathsf{Ctx}[-] \equiv [\,]$) *Direct;*

($\mathsf{Ctx}[-] \equiv A[\,]$) *Let* $\Gamma_1 \vdash AB_1 : [E \ll_{\Gamma_2} B_1].D$, *and* $B_1 \to B_2$. *Then by (Appl)*

$$\Gamma_1 \vdash A : \Pi C{:}\Gamma_2.D \qquad \Gamma_1 \vdash B_1 : E$$
$$\Gamma_1, \Gamma_2 \vdash C : E \qquad \vdash C \ll_{\Gamma_1,\Gamma_2} B_1 : ok$$

By Subject reduction on the top-level rules

$$\Gamma_1 \vdash B_2 : E$$

By (RPC)

$$\vdash A \ll_{\Gamma_1,\Gamma_2} B_2 : ok$$

By Lemma A.1(2)

$$[E \ll_{\Gamma_2} B_1].D =_{\rho\delta} [E \ll_{\Gamma_2} B_2].D$$

By (Appl)

$$\Gamma_1 \vdash AB_2 : [E \ll_{\Gamma_2} B_2].D$$

($\mathsf{Ctx}[-] \equiv [\,]B$) *The only interesting case is when*

$$\Gamma_1 \vdash (\lambda A_1{:}\Gamma_2.B_1)C_1 : [A_1 \ll_{\Gamma_2} C_1].D$$

and $A_1 \to A_2$. *Then by (Appl)*

$$\Gamma_1 \vdash \lambda A_1{:}\Gamma_2.B_1 : \Pi A_1{:}\Gamma_2.D \qquad \Gamma_1, \Gamma_2 \vdash A_1 : E$$
$$\vdash A_1 \ll_{\Gamma_1,\Gamma_2} C_1 : ok \qquad \Gamma_1 \vdash C_1 : E$$

By Generation Lemma A.9

$$\Gamma_1, \Gamma_2 \vdash B_1 : D$$

By Subject reduction on the top-level rules

$$\Gamma_1, \Gamma_2 \vdash A_2 : E$$

and by Correctness of Types Lemma A.10

$$\Gamma_1, \Gamma_2 \vdash E : s_1 \qquad \Gamma_1, \Gamma_2 \vdash D : s_2$$

By (Prod) (using $(s_1, s_2, s_3) \in \mathcal{R}$, *obtained by Generation Lemma A.9)*

$$\Gamma_1 \vdash \Pi A_2{:}\Gamma_2.D : s_3$$

By (Abs)

$$\Gamma_1 \vdash \lambda A_2{:}\Gamma_2.B_1 : \Pi A_2{:}\Gamma_2.D$$

By (RPC)

$$\vdash A_2 \ll_{\Gamma_1,\Gamma_2} C_1 : ok$$

By (Appl)

$$\Gamma_1 \vdash (\lambda A_1{:}\Gamma_2.B_1)C_1 : [A_2 \ll_{\Gamma_2} C_1].D$$

By Correctness of Types Lemma A.10 and Lemma A.1(2) (here $FV(A_2) \subseteq FV(A_1)$*)*

$$\Gamma_1 \vdash [A_1 \ll_{\Gamma_2} C_1].D : s \quad [A_1 \ll_{\Gamma_2} C_1].D =_{\rho\delta} [A_2 \ll_{\Gamma_2} C_1].D$$

By (Conv)

$$\Gamma_1 \vdash (\lambda A_1{:}\Gamma_2.B_1)C_1 : [A_1 \ll_{\Gamma_2} C_1].D$$

($\mathsf{Ctx}[-] \equiv \lambda[\,]{:}\Gamma.B$) *Direct using* (RPC);

($\mathsf{Ctx}[-] \equiv \lambda A{:}\Gamma_1, \alpha{:}[\,], \Gamma_2.B$) *Direct;*

($\mathsf{Ctx}[-] \equiv \lambda A{:}\Gamma.[\,]$) *Direct;*

($\mathsf{Ctx}[-] \equiv A, [\,]$) *Direct;*

($\mathsf{Ctx}[-] \equiv [\,], B$) *Direct;*

(Multi-step)
Trivial by induction on the number of one-step reductions. □

Lemma A.12 (Unicity of Typing) *If* $\Gamma \vdash A' : B'$ *and* $\Gamma \vdash A' : C'$, *then* $B' =_{\rho\delta} C'$.

Proof: *By induction on the structure of* A.

