A Quantitative Understanding of Pattern Matching

Sandra Alves
DCC-FCUP & CRACS, University of Porto, Portugal

Delia Kesner
Université de Paris, CNRS, IRIF, France
Institut Universitaire de France, France

Daniel Ventura
INF, Universidade Federal de Goiás, Goiânia, Brazil

Abstract

This paper shows that the recent approach to quantitative typing systems for programming languages can be extended to pattern matching features. Indeed, we define two resource-aware type systems, named $\mathscr{U}$ and $\mathscr{E}$, for a λ-calculus equipped with pairs for both patterns and terms. Our typing systems borrow some basic ideas from [19], which characterises (head) normalisation in a *qualitative* way, in the sense that typability and normalisation coincide. But, in contrast to [19], our systems also provide *quantitative* information about the *dynamics* of the calculus. Indeed, system $\mathscr{U}$ provides *upper bounds* for the *length* of (head) normalisation sequences *plus* the *size* of their corresponding normal forms, while system $\mathscr{E}$, which can be seen as a refinement of system $\mathscr{U}$, produces *exact bounds* for *each of* them. This is achieved by means of a non-idempotent intersection type system equipped with different technical tools. First of all, we use product types to type pairs instead of the disjoint unions in [19], which turn out to be an essential quantitative tool because they remove the confusion between "being a pair" and "being duplicable". Secondly, typing sequents in system $\mathscr{E}$ are decorated with tuples of integers, which provide quantitative information about normalisation sequences, notably *time* (cf. length) and *space* (cf. size). Moreover, the time resource information is remarkably refined, because it discriminates between different kinds of reduction steps performed during evaluation, so that beta, substitution and matching steps are counted separately. Another key tool of system $\mathscr{E}$ is that the type system distinguishes between *consuming* (contributing to time) and *persistent* (contributing to space) constructors.

2012 ACM Subject Classification Theory of computation → Lambda calculus; Theory of computation → Operational semantics

Keywords and phrases Intersection Types, Pattern Matching, Exact Bounds

Digital Object Identifier 10.4230/LIPIcs.TYPES.2019.3

Funding This work was partially done within the framework of ANR COCA HOLA (16-CE40-004-01). It was partially funded by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020 and by the Brazilian Research Council (CNPq) grant Universal 430667/2016-7.

Acknowledgements We are grateful to Antonio Bucciarelli and Simona Ronchi Della Rocca for fruitful discussions.

1 Introduction

Pattern matching mechanisms are used in several modern programming languages and proof assistants as they provide an efficient way to process and decompose data. However, the semantics of programming languages usually focus on λ-calculi –a much more basic formalism– thus causing a conceptual gap between theory and practice, simply because some properties of the λ-calculus do not translate directly to languages with matching primitives. Several

examples of this mismatch can be cited, e.g. solvability [19], standardisation for pattern calculi [37], and neededness [15]. It is then crucial to study the semantics of programming languages with pattern matching features by means of formal calculi equipped with built-in patterns, referred to as *pattern calculi* (e.g. [21, 35, 32, 37, 6]).

The notion of λ-abstraction in pattern-calculi is generalised to functions of the form $\lambda p.t$, where p is a *pattern* specifying the expected structure of their arguments. For instance, in calculi equipped with *pair constructors* for both patterns and terms, the term $\lambda\langle x, y\rangle.x$ becomes a valid abstraction, to be only successfully evaluated against pairs, i.e. arguments of the form $\langle u, v\rangle$, and yielding the first projection of this pair, i.e. the first component u of the pair. In this work we focus on such a pattern calculus. This can be seen as a simplified form of *algebraic* pattern matching, but still powerful enough to reason about the most interesting features of existing syntactical matching mechanisms.

Type information, and in particular the size of arbitrary type derivations in some special type disciplines, has been used as a powerful quantitative tool to reason about *time* (*length* of evaluation sequences) and *space* (*size* of normal forms). More precisely, when t evaluates to t', then the size of the type derivation of t' is smaller than that of t, thus the size of type derivations provides an *upper bound* for the length of normalisation sequences as well as for the size of their corresponding normal forms. This was first done for the (call-by-name) notions of head and leftmost evaluation implemented by two variants of the Krivine's abstract machine (KAM) [22, 23, 44], and it was later appropriately extended to other formalisms, e.g. [8, 20, 25, 38].

Now we discuss some interesting features of the underlying type system that we use in this paper. While (idempotent) *intersection types* [11] allow terms to be typed with distinct types by means of an intersection operator $\cap$, which verifies not only associativity and commutativity but also idempotency given by $\sigma \cap \sigma = \sigma$, *non-idempotent* intersection types distinguishes between $\sigma \cap \sigma$ and σ, thus also discriminating *quantitative* information in type derivations. For this reason, idempotent (resp. non-idempotent) types are often represented by sets (resp. *multisets*). For example, the term $\lambda x.\lambda y.xyy$ can be typed with $\{\{\sigma\} \to \{\sigma\} \to \tau\} \to \{\sigma\} \to \tau$ in the first model, while the non-idempotent version becomes $[[\sigma] \to [\sigma] \to \tau] \to [\sigma, \sigma] \to \tau$. As a consequence, a type derivation for $(\lambda x.\lambda y.xyy)uv : \tau$ in the idempotent system only depends on one type derivation for $u : \{\sigma\} \to \{\sigma\} \to \tau$ and another one for $v : \sigma$, while for its reduct $uvv : \tau$, two derivations for $v : \sigma$ are needed. In contrast, a type derivation in the non-idempotent system already requires two derivations for $v : \sigma$ to correctly infer $(\lambda x.\lambda y.xyy)uv : \tau$. Therefore, while type derivations may increase after reductions in the former, they decrease in the latter.

Non-idempotent intersection (also called nowadays *quantitative*) type systems, have been independently introduced in the framework of the λ-calculus by Gardner [27] and Kfoury [43]. Although widely unnoticed, the *quantitative* power of such systems turned out to be crucial in several resource aware consumption investigations. It was only after [16] that this quantitative feature was highlighted, and since De Carvalho's thesis in 2007 (see also [22]) its relation with linear logic [28] and quantitative relational models has been deeply explored. As its idempotent counterpart, non-idempotent intersection type systems may characterise different notions of normalisation (such as head, weak and strong) [23, 13, 20] but, instead of using some semantical argument (e.g. reducibility) to prove such a characterisation, simple combinatorial arguments are enough to guarantee normalisation of typable terms.

If instead of upper bounds one wants to obtain *exact bounds*, then the crucial point is to measure only *minimal* typing derivations, which give the notion of *all and only information* for typings (cf. [56] for an abstract definition). Syntactic notions of minimal typings were

supplied for the *head evaluation* strategy implemented by the KAM [22], then for the *maximal evaluation* strategy [13] for the λ-calculus. The technique was further developed in [2] with the introduction of an appropriate notion of *tightness* capturing minimal typings, thus systematically broadening the definition of exact bounds for different evaluation strategy of the λ-calculus. In all these works, it is possible to extract from a (minimal) type derivation, both the length of the reduction sequence to normal form as well as the size of this normal form. The tightness technique was also applied for call-by-value [3], call-by-need [4], linear head evaluation [3], as well as for several evaluation strategies in classical logic [42]. Our paper extends these results to a λ-calculus with pair pattern matching by providing two sound and complete typing systems, named $\mathscr{U}$ and $\mathscr{E}$, that respectively provide upper bounds an exact measures for the length of (head) normalisation sequences, as well as for the size of the corresponding reached normal forms.

Contributions

The first contribution of this paper is to go beyond the *qualitative* characterisation of head normalisation for pair pattern calculi [19] by providing a typing system $\mathscr{U}$ being able to compute *upper bounds* for head evaluation. To achieve this, we have introduced different key tools on the untyped side –the reduction calculus– as well as on the typed side –the type system itself.

On the *untyped* side, one of the main reasons why the type system in [19] fails to provide upper bounds or exact measures for head normalisation is because *commuting conversions* are considered as *independent* rules of the reduction relation associated to the underlying pattern calculus. A typical example is the commuting rule $t[p\backslash v]u \mapsto_\sigma (tu)[p\backslash v]$ pushing out an explicit matching from an application when there is no capture of free variables. Indeed, the size of type derivations is not strictly decreasing w.r.t. commuting conversions. We solve this problem by integrating these (structural) commuting conversions into the non-structural operational reduction rules, so that the resulting system, based on *explicit matchings*, implements reduction *at a distance* [5]. Thus for example, the operational Beta-rule $(\lambda p.t)u \mapsto t[p\backslash u]$ in [19] becomes here $\mathtt{L}[\![\lambda p.t]\!]u \mapsto \mathtt{L}[\![t[p\backslash u]]\!]$, which combines the commuting conversion $\mapsto_\sigma$ with the (non-structural) Beta-reduction rule into a single rule. Even more interesting cases are presented in Sec. 2. Moreover, our presentation provides a suitable *deterministic* head evaluation strategy which is complete w.r.t. the (non-deterministic) notion of head-normalisation defined in [19], in the sense that both notions turn out to be equivalent, thus answering one of the open questions in [19].

On the *typed* side, we adopt standard product types specified by means of a *pair type*. This stands in contrast to the disjoint unions used in [19], which have an important undesirable consequence, because multisets of types in this model carry two completely different meanings: being a pair (but not necessarily a duplicable pair), or being a duplicable term (but not necessarily a pair). Our product types restore a crucial idea in non-idempotent type theory: multisets of types are only assigned to terms that are going to be duplicated during evaluation.

The new specification of the deterministic reduction system at a distance is now well-behaved w.r.t. our first type system $\mathscr{U}$: if t is well typed in $\mathscr{U}$, then the size of its type derivation gives an upper bound to the (deterministic) head-reduction sequence from t to its (head) normal form. Our system $\mathscr{U}$ can then be seen as a form of quantitative (relational) *model* for the pair pattern calculus (Sec. 4), following the lines of [17, 18, 48].

The second contribution of this paper is to go beyond upper bounds by providing a typing system $\mathscr{E}$ being able to provide *exact bounds* for head evaluation. This is done by using several key tools.

An important notion used in system $\mathscr{E}$ is the clear distinction between *consuming* and *persistent* constructors. This has some intuition coming from the theory of residuals [10], where any symbol occurring in a normal form can be traced back to the original term. A constructor is consuming (resp. persistent) if it is consumed (resp. not consumed) during head-reduction. For instance, in $(\lambda z.z)(\lambda z.z)$ the first abstraction is consuming while the second one is persistent. This dichotomy between consuming/persistent constructors has already been highlighted in [41, 42] for the $\lambda\mu$-calculus, and it is adapted here for the pattern calculus. Indeed, patterns and terms are consumed when the pair constructor is destroyed during the execution of the pattern matching rule. Otherwise, patterns and pairs are persistent, and they do appear in the normal form of the original term. For example, in the term $(\lambda z.(\lambda\langle x, y\rangle.\mathtt{I})zz)\langle u, v\rangle$, the pair $\langle u, v\rangle$ is going to be duplicated, but only one of its copies is going to be consumed by the matching operations. The other copy will be persistent and contribute to the normal form of the term.