($A' \equiv s_1$) *Trivial using functionality;*

($A' \equiv \alpha$) *By Generation Lemma A.9;*

($A' \equiv \lambda A{:}\Gamma_2.B$) *By Generation Lemma A.9;*

($A' \equiv [A \ll_{\Gamma_2} C].B$) *By Generation Lemma A.9;*

($A' \equiv \Pi A{:}\Gamma_2.B$) *Trivial using functionality;*

($A' \equiv AB$) *Suppose* $\Gamma_1 \vdash AB : F$ *and* $\Gamma \vdash AB : G$. *Then by Generation Lemma A.9*

$$\Gamma_1 \vdash A : \Pi C{:}\Gamma_2.D \qquad \Gamma_1 \vdash A : \Pi C'{:}\Gamma_2'.D'$$
$$\Gamma_1 \vdash AB : [C \ll_{\Gamma_2} B].D \qquad \Gamma_1 \vdash AB : [C' \ll_{\Gamma_2} B].D'$$

By I.H.

$$\Pi C{:}\Gamma_2.D =_{\rho\delta} \Pi C'{:}\Gamma_2'.D'$$

Hence

$$C =_{\rho\delta} C \quad D =_{\rho\delta} D \quad [C \ll_{\Gamma_2} B].D =_{\rho\delta} [C' \ll_{\Gamma_2} B].D'$$

($A' \equiv A, B$) *By Generation Lemma A.9;*

□

Conjecture A.1 (Consistency in $\mathbf{P^2TS}$ [4]) $\mathbf{P^2TS}$ *extending* $\lambda^{\ll}2$ *are logically consistent, i.e. given a closed term* A, *we have* $\not\vdash A : \bot$ *where* $\bot \triangleq \Pi X{:}*.X$.

References

[1] F. Barbanera, M. Fernández, and H. Geuvers. Modularity of strong normalisation and confluence in the algebraic λ-cube. In *Proc. of LICS*, pages 406–415, 1994.

[2] H. Barendregt. *Lambda Calculus: its Syntax and Semantics*. North Holland, 1984.

[3] H. Barendregt. Introduction to Generalised Type Systems. *Journal of Functional Programming*, 1(2):125–154, 1991.

[4] H. Barendregt. Lambda Calculi with Types. In *Handbook of Logic in Computer Science*, volume II, pages 118–310. Oxford University Press, 1992.

[5] G. Barthe and H. Geuvers. Modular properties of algebraic type systems. In *Proc. of HOA*, pages 37–56, 1995.

[6] F. Blanqui. *Type Theory and Rewriting*. PhD thesis, University Paris-Sud, 2001.

[7] V. Breazu-Tannen. Combining algebra and higher-order types. In *Proc. of LICS*, pages 82–90, 1988.

[8] H.-J. Bürckert. Matching–A Special Case of Unification? *Journal of Symbolic Computation*, 8(5):523–536, 1989.

[9] A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1941.

[10] H. Cirstea, C. Kirchner, and L. Liquori. Matching Power. In *Proc. of RTA*, volume 2051 of *LNCS*. Springer-Verlag, 2001.

[11] H. Cirstea, C. Kirchner, and L. Liquori. The Rho Cube. In *Proc. of FOSSACS*, volume 2030 of *LNCS*, pages 166–180, 2001.

[12] T. Coquand, R. Pollack, and M. Takeyama. Modules as dependently typed records. Manuscript, 2002.

[13] G. Faure and C. Kirchner. Exceptions in the rewriting calculus. In *Proc. of RTA*, volume 2378 of *LNCS*, pages 66–82. Springer-Verlag, 2002.

[14] H. Geuvers. *Logics and type systems*. PhD thesis, University of Nijmegen, 1993.

[15] H. Geuvers and M.J. Nederhof. A modular proof of strong normalisation for the Calculus of Constructions. *Journal of Functional Programming*, 1(2):155–189, 1991.

[16] J.-M. Hullot. Associative-Commutative Pattern Matching. In *Proc. of IJCAI*, 1979.

[17] J.P. Jouannaud and M. Okada. Executable higher-order algebraic specification languages. In *Proc. of LICS*, pages 350–361, 1991.

[18] D. Kesner, L. Puel, and V. Tannen. A typed pattern calculus. *Information and Computation*, 124(1):32–61, 10 1996.

[19] X. Leroy. A modular module system. *Journal of Functional Programming*, 10(3):269–303, May 2000.

[20] D. Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. In *Proc. of ELP*, volume 475 of *LNCS*, pages 253–281. Springer-Verlag, 1991.

[21] D. Miller, G. Nadathur, F. Pfenning, and A. Shedrov. Uniform Proofs as a Foundation for Logic Programming. *Annals of Pure and Applied Logics*, 51:125–157, 1991.

[22] S. Peyton Jones. *The Implementation of Functional Programming Languages*. Prentice Hall, 1987.

[23] S.L. Peyton Jones and E. Meijer. Henk: a Typed Intermediate Language. In *Types in Compilation Workshop*, 1997.

[24] R. Pollack. *The Theory of LEGO: A Proof Checker for the Extended Calculus of Constructions*. PhD thesis, University of Edinburgh, 1994.

[25] M. Takahashi. Parallel Reductions in λ-calculus. *Journal of Symbolic Computation*, 7:113–123, 1989.

[26] V. van Oostrom. Lambda Calculus with Patterns. Technical Report IR-228, Faculteit der Wiskunde en Informatica, Vrije Universiteit Amsterdam, 1990.

[27] J. Zwanenburg. Pure type systems with subtyping. In *Proc. of TLCA*, volume 1581 of *LNCS*. Springer-Verlag, 1999.