Another major ingredient of our approach is the use of *tight* types, inspired by [2], and the corresponding notion of tight (cf. minimal) derivations. This is combined with the introduction of *counters* in the typing judgements, which are now of the form $\Gamma \vdash^{(b,e,m,f)} t : \sigma$. These counters are used to discriminate between the different sorts of reduction steps performed during evaluation, so that firing beta (b), computing substitution (e) or matching (m) steps are exactly and independently counted for each tight type derivation. More precisely, *soundness* of our system $\mathscr{E}$guarantees that if a judgement $\Gamma \vdash^{(b,e,m,f)} t : \sigma$ is tightly derivable, then b (resp e and m) corresponds to the number of beta firing (resp. substitution and matching) rules used to head evaluate the term t, while f is exactly the size of the corresponding normal form. Moreover, *completeness*, given by the reverse implication of the previous statement, also holds.

The following list summarises our contributions:

- A deterministic head-strategy for the pattern calculus which is complete w.r.t. the notion of head-normalisation.
- A sound and complete type system $\mathscr{U}$, which provides upper bounds for the length of head-normalisation sequences plus the size of its corresponding normal forms.
- Refinement of system $\mathscr{U}$ to a sound and complete system $\mathscr{E}$, being able to provide independent exact bounds for both the length of head-normalisation sequences and the size of its corresponding normal forms.

Other Related Works

Non-idempotent intersection types have been applied to the λ-calculus for the characterisation of termination with respect to a variety of evaluation strategies, such as call-by-value [25, 3], call-by-need [36, 8, 4] and (linear) head reduction [27, 2]. They have been well-adapted also to some explicit resource calculi [13, 38, 39], as well as to pattern calculi [12, 19, 9], proof-nets [24], classical logic [40, 42] and call-by-push-value [26, 29].

Closer to our work, non-idempotent intersection types have been used to characterise strong normalisation in a calculus with fix-point operators and pattern matching on constructors [12]. Similarly, a strong call-by-need strategy for a pattern matching language was defined in [9], and completeness of the strategy was shown by means of non-idempotent intersection types by extending the technique introduced in [36, 8]. In both cases, despite the use of non-idempotent types, the result was qualitative, as no quantitative results were obtained by means of the typing system.

Even closer to our work, [19] studied the solvability property in a pair pattern calculus, the main result being that solvability is equivalent to typability plus inhabitation in a non-idempotent intersection type system. One of the contributions of [19] is a characterisation of

(non-deterministic) head-normalisation by means of typability, which is merely *qualitative*, as it does not give any *upper bound/exact measure* for head-evaluation, as discussed in the previous subsection.

More practical type-based approaches (i.e. mostly sound but not necessarily complete) to quantitative analysis are sized-types [49, 54, 7, 45, 46] and (automatic) amortised resource analysis [33, 53, 30, 34, 31, 51].

Sized Types. This line of work is based on the use of types indexed by sizes, which are essentially ordinals. The approach is based on the fact that the compiler checks if the program is typed with the correct size, so that resource usages of programs can be derived from the sized types informations.

While space cost is determined in [54], upper bounds for both space and time costs are obtained in [49, 7]. In particular, sized types are used in [7] to obtain space bounds, while time bounds are computed by using a kind of clock, achieved by means of ticking monadic transformations, originally introduced in [55] as ticking monads to get time complexity for lazy languages. This is done for a call-by-value functional language with (a fixed set of) inductive datatypes, enriched with index polymorphism: functions can be polymorphic in their size annotations. In this respect, sized types enriched with intersection types have been used in [52] to handle time costs for a call-by-value strategy in a more restricted language. Sized types are also extended in [14] to guarantee termination of general higher-order rewriting. Yet another approach based on (first-order) size indices is given by linear dependent types [45, 46], where time and space bounds are obtained by establishing a relation with linear logic, a key tool used to define quantitative types through non-idempotent intersection types. Completeness depends on an oracle for the first order theory on indices describing the semantical properties of the function symbols, so the approach cannot be fully turned into an automatic tool.

Amortised resource analysis. This line of work is motivated by the fact that the worst-case run time analysis per operation, rather than per algorithm, can lead to very pessimistic complexity bounds. This is then replaced by an approach considering both the costly and less costly operations together over the whole set of operations of the algorithm. Automatisation of amortised resource analysis has been achieved in a series of works [33, 30, 53, 34, 31, 51], including space usage [53] and more general usages [31], all regarding a lazy functional language. The pioneer work in [30] has evolved to more sophisticated tools, leading today to RAML [50], a language applied to an industrial strength compiler [31]. Lazy evaluation is not handled in RAML, however, [51] proposes a practical tool to estimate resource usage for Haskell expressions.

Road-map: Sec. 2 introduces the pattern calculus. Sec. 3 presents the typing system $\mathscr{U}$, together with some of its quantitative properties, and Sec. 4 suggests a relational model for our pattern calculus based on the type system. In Sec. 5, we refine $\mathscr{U}$ to extract exact bounds, which leads to the definition of our second typing system $\mathscr{E}$. The soundness (resp. completeness) proof for $\mathscr{E}$ is given in Sec. 6 (resp. Sec. 7). Conclusions and future work are discussed in Sec. 8. All proofs are presented in the Appendix.

2 The Pattern Calculus

In this section we introduce the pattern calculus, an extension of the λ-calculus where abstraction is extended to *pair patterns* and terms are extended to *pairs*. We start by introducing the syntax of the calculus.

Terms and contexts of the pattern calculus are defined by means of the following grammars:

$$\begin{array}{lllll}
(\textbf{Patterns}) & p,q & ::= & x \mid \langle p,q\rangle \\
(\textbf{Terms}) & t,u,v & ::= & x \mid \lambda p.t \mid \langle t,u\rangle \mid tu \mid t[p\backslash u] \\
(\textbf{List Contexts}) & \mathtt{L} & ::= & \Box \mid \mathtt{L}[p\backslash u] \\
(\textbf{Contexts}) & \mathtt{C} & ::= & \Box \mid \lambda p.\mathtt{C} \mid \langle \mathtt{C},t\rangle \mid \langle t,\mathtt{C}\rangle \mid \mathtt{C}t \mid t\mathtt{C} \mid \mathtt{C}[p\backslash t] \mid t[p\backslash \mathtt{C}]
\end{array}$$

where $x, y, z, w \ldots$ range over a countable set of variables, and every pattern p is assumed to be *linear*, i.e. every variable appears at most once in p. The term x is called a **variable**, $\lambda p.t$ is an **abstraction**, $\langle t,u\rangle$ is a **pair**, tu is an **application** and $t[p\backslash u]$ is a **closure**, where $[p\backslash u]$ is an **explicit matching** operator. Special terms are $\mathtt{I} := \lambda z.z$, $\Delta := \lambda z.zz$ and $\Omega := \Delta\Delta$. As usual we use the abbreviation $\lambda p_1 \ldots p_n.t_1 \ldots t_m$ for $\lambda p_1(\ldots(\lambda p_n.((t_1t_2)\ldots t_m))\ldots)$, $n \geq 0$, $m \geq 1$.

We write $\mathtt{var}(p)$ to denote the variables in the pattern p. **Free** and **bound variables** of terms and contexts are defined as expected, in particular $\mathtt{fv}(\lambda p.t) := \mathtt{fv}(t) \setminus \mathtt{var}(p)$, $\mathtt{fv}(t[p\backslash u]) := (\mathtt{fv}(t) \setminus \mathtt{var}(p)) \cup \mathtt{fv}(u)$ and $\mathtt{bv}(\lambda p.t) := \mathtt{bv}(t) \cup \mathtt{var}(p)$, $\mathtt{bv}(t[p\backslash u]) := \mathtt{bv}(t) \cup \mathtt{var}(p) \cup \mathtt{bv}(u)$. We also define the **domain of a list context** as $\mathtt{dlc}(\Box) = \emptyset$ and $\mathtt{dlc}(\mathtt{L}[p\backslash u]) = \mathtt{dlc}(\mathtt{L}) \cup \mathtt{var}(p)$. We write $p\#q$ if $\mathtt{var}(p)$ and $\mathtt{var}(q)$ are disjoint. As usual, terms are considered modulo α-conversion, so that for example $\lambda\langle x,y\rangle.xz =_\alpha \lambda\langle x',y'\rangle.x'z$ and $x[\langle x,y\rangle\backslash z] =_\alpha x'[\langle x',y'\rangle\backslash z]$. Given a list context $\mathtt{L}$ and a term t, $\mathtt{L}[\![t]\!]$ denotes the term obtained by replacing the unique occurrence of $\Box$ in $\mathtt{L}$ by t, possibly allowing the capture of free variables of t. We use $t\{x\backslash u\}$ to denote the meta-level substitution operation which replaces all the free occurrences of x in t by the term u. As usual, this operation is performed modulo α-conversion so that capture of free variables is avoided. We use the predicate $\mathtt{abs}(t)$ when t is of the form $\mathtt{L}[\![\lambda p.u]\!]$. The **reduction relation** $\to_{\mathtt{p}}$ on terms is given by the closure over *all* contexts of the following rewriting rules.

$$\begin{array}{llll}
\mathtt{L}[\![\lambda p.t]\!]u & \mapsto & \mathtt{L}[\![t[p\backslash u]]\!] & \mathtt{dlc}(\mathtt{L}) \cap \mathtt{fv}(u) = \emptyset \\
t[\langle p_1,p_2\rangle\backslash \mathtt{L}[\![\langle u_1,u_2\rangle]\!]] & \mapsto & \mathtt{L}[\![t[p_1\backslash u_1][p_2\backslash u_2]]\!] & \mathtt{dlc}(\mathtt{L}) \cap \mathtt{fv}(t) = \emptyset \\
t[x\backslash u] & \mapsto & t\{x\backslash u\}
\end{array}$$

The reduction relation $\to_{\mathtt{p}}$ defined above is related to that in [19], called $\to_{\Lambda_{\mathtt{p}}}$, in the following sense: $\to_{\Lambda_{\mathtt{p}}}$ contains two subsystem relations, one to deal with *clashes*, which are not handled by the reduction system in the present calculus since we consider typable terms only (cf. Lem. 6), and another one containing the following five rules:

$$\begin{array}{llll}
(\lambda p.t)u & \mapsto & t[p\backslash u] & \\
t[\langle p_1,p_2\rangle\backslash\langle u_1,u_2\rangle] & \mapsto & t[p_1\backslash u_1][p_2\backslash u_2] & \\
t[x\backslash u] & \mapsto & t\{x\backslash u\} & \\
t[p\backslash v]u & \mapsto & (tu)[p\backslash v] & \mathtt{fv}(u) \cap \mathtt{var}(p) = \emptyset \\
t[\langle p_1,p_2\rangle\backslash u[q\backslash v]] & \mapsto & t[\langle p_1,p_2\rangle\backslash u][q\backslash v] & \mathtt{fv}(t) \cap \mathtt{var}(q) = \emptyset
\end{array}$$

The two last rules can be seen as commuting conversions, which are integrated in the first (two) rules of our reduction system $\to_{\mathtt{p}}$ by using the *substitution at a distance paradigm* [5]. It is worth noticing that $t \to_{\mathtt{p}} t'$ can be simulated by $t \to^+_{\Lambda_{\mathtt{p}}} t'$. For instance, $(\lambda p.t)[p_1\backslash u_1][p_2\backslash u_2]u \to_{\mathtt{p}} t[p\backslash u][p_1\backslash u_1][p_2\backslash u_2]$ can be simulated by:

$$\begin{array}{l}
(\lambda p.t)[p_1\backslash u_1][p_2\backslash u_2]u \to_{\Lambda_{\mathtt{p}}} ((\lambda p.t)[p_1\backslash u_1]u)[p_2\backslash u_2] \to_{\Lambda_{\mathtt{p}}} ((\lambda p.t)u)[p_1\backslash u_1][p_2\backslash u_2] \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \to_{\Lambda_{\mathtt{p}}} t[p\backslash u][p_1\backslash u_1][p_2\backslash u_2]
\end{array}$$

Our formulation of the pattern calculus at a distance, given by the relation $\to_{\mathtt{p}}$, as well as the corresponding head strategy that we present below, are one of the essential untyped tools used in this paper to get quantitative results about head-normalisation (cf. Sec. 3 and Sec. 5).

Although the reduction relation $\to_{\mathtt{p}}$ is *non-deterministic*, it can easily been shown to be *confluent*, for example using the same technique in [19]. However, in order to study exact bounds of evaluation, we need to define a *deterministic* strategy for the pattern calculus, i.e. a subrelation of $\to_{\mathtt{p}}$ that is able to compute the same normal forms. Fig. 1 gives an operational semantics for the pattern calculus, which turns out to be an extension of the well-known notion of *head-reduction* for λ-calculus, then also named **head-reduction**, and denoted by $\to_{\mathtt{h}}$. In the following inductive definition $t \to_{\mathtt{h}} u$ means that t head-reduces to u, and $t \not\to_{\mathtt{h}}$ means that t is a **head normal-form**, i.e. there is no u such that $t \to_{\mathtt{h}} u$.

$$\frac{\mathtt{dlc}(\mathtt{L}) \cap \mathtt{fv}(u) = \emptyset}{\mathtt{L}[\![\lambda p.t]\!]u \to_{\mathtt{h}} \mathtt{L}[\![t[p\backslash u]]\!]}\,(\mathtt{b}) \qquad \frac{t \not\to_{\mathtt{h}} \qquad \mathtt{dlc}(\mathtt{L}) \cap \mathtt{fv}(t) = \emptyset}{t[\langle p_1, p_2\rangle \backslash \mathtt{L}[\![\langle u_1, u_2\rangle]\!]] \to_{\mathtt{h}} \mathtt{L}[\![t[p_1\backslash u_1][p_2\backslash u_2]]\!]}\,(\mathtt{m}) \qquad \frac{t \not\to_{\mathtt{h}}}{t[x\backslash u] \to_{\mathtt{h}} t\{x\backslash u\}}\,(\mathtt{e})$$

$$\frac{t \to_{\mathtt{h}} t'}{\lambda p.t \to_{\mathtt{h}} \lambda p.t'} \qquad \frac{t \to_{\mathtt{h}} t' \qquad \neg\mathtt{abs}(t)}{tu \to_{\mathtt{h}} t'u} \qquad \frac{t \to_{\mathtt{h}} t'}{t[p\backslash u] \to_{\mathtt{h}} t'[p\backslash u]} \qquad \frac{t \not\to_{\mathtt{h}} \qquad p \neq x \qquad u \to_{\mathtt{h}} u'}{t[p\backslash u] \to_{\mathtt{h}} t[p\backslash u']}$$

Figure 1 The head-reduction strategy for the pattern calculus.

Rule `b` fires the computation of terms by transforming an application of a function to an argument into a closure term. Decomposition of patterns and terms is performed by means of rule `m`, when a pair pattern is matched against a pair term. Substitution is performed by rule `e`, i.e. an explicit (simple) matching of the form $[x\backslash u]$ is executed. This form of syntactic pattern matching is very simple, and does not consider any kind of failure result, but is already expressive enough to specify the well-known mechanism of successful matching. Context closure is *similar* to the call-by-name λ-calculus case, but not exactly the same. Indeed, head-reduction is performed on the left-hand side of applications and closures whenever possible. Otherwise, arguments of explicit matching operators must be head-reduced in order to unblock these operators, i.e. in order to decompose $[p\backslash u]$ when p is a pair pattern but u is still not a pair. Notice however that when u is already a pair, no head-reduction inside u can take place, thus implementing a *lazy* strategy for pattern matching. Standardisation of calculi as the one in this paper has been studied in [37].

Given any (one-step) reduction relation $\to_{\mathcal{R}}$, we use $\to^*_{\mathcal{R}}$, or more precisely $\to^k_{\mathcal{R}}$ $(k \geq 0)$ to denote the reflexive-transitive closure of $\to_{\mathcal{R}}$, i.e. the composition of k $\mathcal{R}$-steps. In the case of head-reduction, we may use the alternative notation $\to_{\mathtt{h}}^{(b,e,m)}$ to emphasize the number of reduction steps in a given reduction sequence, i.e. if $\rho : t \to_{\mathtt{h}}^{(b,e,m)} u$, then there are exactly b `b`-steps, e `e`-steps and m `m`-steps in the reduction sequence ρ. We will often use the notation $\to_{\mathtt{b}}$ to explicitly refer to a `b`-step (resp. $\to_{\mathtt{e}}$ and $\to_{\mathtt{m}}$ for `e` and `m` steps). The reduction relation $\to_{\mathtt{h}}$ is in fact a function:

▶ **Proposition 1.** *The relation $\to_{\mathtt{h}}$ is deterministic.*

▶ **Example 2.** Let us consider the combinators $\mathtt{I} := \lambda z.z$ and $\mathtt{K} := \lambda x_1.\lambda y_1.x_1$. Then we have $(\lambda\langle x,y\rangle.x(\mathtt{I}y))[z\backslash\mathtt{I}](\mathtt{I}\langle\mathtt{K},w\rangle) \to_{\mathtt{h}}^{(4,6,1)} \lambda y_1.w$:

$$\begin{array}{lll}
 & (\lambda\langle x,y\rangle.x(\mathtt{I}y))[z\backslash\mathtt{I}](\mathtt{I}\langle\mathtt{K},w\rangle) & \to_{\mathtt{b}} \quad (x(\mathtt{I}y))[\langle x,y\rangle\backslash\mathtt{I}\langle\mathtt{K},w\rangle][z\backslash\mathtt{I}] \\
\to_{\mathtt{b}} & (x(\mathtt{I}y))[\langle x,y\rangle\backslash z[z\backslash\langle\mathtt{K},w\rangle]][z\backslash\mathtt{I}] & \to_{\mathtt{e}} \quad (x(\mathtt{I}y))[\langle x,y\rangle\backslash\langle\mathtt{K},w\rangle][z\backslash\mathtt{I}] \\
\to_{\mathtt{m}} & (x(\mathtt{I}y))[x\backslash\mathtt{K}][y\backslash w][z\backslash\mathtt{I}] & \to_{\mathtt{e}} \quad (\mathtt{K}(\mathtt{I}y))[y\backslash w][z\backslash\mathtt{I}] \\
\to_{\mathtt{b}} & (\lambda y_1.x_1)[x_1\backslash\mathtt{I}y][y\backslash w][z\backslash\mathtt{I}] & \to_{\mathtt{e}} \quad (\lambda y_1.\mathtt{I}y)[y\backslash w][z\backslash\mathtt{I}] \\
\to_{\mathtt{b}} & (\lambda y_1.z[z\backslash y])[y\backslash w][z\backslash\mathtt{I}] & \to_{\mathtt{e}} \quad (\lambda y_1.y)[y\backslash w][z\backslash\mathtt{I}] \\
\to_{\mathtt{e}} & (\lambda y_1.w)[z\backslash\mathtt{I}] & \to_{\mathtt{e}} \quad \lambda y_1.w
\end{array}$$

Head normal-forms may contain ill-formed terms called *(head)* clashes not representing a desired result for a computation, i.e. (head) terms not syntactically well-formed. For example, a pair applied to another term $\langle u_1,u_2\rangle v$, or a matching between a pair pattern and a function $t[\langle p_1,p_2\rangle\backslash\lambda p.u]$ are considered to be (head) normal clashes. Formally, a term is said to be a **(head) clash** if it is generated by the following grammar:

$$\begin{array}{llll}
(\textbf{Head Clash}) & \mathtt{U} & ::= & c \mid \lambda p.\mathtt{U} \mid \mathtt{U}t \mid \mathtt{U}[p\backslash t] \mid t[\langle p_1,p_2\rangle\backslash\mathtt{U}] \\
(\textbf{Clash}) & c & ::= & \mathtt{L}[\![\langle u_1,u_2\rangle]\!]v \mid t[\langle p_1,p_2\rangle\backslash\mathtt{L}[\![\lambda p.u]\!]]
\end{array}$$

Then, a term t is said to be **(head) clash-free** if t does not head-reduce to a (head) clash, i.e. if there is no $u \in \mathtt{U}$ such that $t \to_{\mathtt{h}}^* u$. Remark in particular that every pair is (head) clash-free. A rewriting system raising a warning (i.e. a failure) when detecting a (head) clash has been defined in [19], allowing to restrict the attention to a smaller set of terms, called *canonical* terms, that are intended to be the (head) clash-free terms that are not reducible by the relation $\to_{\mathtt{h}}$. Canonical terms can be characterised inductively as follows:

$$\begin{array}{llll}
(\textbf{canonical forms}) & \mathcal{M} & ::= & \lambda p.\mathcal{M} \mid \langle t,t\rangle \mid \mathcal{M}[\langle p_1,p_2\rangle\backslash\mathcal{N}] \mid \mathcal{N} \\
(\textbf{pure canonical forms}) & \mathcal{N} & ::= & x \mid \mathcal{N}t \mid \mathcal{N}[\langle p_1,p_2\rangle\backslash\mathcal{N}]
\end{array}$$

In summary, canonical terms and irreducible terms are related as follows:

▶ **Proposition 3.** *$t \in \mathcal{M}$ if and only if t is (head) clash-free and $t \not\to_{\mathtt{h}}$.*

Size of canonical terms is given by: $|x| := 0$, $|\langle t,u\rangle| := 1$, $|\mathcal{N}t| := |\mathcal{N}| + 1$, $|\lambda p.\mathcal{M}| := |\mathcal{M}| + 1$, and $|\mathcal{M}[\langle p_1,p_2\rangle\backslash\mathcal{N}]| := |\mathcal{M}| + |\mathcal{N}| + 1$. As an example, the terms $\lambda\langle x,y\rangle.\langle x,\mathtt{I}\rangle$ and $\lambda x.y(\langle x,z\rangle\mathtt{I})$ are canonical forms of size 2 while $x\Omega$ and $z[\langle z,w\rangle\backslash x\Omega]$ are pure canonical terms of size 1 and 2 respectively. The term $\langle x,\mathtt{I}\rangle w$ is none of them, and the term $\mathtt{I}x$ can head-reduce to the canonical term x.

Finally, we define a term t to be **head-normalisable** if there exists a canonical form $u \in \mathcal{M}$ such that $t \to_{\mathtt{p}}^* u$. Moreover, t is said to be **head-terminating** if there exists a canonical form $u \in \mathcal{M}$ and an integer $k \geq 0$ such that $t \to_{\mathtt{h}}^k u$. The relation between the non-deterministic reduction relation $\to_{\mathtt{p}}$ and the deterministic strategy $\to_{\mathtt{h}}$ will be established later, but we can already say that, while t head-terminating immediately implies t head-normalisable, the completeness of the head-strategy w.r.t. head-normalisation is not trivial (Thm. 7).

3 The $\mathscr{U}$ Typing System

In this section we introduce our first typing system $\mathscr{U}$ for the pattern calculus. We start by defining the sets of **types** and **multiset types**, given by means of the following grammars:

$$\begin{array}{llll}
(\textbf{Product Types}) & \mathcal{P} & ::= & \times(\mathcal{A}_1,\mathcal{A}_2) \\
(\textbf{Types}) & \sigma & ::= & \bullet \mid \mathcal{P} \mid \mathcal{A} \to \sigma \\
(\textbf{Multiset Types}) & \mathcal{A} & ::= & [\sigma_k]_{k\in K}
\end{array}$$

where $\bullet$ is an atomic type, K is a (possibly empty) finite set of indexes, and a multiset type is an unordered list of (not necessarily different) elements, where $[\,]$ denotes the *empty* multiset. We write $|\mathcal{A}|$ to denote the number of elements of the multiset $\mathcal{A}$. For example $[\bullet, [\,] \to \bullet, \bullet]$ is a multiset type of 3 elements, representing the intersection type $(\bullet \cap ([\,] \to \bullet)) \cap \bullet$, where $\cap$ is an associative, commutative and non-idempotent intersection type constructor. We write $\sqcup$ to denote multiset union. Multiset types are used to specify how programs consume terms: intuitively, the empty multiset is assigned to terms that are erased during (head) reduction, while duplicable terms are necessarily typed with non-empty multisets. As usual the arrow type is right-associative.

A product type, representing the type of a pair, is defined as the product of two (possibly empty) multisets of types. This formulation of product types turns out to be a key tool in our quantitative framework, and constitutes an essential difference with the product types proposed in [19], which are modeled by disjoint unions, so that any pair $\langle t, u\rangle$ of *typed* terms t and u has necessarily *at least* two types, one of the form $\times_1(\sigma)$ where σ is the type of t, and one of the form $\times_2(\tau)$, where τ is the type of u. Indeed, in op. cit., multiset types carry two completely different meanings: being a pair (but not necessarily a pair to be duplicated), or being a duplicable term (but not necessarily a pair). Our specification of products can then be interpreted as the use of the exponential isomorphism $!(A \parr B) \equiv !A \otimes !B$ of multiplicative exponential linear logic [28].

A **typing context** Γ is a map from variables to multiset types, such that only finitely many variables are not mapped to the empty multiset $[\,]$. We write $\mathtt{dom}(\Gamma)$ to denote the domain of Γ, which is the set $\{x \mid \Gamma(x) \neq [\,]\}$. We may write $\Gamma \# \Delta$ if and only if $\mathtt{dom}(\Gamma)$ and $\mathtt{dom}(\Delta)$ are disjoint. Given typing contexts $\{\Gamma_i\}_{i\in I}$ we write $\wedge_{i\in I}\Gamma_i$ for the context that maps x to $\sqcup_{i\in I}\Gamma_i(x)$. One particular case is $\Gamma \wedge \Delta$. We sometimes write $\Gamma; \Delta$ instead of $\Gamma \wedge \Delta$, when $\Gamma \# \Delta$, and we do not distinguish $\Gamma; x : [\,]$ from Γ. The typing context $\Gamma|_p$ is such that $\Gamma|_p(x) = \Gamma(x)$, if $x \in \mathtt{var}(p)$ and $[\,]$ otherwise. The typing context $\Gamma \setminus\!\setminus \mathcal{V}$ is defined by $(\Gamma \setminus\!\setminus \mathcal{V})(x) = \Gamma(x)$ if $x \notin \mathcal{V}$ and $[\,]$ otherwise. Finally, $\Gamma \subseteq \Delta$ means that $\mathtt{dom}(\Gamma) \subseteq \mathtt{dom}(\Delta)$ and $\Gamma(x) \sqsubseteq \Delta(x)$ for every $x \in \mathtt{dom}(\Gamma)$, where $\sqsubseteq$ denotes multiset inclusion.

The **type assignment system** $\mathscr{U}$ is given in Fig. 2 and can be seen as a natural extension of Gardner's system [27] to explicit matching operators, pairs and product types. It assigns types (resp. multiset types) to terms, using an auxiliary (sub)system that assigns multiset types to patterns. We use $\Phi \triangleright \Gamma \vdash t : \sigma$ (resp. $\Phi \triangleright \Gamma \vdash t : \mathcal{A}$) to denote **term type derivations** ending with the sequent $\Gamma \vdash t : \sigma$ (resp. $\Gamma \vdash t : \mathcal{A}$), and $\Pi \triangleright \Gamma \Vdash p : \mathcal{A}$ to denote **pattern type derivations** ending with the sequent $\Gamma \Vdash p : \mathcal{A}$. The size of a derivation Φ, denoted by $\mathtt{sz}(\Phi)$, is the number of all the typing rules used in Φ except $\mathtt{many}$[1] (this is particularly appropriate in the proof of the substitution lemma).

Note that when assigning types (multiset types) to terms, we only allow the introduction of multiset types on the right through the $\mathtt{many}$ rule.

Most of the rules for terms are straightforward. Rule $\mathtt{match}$ is used to type the explicit matching operator $t[p \backslash u]$ and can be seen as a combination of rules $\mathtt{app}$ and $\mathtt{abs}$. Rule $\mathtt{pat}_{\mathtt{v}}$ is used when the pattern is a variable x. Its multiset type is the type declared for x in the typing context. Rule $\mathtt{pat}_{\times}$ is used when the pattern has a product type, which means that the pattern will be matched with a pair. The condition $p \# q$ ensures linearity of patterns. Note that any pair term can be typed, in particular, with $\times([\,], [\,])$.

The system enjoys the key property of relevance:

[1] An equivalent type system can be presented without the $\mathtt{many}$ rule, for example [19]. However, the inductive proofs in the current presentation turn to be more elegant.

$$\frac{}{x : \mathcal{A} \Vdash x : \mathcal{A}}\,(\mathtt{pat_v}) \qquad \frac{\Gamma \Vdash p : \mathcal{A} \quad \Delta \Vdash q : \mathcal{B} \quad p\#q}{\Gamma \wedge \Delta \Vdash \langle p, q\rangle : [\times(\mathcal{A},\mathcal{B})]}\,(\mathtt{pat}_\times)$$

$$\frac{}{x : [\sigma] \vdash x : \sigma}\,(\mathtt{ax}) \qquad \frac{(\Gamma_k \vdash t : \sigma_k)_{k\in K}}{\wedge_{k\in K}\Gamma_k \vdash t : [\sigma_k]_{k\in K}}\,(\mathtt{many})$$

$$\frac{\Gamma \vdash t : \sigma \quad \Gamma|_p \Vdash p : \mathcal{A}}{\Gamma \setminus\!\setminus \mathtt{var}(p) \vdash \lambda p.t : \mathcal{A} \to \sigma}\,(\mathtt{abs}) \qquad \frac{\Gamma \vdash t : \mathcal{A} \to \sigma \qquad \Delta \vdash u : \mathcal{A}}{\Gamma \wedge \Delta \vdash t\,u : \sigma}\,(\mathtt{app})$$

$$\frac{\Gamma \vdash t : \mathcal{A} \quad \Delta \vdash u : \mathcal{B}}{\Gamma \wedge \Delta \vdash \langle t, u\rangle : \times(\mathcal{A},\mathcal{B})}\,(\mathtt{pair}) \qquad \frac{\Gamma \vdash t : \sigma \quad \Gamma|_p \Vdash p : \mathcal{A} \quad \Delta \vdash u : \mathcal{A}}{(\Gamma \setminus\!\setminus \mathtt{var}(p)) \wedge \Delta \vdash t[p\backslash u] : \sigma}\,(\mathtt{match})$$

Figure 2 Typing System $\mathscr{U}$.

▶ **Lemma 4** (Relevance). *Let* $\Phi \triangleright \Gamma \vdash t : \sigma$. *Then,* $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(t)$.

Proof. By induction on Φ (cf. App. A). ◀

Moreover, typing is stable by reduction and expansion, and the size of derivations is decreasing (resp. strictly decreasing) for $\to_{\mathtt{p}}$ reduction (resp. $\to_{\mathtt{h}}$ reduction).

▶ **Lemma 5.** *Let* $\Phi \triangleright \Gamma \vdash t : \sigma$. *Then,*

1. (Upper Subject Reduction). $t \to_p t'$ *implies there is* $\Phi' \triangleright \Gamma \vdash t' : \sigma$ *s.t.* $\mathtt{sz}(\Phi) \geq \mathtt{sz}(\Phi')$, *and* $t \to_h t'$ *implies there is* $\Phi' \triangleright \Gamma \vdash t' : \sigma$ *s.t.* $\mathtt{sz}(\Phi) > \mathtt{sz}(\Phi')$.
2. (Upper Subject Expansion). $t' \to_p t$ *implies there is* $\Phi' \triangleright \Gamma \vdash t' : \sigma$ *such that* $\mathtt{sz}(\Phi') \geq \mathtt{sz}(\Phi)$ *and* $t' \to_h t$ *implies there is* $\Phi' \triangleright \Gamma \vdash t' : \sigma$ *such that* $\mathtt{sz}(\Phi') > \mathtt{sz}(\Phi)$.

Proof. By induction on Φ, item 1 (resp. item 2) uses a substitution (resp. anti-substitution) lemma (see Lem.22 and Lem. 23 in App. A for details). ◀

Typed terms are (head) clash-free, i.e. they cannot head reduce to a clash.

▶ **Lemma 6** (Clash-Free). *Let* $\Phi \triangleright \Gamma \vdash t : \sigma$. *Then* t *is (head) clash-free.*

Proof. By induction on Φ (cf. App. A). ◀

Although the system in [19] already characterises head-normalisation in the pattern calculus, it does not provide upper bounds for the length of the head strategy. This is mainly due to the fact that the reduction system in [19] does not always decrease the measure of the typed terms, even when reduction is performed in the so-called *typed* occurrences. We can recover this situation, as witnessed by the following soundness and completeness result:

▶ **Theorem 7** (Characterisation of Head-Normalisation and Upper Bounds). *Let* t *be a term in the pattern calculus. Then (1)* t *is typable in system* $\mathscr{U}$ *iff (2)* t *is head-normalisable iff (3)* t *is head-terminating. Moreover, if* $\Phi \triangleright \Gamma \vdash t : \sigma$, *then the head-strategy terminates on* t *in at most* $\mathtt{sz}(\Phi)$ *steps.*

Proof. The statement (1) ⇒ (3) holds by upper subject reduction (Lem. 5.1) for $\to_{\mathtt{h}}$. The statement (3) ⇒ (2) is straightforward since $\to_{\mathtt{h}}$ is included in $\to_{\mathtt{p}}$. Finally, the statement (2) ⇒ (1) holds by the fact that canonical terms are typable (easy), and by using upper subject expansion for $\to_{\mathtt{p}}$ (Lem. 5.2). ◀

The previous upper bound result is especially possible thanks to the upper subject reduction property, stating in particular that reduction $\to_{\mathtt{h}}$ *strictly* decreases the size of typing derivations. It is worth noticing that the reduction relation in [19] does not enjoy this property, particularly in the case of the rule $t[p\backslash v]u \to (tu)[p\backslash v]$, which is a permuting conversion rule, (slightly) changing the structure of the type derivation, but not its size.

4 Towards a Relational Model for the Pattern Calculus

Denotational and operational semantics have tended to abstract quantitative information (e.g. time and space) as computational resource consumption. Since the invention of Girard's linear logic [28], where formulas are interpreted as resources, quantitative interpretation of programs, such as relational models [17, 18, 22], have been naturally defined and studied by following the simple idea that multisets are used to record the number of times a resource is consumed. Thus, relational models for the λ-calculus use multisets to keep track of how many times a resource is used during a computation.

In this brief section we emphasize a semantical result that is implicit in the previous section. Since relational models are often presented by means of typing systems [48, 47], our system $\mathscr{U}$ suggests a quantitative model for our pair pattern calculus in the following way. Indeed, consider a term t such that $\mathtt{fv}(t) \subseteq \{x_1, \dots, x_n\}$, in which case we say that the list $\vec{x} = (x_1, \dots, x_n)$ is **suitable** for t. Then, given $\vec{x} = (x_1, \dots, x_n)$ suitable for t, define the interpretation of a term t for $\vec{x}$ as

$$[\![t]\!]_{\vec{x}} = \{((\mathcal{A}_1, \dots, \mathcal{A}_n), \sigma) \mid \text{ there exists } \Phi \rhd x_1 : \mathcal{A}_1, \dots, x_n : \mathcal{A}_n \vdash t : \sigma\}$$

A straightforward corollary of upper subject reduction and expansion properties (Lem. 5.1 and Lem. 5.2, respectively) is that $t =_{\mathtt{p}} u$ implies $[\![t]\!]_{\vec{x}} = [\![u]\!]_{\vec{x}}$, where $=_{\mathtt{p}}$ is the equational theory generated by the reduction relation $\to_{\mathtt{p}}$. Thus, p-equivalent programs have the same meaning.

5 The $\mathscr{E}$ Typing System

In this section we introduce our second typing system $\mathscr{E}$ for the pattern calculus, which is obtained by refining the System $\mathscr{U}$ presented in Sec. 3.

(**Product Types**)	$\mathcal{P}$	::=	$\times(\mathcal{A}_1, \mathcal{A}_2)$
(**Tight Types**)	$\mathtt{t}$	::=	$\bullet_{\mathcal{N}} \mid \bullet_{\mathcal{M}}$
(**Types**)	σ	::=	$\mathtt{t} \mid \mathcal{P} \mid \mathcal{A} \to \sigma$
(**Multiset Types**)	$\mathcal{A}$	::=	$[\sigma_k]_{k \in K}$

Types in $\mathtt{t}$, which can be seen as a refinement of the base type $\bullet$ of System $\mathscr{U}$, denote the so-called tight types. The constant $\bullet_{\mathcal{M}}$ denotes the type of any term head reducing to a canonical form, while $\bullet_{\mathcal{N}}$ denotes the type of any term head reducing to a pure canonical form. We write $\mathtt{tight}(\sigma)$, if σ is of the form $\bullet_{\mathcal{M}}$ or $\bullet_{\mathcal{N}}$ (we use $\bullet$ to denote either form). We extend this notion to multisets of types and typing contexts as expected, that is, $\mathtt{tight}([\sigma_i]_{i \in I})$ if $\mathtt{tight}(\sigma_i)$ for all $i \in I$, and $\mathtt{tight}(\Gamma)$ if $\mathtt{tight}(\Gamma(x))$, for all $x \in \mathtt{dom}(\Gamma)$.

The crucial idea behind the grammar of types is to distinguish between *consuming* constructors typed with standard types, and *persistent* constructors typed with tight types, as hinted in the introduction. A constructor is consuming (resp. persistent) if it is consumed (resp. not consumed) during head-reduction. Indeed, the pair constructor is consumed (on the pattern side as well as on the term side) during the execution of the pattern matching rule $\mathtt{m}$. Otherwise, patterns and pairs are persistent, and they do appear in the normal form of the original term. This dichotomy between consuming and persistent constructors, inspired from [41, 42], is reflected in the typing system by using different typing rules to type them, notably for the abstraction, the application, the pair terms and the pair patterns.

The **type assignment system** $\mathscr{E}$, given in Fig. 3, is based on sequents for terms (resp. patterns) with *counters* having the form $\Gamma \vdash^{(b,e,m,f)} t : \sigma$ or $\Gamma \vdash^{(b,e,m,f)} t : \mathcal{A}$ (resp. $\Gamma \Vdash^{(e,m,f)} p : \mathcal{A}$). Intuitively, if $\Gamma \vdash^{(b,e,m,f)} t : \sigma$ is "tightly" derivable (defined below), then $t \to_{\mathtt{h}}^{(b,e,m)} v$, where b is the number of $\mathtt{b}$-steps, e the number of $\mathtt{e}$-steps, m the number of $\mathtt{m}$-steps and f is the size of the head normal-form v. Similarly, the derivability of $\Gamma \Vdash^{(e,m,f)} p : \mathcal{A}$ means that the pattern p generates e substitution $\mathtt{e}$-steps, m matching $\mathtt{m}$-steps and f symbols contributing to the normal form.

We write $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \sigma$ (resp. $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \mathcal{A}$) to denote **term type derivations** ending with the sequent $\Gamma \vdash^{(b,e,m,f)} t : \sigma$ (resp. $\Gamma \vdash^{(b,e,m,f)} t : \mathcal{A}$), and $\Pi \triangleright \Gamma \Vdash^{(e,m,f)} p : \mathcal{A}$ to denote **pattern type derivations** ending with the sequent $\Gamma \Vdash^{(e,m,f)} p : \mathcal{A}$. Often in examples, we will use the notation $\Phi^{(b,e,m,f)}$ (resp. $\Pi^{(e,m,f)}$) to refer to a term derivation (resp. pattern derivation) ending with a sequent annotated with indexes (b, e, m, f) (resp. (e, m, f)).

As mentioned in the introduction, exact bounds can only be extractable from *minimal* derivations. In our framework this notion is implemented by means of tightness [2]. We say that a derivation $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \sigma$ (resp. $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \mathcal{A}$) is **tight**, denoted by $\mathtt{tight}(\Phi)$, if and only if $\mathtt{tight}(\Gamma)$ and $\mathtt{tight}(\sigma)$ (resp. $\mathtt{tight}(\mathcal{A})$). The size of derivations is defined as in System $\mathscr{U}$.

We now give some intuition behind the typing rules in Fig. 3, by addressing in particular the consuming/persistent paradigm.

- Rule $\mathtt{ax}$: Since x is itself a head normal-form, it will not generate any $\mathtt{b}$, $\mathtt{e}$ or $\mathtt{m}$ steps, and its size is 0.
- Rule $\mathtt{abs}$: Used to type abstractions $\lambda p.t$ to be applied (i.e. consumed), therefore it has a functional type $\mathcal{A} \to \sigma$. Final indexes of the abstraction are obtained from the ones of the body and the pattern, and 1 is added to the first index since the abstraction will be consumed by a $\mathtt{b}$-reduction step.
- Rule $\mathtt{abs}_{\mathtt{p}}$: Used to type abstractions $\lambda p.t$ that are not going to be applied/consumed (they are persistent). Only the last index (size of the normal form) is incremented by one since the abstraction remains in the normal form (the abstraction is persistent). Note that both the body t and the variables in p should be typed with a tight type.
- Rule $\mathtt{app}$: Types applications tu where t will eventually become an abstraction, and thus the application constructor will be consumed. Indexes for tu are exactly the sum of the indexes for t and u. Note that we do not need to increment the counter for $\mathtt{b}$ steps, since this was already taken into account in the $\mathtt{abs}$ rule.
- Rule $\mathtt{app}_{\mathtt{p}}$: Types applications tu where t is neutral, therefore will never become an abstraction, and the application constructor becomes persistent. Indexes are the ones for t, adding one to the (normal term) size to count for the (persistent) application.
- Rule $\mathtt{pair}$: Types pairs consumed during some matching step. We add the indexes for the two components of the pair without incrementing the number of $\mathtt{m}$ steps, since it is incremented when typing a consuming abstraction, with rule $\mathtt{abs}$.

$$\frac{}{x : \mathcal{A} \Vdash^{(1,0,0)} x : \mathcal{A}}\ (\mathtt{pat_v})$$

$$\frac{\Gamma \Vdash^{(e_p,m_p,n_p)} p : \mathcal{A} \qquad \Delta \Vdash^{(e_q,m_q,n_q)} q : \mathcal{B} \qquad p \# q}{\Gamma \wedge \Delta \Vdash^{(e_p+e_q,1+m_p+m_q,n_p+n_q)} \langle p, q\rangle : [\times(\mathcal{A},\mathcal{B})]}\ (\mathtt{pat}_\times)$$

$$\frac{\mathtt{dom}(\Gamma) \subseteq \mathtt{var}(\langle p, q\rangle) \qquad \mathtt{tight}(\Gamma)}{\Gamma \Vdash^{(0,0,1)} \langle p, q\rangle : [\bullet_{\mathcal{N}}]}\ (\mathtt{pat_p})$$

$$\frac{}{x : [\sigma] \vdash^{(0,0,0,0)} x : \sigma}\ (\mathtt{ax})$$

$$\frac{\Gamma \vdash^{(b_t,e_t,m_t,f_t)} t : \sigma \qquad \Gamma|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{A}}{\Gamma \setminus\!\setminus \mathtt{var}(p) \vdash^{(b_t+1,e_t+e_p,m_t+m_p,f_t+f_p)} \lambda p.t : \mathcal{A} \to \sigma}\ (\mathtt{abs})$$

$$\frac{\Gamma \vdash^{(b,e,m,f)} t : \mathtt{t} \qquad \mathtt{tight}(\Gamma|_p)}{\Gamma \setminus\!\setminus \mathtt{var}(p) \vdash^{(b,e,m,f+1)} \lambda p.t : \bullet_{\mathcal{M}}}\ (\mathtt{abs_p})$$

$$\frac{(\Gamma_k \vdash^{(b_k,e_k,m_k,f_k)} t : \sigma_k)_{k\in K}}{\wedge_{k\in K}\Gamma_k \vdash^{(+_{k\in K}b_k,+_{k\in K}e_k,+_{k\in K}m_k,+_{k\in K}f_k)} t : [\sigma_k]_{k\in K}}\ (\mathtt{many})$$

$$\frac{\Gamma \vdash^{(b_t,e_t,m_t,f_t)} t : \mathcal{A} \to \sigma \qquad \Delta \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}}{\Gamma \wedge \Delta \vdash^{(b_t+b_u,e_t+e_u,m_t+m_u,f_t+f_u)} t\, u : \sigma}\ (\mathtt{app})$$

$$\frac{\Gamma \vdash^{(b_t,e_t,m_t,f_t)} t : \bullet_{\mathcal{N}}}{\Gamma \vdash^{(b_t,e_t,m_t,f_t+1)} t\, u : \bullet_{\mathcal{N}}}\ (\mathtt{app_p})$$

$$\frac{\Gamma \vdash^{(b_t,e_t,m_t,f_t)} t : \mathcal{A} \qquad \Delta \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{B}}{\Gamma \wedge \Delta \vdash^{(b_t+b_u,e_t+b_u,m_t+m_u,f_t+f_u)} \langle t, u\rangle : \times(\mathcal{A},\mathcal{B})}\ (\mathtt{pair})$$

$$\frac{}{\vdash^{(0,0,0,1)} \langle t, u\rangle : \bullet_{\mathcal{M}}}\ (\mathtt{pair_p})$$

$$\frac{\Gamma \vdash^{(b_t,e_t,m_t,f_t)} t : \sigma \qquad \Gamma|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{A} \qquad \Delta \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}}{(\Gamma \setminus\!\setminus \mathtt{var}(p)) \wedge \Delta \vdash^{(b_t+b_u,e_t+e_u+e_p,m_t+m_u+m_p,f_t+f_u+f_p)} t[p\backslash u] : \sigma}\ (\mathtt{match})$$

Figure 3 Typing System $\mathscr{E}$.

- Rule $\mathtt{pair_p}$: Used to type pairs that are not consumed in a matching step (they are persistent), therefore appear in the head normal-form. Since the pair is already a head normal-form its indexes are zero except for the size, which counts the pair itself.
- Rule `match`: Note that we do not need separate cases for consuming and persistent explicit matchings, since in both cases typable occurrences of u represent potential head reduction steps for u, which need to be taken into account in the final counter of the term.

- Rule $\mathtt{pat_v}$: Typed variables always generate one $\mathtt{e}$ and zero $\mathtt{m}$ steps, even when erased.
- Rule $\mathtt{pat_\times}$: Used when the pattern has a product type, which means that the pattern will be matched with a pair. We add the counters for the two components of the pair and increment the counter for the $\mathtt{m}$ steps.
- Rule $\mathtt{pat_p}$: Used when the pattern has a tight type, which means that it will not be matched with a pair and therefore will be blocked (it is persistent). This kind of pairs generate zero $\mathtt{e}$ and $\mathtt{m}$ steps, and will contribute with one blocked pattern to the size of the normal form.

The system also enjoys the relevance and clash-free properties, easily proved by induction:

▶ **Lemma 8** (Relevance). *Let $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \sigma$. Then, $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(t)$.*

▶ **Lemma 9** (Clash-Free). *Let $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \sigma$. Then, t is (head) clash-free.*

We now discuss two examples.

▶ **Example 10.** Let us consider $t_0 = (\lambda\langle x, y\rangle.(\lambda\langle w, z\rangle.wyz)x)\langle\langle \mathtt{K}, a\rangle, b\rangle$, with the following head-reduction sequence:

$$
\begin{array}{lll}
 & (\lambda\langle x,y\rangle.(\lambda\langle w,z\rangle.wyz)x)\langle\langle \mathtt{K},a\rangle,b\rangle & \to_{\mathtt{b}} \quad ((\lambda\langle w,z\rangle.wyz)x)[\langle x,y\rangle\backslash\langle\langle \mathtt{K},a\rangle,b\rangle] \\
\to_{\mathtt{b}} & (wyz)[\langle w,z\rangle\backslash x][\langle x,y\rangle\backslash\langle\langle \mathtt{K},a\rangle,b\rangle] & \to_{\mathtt{m}} \quad (wyz)[\langle w,z\rangle\backslash x][x\backslash\langle \mathtt{K},a\rangle][y\backslash b] \\
\to_{\mathtt{e}} & (wyz)[\langle w,z\rangle\backslash\langle \mathtt{K},a\rangle][y\backslash b] & \to_{\mathtt{m}} \quad (wyz)[w\backslash \mathtt{K}][z\backslash a][y\backslash b] \\
\to_{\mathtt{e}} & (\mathtt{K}yz)[z\backslash a][y\backslash b] & \to_{\mathtt{b}} \quad ((\lambda y_1.x_1)[x_1\backslash y]z)[z\backslash a][y\backslash b] \\
\to_{\mathtt{b}} & x_1[y_1\backslash z][x_1\backslash y][z\backslash a][y\backslash b] & \to_{\mathtt{e}} \quad x_1[x_1\backslash y][z\backslash a][y\backslash b] \\
\to_{\mathtt{e}} & y[z\backslash a][y\backslash b] & \to_{\mathtt{e}} \quad y[y\backslash b] \\
\to_{\mathtt{e}} & b &
\end{array}
$$

Note that, there are two matching steps in the head-reduction sequence, but the second step is only created after the substitution of x by $\langle \mathtt{K}, a\rangle$. Our method allows us to extract this information from the typing derivations because of the corresponding types for $\langle x, y\rangle$ and $\langle w, z\rangle$. Indeed, both patterns are typed with a product type (cf. the forthcoming tight typing derivations), and therefore the corresponding pairs are consumed and not persistent.

Since $t_0 = (\lambda\langle x, y\rangle.(\lambda\langle w, z\rangle.wyz)x)\langle\langle \mathtt{K}, a\rangle, b\rangle \to_{\mathtt{h}}^{(4,6,2)} b$, the term t_0 should be tightly typable with counter $(4, 6, 2, 0)$, where 0 is the size of b. In the construction of such tight derivation we proceed by pieces. Let $T_{\mathtt{K}} = [\bullet_{\mathcal{N}}] \to [\,] \to \bullet_{\mathcal{N}}$. We first construct the following pattern derivation for $\langle w, z\rangle$:

$$
\Pi_{\langle w,z\rangle} \triangleright \quad \frac{w : [T_{\mathtt{K}}] \Vdash^{(1,0,0)} w : [T_{\mathtt{K}}] \qquad \Vdash^{(1,0,0)} z : [\,]}{w : [T_{\mathtt{K}}] \Vdash^{(2,1,0)} \langle w, z\rangle : [\times([T_{\mathtt{K}}], [\,])]}
$$

In the following $T_{\langle w,z\rangle} = [\times([T_{\mathtt{K}}], [\,])]$. We construct a similar pattern derivation for $\langle x, y\rangle$:

$$
\Pi_{\langle x,y\rangle} \triangleright \quad \frac{x : T_{\langle w,z\rangle} \Vdash^{(1,0,0)} x : T_{\langle w,z\rangle} \qquad y : [\bullet_{\mathcal{N}}] \Vdash^{(1,0,0)} y : [\bullet_{\mathcal{N}}]}{x : T_{\langle w,z\rangle}; y : [\bullet_{\mathcal{N}}] \Vdash^{(2,1,0)} \langle x, y\rangle : [\times(T_{\langle w,z\rangle}, [\bullet_{\mathcal{N}}])]}
$$

In the rest of the example $T_{\langle x,y\rangle} = [\times(T_{\langle w,z\rangle}, [\bullet_{\mathcal{N}}])]$. We build a type derivation for $\lambda\langle x, y\rangle.(\lambda\langle w, z\rangle.wyz)x$, where $\Gamma_w = w : [T_{\mathtt{K}}]$, $\Gamma_y = y : [\bullet_{\mathcal{N}}]$, $\Gamma = \Gamma_w; \Gamma_y$, and $\Gamma_x = x : T_{\langle w,z\rangle}$. Furthermore, in this example and throughout the paper, we will use $(\overline{0})$ to denote the tuple $(0, 0, 0, 0)$.

$$\Phi_1 \triangleright \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_w \vdash^{(\bar{0})} w : T_{\mathtt{K}} \quad \Gamma_y \vdash^{(\bar{0})} y : [\bullet_{\mathcal{N}}]}{\Gamma \vdash^{(\bar{0})} wy : [\,] \to \bullet_{\mathcal{N}}} \quad \vdash^{(\bar{0})} z : [\,]}{\Gamma \vdash^{(\bar{0})} wyz : \bullet_{\mathcal{N}}}\,\Pi^{(2,1,0)}_{\langle w,z\rangle}}{\Gamma_y \vdash^{(1,2,1,0)} \lambda\langle w,z\rangle.wyz : T_{\langle w,z\rangle} \to \bullet_{\mathcal{N}}} \quad \Gamma_x \vdash^{(\bar{0})} x : T_{\langle w,z\rangle}}{\Gamma_y;\Gamma_x \vdash^{(1,2,1,0)} (\lambda\langle w,z\rangle.wyz)x : \bullet_{\mathcal{N}}}\,\Pi^{(2,1,0)}_{\langle x,y\rangle}}{\vdash^{(2,4,2,0)} \lambda\langle x,y\rangle.(\lambda\langle w,z\rangle.wyz)x : T_{\langle x,y\rangle} \to \bullet_{\mathcal{N}}}$$

$$\Phi_{\mathtt{K}} \triangleright \dfrac{\dfrac{x_1 : [\bullet_{\mathcal{N}}] \vdash^{(\bar{0})} x_1 : \bullet_{\mathcal{N}} \quad \Vdash^{(1,0,0)} y_1 : [\,]}{x_1 : [\bullet_{\mathcal{N}}] \vdash^{(1,1,0,0)} \lambda y_1.x_1 : [\,] \to \bullet_{\mathcal{N}}} \quad x_1 : [\bullet_{\mathcal{N}}] \Vdash^{(1,0,0)} x_1 : [\bullet_{\mathcal{N}}]}{\vdash^{(2,2,0,0)} \mathtt{K} : T_{\mathtt{K}}}$$

From Φ_1 and $\Phi_{\mathtt{K}}$ we build the final tight derivation for $t_0 = (\lambda\langle x,y\rangle.(\lambda\langle w,z\rangle.wyz)x)\langle\langle \mathtt{K},a\rangle,b\rangle$:

$$\Phi \triangleright \dfrac{\Phi_1^{(2,4,2,0)} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Phi_{\mathtt{K}}^{(2,2,0,0)}}{\vdash^{(2,2,0,0)} \mathtt{K} : [T_{\mathtt{K}}]} \quad \vdash^{(\bar{0})} a : [\,]}{\vdash^{(2,2,0,0)} \langle \mathtt{K},a\rangle : \times([T_{\mathtt{K}}],[\,])}}{\vdash^{(2,2,0,0)} \langle \mathtt{K},a\rangle : T_{\langle w,z\rangle}} \quad b : [\bullet_{\mathcal{N}}] \vdash^{(\bar{0})} b : [\bullet_{\mathcal{N}}]}{b : [\bullet_{\mathcal{N}}] \vdash^{(2,2,0,0)} \langle\langle \mathtt{K},a\rangle,b\rangle : \times(T_{\langle w,z\rangle},[\bullet_{\mathcal{N}}])}}{b : [\bullet_{\mathcal{N}}] \vdash^{(2,2,0,0)} \langle\langle \mathtt{K},a\rangle,b\rangle : [\times(T_{\langle w,z\rangle},[\bullet_{\mathcal{N}}])]}}{b : [\bullet_{\mathcal{N}}] \vdash^{(4,6,2,0)} (\lambda\langle x,y\rangle.(\lambda\langle w,z\rangle.wyz)x)\langle\langle \mathtt{K},a\rangle,b\rangle : \bullet_{\mathcal{N}}}$$

Therefore, $\Phi^{(4,6,2,0)}$ gives the expected exact bounds. It is worth noticing that the pair $\langle\langle \mathtt{K},a\rangle,b\rangle$ is typed here with a singleton multiset, while it would be typable with a multiset having at least two elements in the typing system proposed in [19], even if the term is not going to be duplicated.

▶ **Example 11.** We now consider the term $t_1 = (\lambda z.(\lambda\langle x,y\rangle.\mathtt{I})zz)\langle u,v\rangle$, having the following head-reduction sequence to head normal-form:

$$\begin{array}{lll}
 & (\lambda z.(\lambda\langle x,y\rangle.\mathtt{I})zz)\langle u,v\rangle & \to_{\mathtt{b}} \quad ((\lambda\langle x,y\rangle.\mathtt{I})zz)[z\backslash\langle u,v\rangle] \\
\to_{\mathtt{b}} & (\mathtt{I}[\langle x,y\rangle\backslash z]z)[z\backslash\langle u,v\rangle] & \to_{\mathtt{b}} \quad w[w\backslash z][\langle x,y\rangle\backslash z][z\backslash\langle u,v\rangle] \\
\to_{\mathtt{e}} & z[\langle x,y\rangle\backslash z][z\backslash\langle u,v\rangle] & \to_{\mathtt{e}} \quad \langle u,v\rangle[\langle x,y\rangle\backslash\langle u,v\rangle] \\
\to_{\mathtt{m}} & \langle u,v\rangle[x\backslash u][y\backslash v] & \to_{\mathtt{e}} \quad \langle u,v\rangle[y\backslash v] \\
\to_{\mathtt{e}} & \langle u,v\rangle &
\end{array}$$

We have 3 `b`-steps, 4 `e`-steps, and 1 `m`-step to the normal form $\langle u,v\rangle$ of size 1. Note that the pair $\langle u,v\rangle$ is copied twice during the reduction, but only one of the copies is consumed by a matching. The copy of the pair that is not consumed will persist in the term, therefore it will be typed with $\bullet_{\mathcal{M}}$. The other copy will be consumed in a matching step, however its components are not going to be used, therefore we will type it with $[o]$, where o denotes $\times([\,],[\,])$.

Since $t_1 = (\lambda z.(\lambda\langle x,y\rangle.\mathtt{I})zz)\langle u,v\rangle \to_{\mathtt{h}}^{(3,4,1)} \langle u,v\rangle$, we need to derive a tight derivation for t_1 decorated with counter $(3,4,1,1)$. We first consider the following derivation:

$$\Phi_1 \triangleright \dfrac{\dfrac{w : [\bullet_{\mathcal{M}}] \vdash^{(\bar{0})} w : \bullet_{\mathcal{M}} \quad w : [\bullet_{\mathcal{M}}] \Vdash^{(1,0,0)} w : [\bullet_{\mathcal{M}}]}{\vdash^{(1,1,0,0)} \mathtt{I} : [\bullet_{\mathcal{M}}] \to \bullet_{\mathcal{M}}} \quad \dfrac{\Vdash^{(1,0,0)} x : [\,] \quad \Vdash^{(1,0,0)} y : [\,]}{\Vdash^{(2,0,0)} \langle x,y\rangle : [o]}}{\vdash^{(2,3,1,0)} \lambda\langle x,y\rangle.\mathtt{I} : [o] \to [\bullet_{\mathcal{M}}] \to \bullet_{\mathcal{M}}}$$

From Φ_1 we obtain the following derivation Φ_2, where $\mathcal{A}_0 = [o, \bullet_{\mathcal{M}}]$:

$$\Phi_2 \triangleright \dfrac{\dfrac{\dfrac{\Phi_1^{(2,3,1,0)} \quad z : [o] \vdash^{(\bar{0})} z : [o]}{z : [o] \vdash^{(2,3,1,0)} (\lambda\langle x, y\rangle.\mathtt{I})z : [\bullet_{\mathcal{M}}] \to \bullet_{\mathcal{M}}} \quad z : [\bullet_{\mathcal{M}}] \vdash^{(\bar{0})} z : [\bullet_{\mathcal{M}}]}{z : \mathcal{A}_0 \vdash^{(2,3,1,0)} (\lambda\langle x, y\rangle.\mathtt{I})zz : \bullet_{\mathcal{M}}} \quad z : \mathcal{A}_0 \Vdash^{(1,0,0)} z : \mathcal{A}_0}{\vdash^{(3,4,1,0)} \lambda z.(\lambda\langle x, y\rangle.\mathtt{I})zz : \mathcal{A}_0 \to \bullet_{\mathcal{M}}}$$

Using Φ_2 we obtain the final tight derivation, and its expected counter:

$$\dfrac{\Phi_2^{(3,4,1,0)} \quad \dfrac{\dfrac{\vdots}{\vdash^{(\bar{0})} \langle u, v\rangle : o} \quad \vdash^{(0,0,0,1)} \langle u, v\rangle : \bullet_{\mathcal{M}}}{\vdash^{(0,0,0,1)} \langle u, v\rangle : \mathcal{A}_0}}{\vdash^{(3,4,1,1)} (\lambda z.(\lambda\langle x, y\rangle.\mathtt{I})zz)\langle u, v\rangle : \bullet_{\mathcal{M}}}$$

6 Soundness of System $\mathscr{E}$

This section studies the implication "tight typability implies head-normalisable". The two key properties used to show this implication are *minimal counters for canonical forms* (Lem. 13) and the *exact subject reduction property* (Lem. 15). Indeed, Lem. 13 guarantees that a tight derivation for a canonical form t holds the right counter of the form $(0, 0, 0, |t|)$. Lem. 15 gives in fact an *(exact) weighted subject reduction* property, weighted because head-reduction strictly decreases the counters of typed terms, and exact because only *one* counter is decreased by 1 for each head-reduction step. Subject reduction is based on a *substitution* property (Lem. 14). We start with a key auxiliary lemma.

▶ **Lemma 12** (Tight Spreading). *Let $t \in \mathcal{N}$. Let $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \sigma$ be a typing derivation such that $\mathtt{tight}(\Gamma)$. Then σ is tight and the last rule of Φ does not belong to $\{\mathtt{app}, \mathtt{abs}, \mathtt{abs}_{\mathtt{p}}, \mathtt{pair}, \mathtt{pair}_{\mathtt{p}}\}$.*

Proof. By induction on $t \in \mathcal{N}$, taking into account the fact that t is not an abstraction nor a pair (cf. App. B). ◀

▶ **Lemma 13** (Canonical Forms and Minimal Counters). *Let $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \sigma$ be a tight derivation. Then $t \in \mathcal{M}$ if and only if $b = e = m = 0$.*

Proof. The left-to-right implication is by induction on the definition of the set $\mathcal{M}$, using the tight spreading property (Lem. 12) for the cases of application and explicit matching. The right-to-left implication is by induction on Φ and also uses Lem. 12 (cf. App. B). ◀

▶ **Lemma 14** (Substitution for System $\mathscr{E}$). *If $\Phi_t \triangleright \Gamma; x : \mathcal{A} \vdash^{(b_t,e_t,m_t,f_t)} t : \sigma$, and $\Phi_u \triangleright \Delta \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$, then there exists $\Phi_{t\{x\backslash u\}} \triangleright \Gamma \wedge \Delta \vdash^{(b_t+b_u,e_t+e_u,m_t+m_u,f_t+f_u)} t\{x\backslash u\} : \sigma$.*

Proof. By induction on Φ_t (cf. App. B). ◀

▶ **Lemma 15** (Exact Subject Reduction). *If $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \sigma$, and $t \to_{\mathtt{h}} t'$ is an $\mathtt{s}$-step, with $\mathtt{s} \in \{\mathtt{b}, \mathtt{e}, \mathtt{m}\}$, then $\Phi' \triangleright \Gamma \vdash^{(b',e',m',f)} t' : \sigma$, where*

- *$\mathtt{s} = \mathtt{b}$ implies $b' = b - 1$, $e' = e$, $m' = m$.*
- *$\mathtt{s} = \mathtt{e}$ implies $b' = b$, $e' = e - 1$, $m' = m$.*
- *$\mathtt{s} = \mathtt{m}$ implies $b' = b$, $e' = e$, $m' = m - 1$.*

Proof. By induction on $\to_{\mathtt{h}}$, using the substitution property (Lem. 14) (cf. App. B). ◀

The exact subject reduction property provides a simple argument to obtain the implication "tightly typable implies head-normalisable": if t is tightly typable, and reduction decreases the counters, then head-reduction necessarily terminates. But the soundness implication is in fact more precise than that. Indeed:

▶ **Theorem 16** (Soundness). *Let $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \sigma$ be a tight derivation. Then there exists $u \in \mathcal{M}$ and a head reduction sequence ρ such that $\rho : t \to_h^{(b,e,m)} u$ and $|u| = f$.*

Proof. By induction on $b + e + m$.

If $b+e+m = 0$ (i.e. $b = e = m = 0$), then canonical forms and minimal counters property (Lem. 13) gives $t \in \mathcal{M}$, so that $t \not\to_{\mathtt{h}}$ holds by Prop. 3. We let $u := t$ and thus $t \to_{\mathtt{h}}^{(0,0,0)} t$. It is easy to show that tight derivations $\Phi \triangleright \Gamma \vdash^{(0,0,0,f)} t : \sigma$ for terms in $\mathcal{M}$ verify $|t| = f$.

If $b+e+m > 0$, we know by Lem. 13 that $t \notin \mathcal{M}$, and we know by the clash-free property (Lem. 9) that t is (head) clash-free. Then, t turns to be head-reducible by Prop. 3, i.e. there exists t' such that $t \to_{\mathtt{h}} t'$. By the exact subject reduction property (Lem. 15) there is a derivation $\Phi' \triangleright \Gamma \vdash^{(b',e',m',f)} t' : \sigma$ such that $b'+e'+m'+1 = b+e+m$. The i.h. applied to Φ' then gives $t' \to_{\mathtt{h}}^{(b',e',m')} u$ and $|u| = f$. We conclude with the sequence $t \to_{\mathtt{h}} t' \to_{\mathtt{h}}^{(b',e',m')} u$, with the counters as expected. ◀

7 Completeness for System $\mathscr{E}$

In this section we study the reverse implication "head-normalisable implies tight typability". In this case the key properties are the existence of *tight derivations for canonical forms* (Lem. 17) and the *subject expansion property* (Lem. 19). As in the previous section these properties are *(exact) weighted* in the sense that Lem. 17 guarantees that a canonical form t has a tight derivation with the right counter, and Lem. 19 shows that each step of head-expansion strictly increases exactly one of the counters of tightly typed terms. Subject expansion relies on an *anti-substitution* property (Lem. 18).

▶ **Lemma 17** (Canonical Forms and Tight Derivations). *Let $t \in \mathcal{M}$. There exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(0,0,0,|t|)} t : \mathtt{t}$.*

Proof. We generalise the property to the two following statements:

- If $t \in \mathcal{N}$, then there exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(0,0,0,|t|)} t : \bullet_{\mathcal{N}}$.
- If $t \in \mathcal{M}$, then there exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(0,0,0,|t|)} t : \mathtt{t}$.

The proof then proceeds by induction on $\mathcal{N}, \mathcal{M}$, using relevance (Lem. 8). ◀

▶ **Lemma 18** (Anti-Substitution for System $\mathscr{E}$). *Let $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t\{x\backslash u\} : \sigma$. Then, there exist derivations Φ_t, Φ_u, integers $b_t, b_u, e_t, e_u, m_t, m_u, f_t, f_u$, contexts Γ_t, Γ_u, and multitype $\mathcal{A}$ such that $\Phi_t \triangleright \Gamma_t; x : \mathcal{A} \vdash^{(b_t,e_t,m_t,f_t)} t : \sigma$, $\Phi_u \triangleright \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$, $b = b_t + b_u$, $e = e_t + e_u$, $m = m_t + m_u$, $f = f_t + f_u$, and $\Gamma = \Gamma_t \wedge \Gamma_u$.*

Proof. By induction on Φ (cf. App. C). ◀

▶ **Lemma 19** (Exact Subject Expansion). *If $\Phi' \triangleright \Gamma \vdash^{(b',e',m',f')} t' : \sigma$, and $t \to_h t'$ is an $\mathtt{s}$-step, with $\mathtt{s} \in \{\mathtt{b}, \mathtt{e}, \mathtt{m}\}$, then $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \sigma$, where*

- *$\mathtt{s} = \mathtt{b}$ implies $b = b' + 1$, $e' = e$, $m' = m$.*
- *$\mathtt{s} = \mathtt{e}$ implies $b' = b$, $e = e' + 1$, $m' = m$.*
- *$\mathtt{s} = \mathtt{m}$ implies $b' = b$, $e' = e$, $m = m' + 1$.*

Proof. By induction on $\to_{\mathtt{h}}$, using the anti-substitution property (Lem. 18) (cf. App. C). ◀

The previous lemma provides a simple argument to obtain the implication "head-normalisable implies tightly typable", which can in fact be stated in a more precise way:

▶ **Theorem 20** (Completeness). *Let t be a head-normalising term such that $t \to_h^{(b,e,m)} u$, $u \in \mathcal{M}$. Then there exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(b,e,m,|u|)} t : \mathtt{t}$.*

Proof. By induction on $b + e + m$.

- If $(b + e + m) = 0$ then $t = u \in \mathcal{M}$, therefore $\Gamma \vdash^{(0,0,0,|t|)} t : \mathtt{t}$, by the canonical forms and tight derivations property (Lem. 17).
- If $(b + e + m) > 0$, then $t \to_{\mathtt{h}} t' \to_{\mathtt{h}}^{(b',e',m')} u$, where $b' + e' + m' + 1 = b + e + m$. By the i.h. $\Gamma \vdash^{(b',e',m',|u|)} t' : \mathtt{t}$. Then from the exact subject expansion property (Lem. 19), it follows that $\Gamma \vdash^{(b,e,m,|u|)} t : \mathtt{t}$. ◀

In summary, soundness and completeness do not only establish an equivalence between tight typability and head-normalisation, but they provide a much refined equivalence property stated as follows:

▶ **Corollary 21.** *Given a term t, the following statements are equivalent*

- *There is a tight derivation $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \mathtt{t}$.*
- *There exists a canonical form $u \in \mathcal{M}$ such that $t \to_h^{(b,e,m)} u$ and $|u| = f$.*

8 Conclusion

This paper provides a quantitative insight of pattern matching by using type systems to study some of its *dynamical* properties. Indeed, our typing system $\mathscr{U}$ (resp. $\mathscr{E}$) provides upper bounds (resp. exact measures) about time and space properties related to (dynamic) computation. More precisely, the tuple of integers in the conclusion of a *tight* $\mathscr{E}$-derivation for a term t provides the exact *length* of the head-normalisation sequence of t and the *size* of its normal form. Moreover, the length of the normalisation sequence is *discriminated* according to different kind of steps performed to evaluate t.

Future work includes generalisations to more powerful notions of (dynamic) patterns, and to other reduction strategies for pattern calculi, as well to programs with recursive schemes. Inhabitation for our typing system is conjectured to be decidable, as the one in [19], but this still needs to be formally proved, in which case the result "solvability = typing+ inhabitation" in opt. cit. would be restated in a simpler framework. The quest of a general notion of model for pattern calculi also remains open, particularly for dynamic pattern calculi [32, 6].

Last, but not least, time cost analysis of a language with constructors and pattern matching is studied in [1], where it is shown that evaluation matching rules other than β-reduction may be negligible, depending on the reduction strategy and the specific notion of value. We expect the type-based quantitative technical tools we provide in this paper to be helpful in such a kind of quantitative analysis.

References

1 Beniamino Accattoli and Bruno Barras. The negligible and yet subtle cost of pattern matching. In *APLAS*, volume 10695 of *LNCS*, pages 426–447. Springer, 2017.

2 Beniamino Accattoli, Stéphane Graham-Lengrand, and Delia Kesner. Tight typings and split bounds. *PACMPL*, 2(ICFP):94:1–94:30, 2018.

3 Beniamino Accattoli and Giulio Guerrieri. Types of fireballs. In *APLAS*, volume 11275 of *LNCS*, pages 45–66. Springer, 2018.

4 Beniamino Accattoli, Giulio Guerrieri, and Maico Leberle. Types by need. In *ESOP*, volume 11423 of *LNCS*, pages 410–439. Springer, 2019.

5 Beniamino Accattoli and Delia Kesner. Preservation of strong normalisation modulo permutations for the structural lambda-calculus. *Logical Methods in Computer Science*, 8(1), 2012. doi:10.2168/LMCS-8(1:28)2012.

6 Sandra Alves, Besik Dundua, Mário Florido, and Temur Kutsia. Pattern-based calculi with finitary matching. *Logic Journal of the IGPL*, 26(2):203–243, 2018.

7 Martin Avanzini and Ugo Dal Lago. Automating sized-type inference for complexity analysis. *PACMPL*, 1(ICFP):43:1–43:29, 2017.

8 Thibaut Balabonski, Pablo Barenbaum, Eduardo Bonelli, and Delia Kesner. Foundations of strong call by need. *PACMPL*, 1(ICFP):20:1–20:29, 2017.

9 Pablo Barenbaum, Eduardo Bonelli, and Kareem Mohamed. Pattern matching and fixed points: Resource types and strong call-by-need: Extended abstract. In *PPDP*, pages 6:1–6:12. ACM Press, 2018.

10 Hendrik Pieter Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in logic and the foundation of mathematics*. North-Holland, Amsterdam, revised edition, 1984.

11 Hendrik Pieter Barendregt, Wil Dekkers, and Richard Statman. *Lambda Calculus with Types*. Perspectives in logic. Cambridge University Press, 2013.

12 Alexis Bernadet. *Types intersections non-idempotents pour raffiner la normalisation forte avec des informations quantitatives*. PhD thesis, École Polytechnique, 2014.

13 Alexis Bernadet and Stéphane Lengrand. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science*, 9(4), 2013.

14 Frédéric Blanqui. Size-based termination of higher-order rewriting. *Journal of Functional Programming*, 28:e11, 2018.

15 Eduardo Bonelli, Delia Kesner, Carlos Lombardi, and Alejandro Ríos. Normalisation for dynamic pattern calculi. In *RTA*, volume 15 of *LIPIcs*, pages 117–132. Schloss Dagstuhl, 2012.

16 Gérard Boudol, Pierre-Louis Curien, and Carolina Lavatelli. A semantics for lambda calculi with resources. *Mathematical Structures in Computer Science*, 9(4):437–482, 1999.

17 Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics: the exponentials. *Annals of Pure and Applied Logic*, 109(3):205–241, 2001.

18 Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. A relational semantics for parallelism and non-determinism in a functional setting. *Annals of Pure and Applied Logic*, 163(7):918–934, 2012.

19 Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. Observability for pair pattern calculi. In *TLCA*, volume 38 of *LIPIcs*, pages 123–137, 2015.

20 Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. Non-idempotent intersection types for the lambda-calculus. *Logic Journal of the IGPL*, 25(4):431–464, 2017.

21 Horatiu Cirstea and Claude Kirchner. The rewriting calculus — Part I. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(3):427–463, 2001.

22 Daniel de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. Phd thesis, Université Aix-Marseille II, 2007.

23 Daniel de Carvalho. Execution time of λ-terms via denotational semantics and intersection types. *Mathematical Structures in Computer Science*, 28(7):1169–1203, 2018.

24 Daniel de Carvalho and Lorenzo Tortora de Falco. A semantic account of strong normalization in linear logic. *Information and Computation*, 248:104–129, 2016.

25 Thomas Ehrhard. Collapsing non-idempotent intersection types. In *CSL*, volume 16 of *LIPIcs*, pages 259–273. Schloss Dagstuhl, 2012.

26 Thomas Ehrhard and Giulio Guerrieri. The bang calculus: an untyped lambda-calculus generalizing call-by-name and call-by-value. In *PPDP*, pages 174–187. ACM Press, 2016.

27 Philippa Gardner. Discovering needed reductions using type theory. In *TACS*, volume 789 of *LNCS*, pages 555–574. Springer, 1994.

28 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987. doi:10.1016/0304-3975(87)90045-4.

29 Giulio Guerrieri and Giulio Manzonetto. The bang calculus and the two girard's translations. In *Linearity-TLLA*, volume 292 of *EPTCS*, pages 15–30, 2018.

30 Jan Hoffmann, Klaus Aehlig, and Martin Hofmann. Resource aware ML. In *CAV*, volume 7358 of *LNCS*, pages 781–786. Springer, 2012.

31 Jan Hoffmann, Ankush Das, and Shu-Chun Weng. Towards automatic resource bound analysis for ocaml. In *POPL*, pages 359–373. ACM Press, 2017.

32 Barry Jay and Delia Kesner. First-class patterns. *Journal of Functional Programming*, 19(2):191–225, 2009.

33 Steffen Jost, Kevin Hammond, Hans-Wolfgang Loidl, and Martin Hofmann. Static determination of quantitative resource usage for higher-order programs. In *POPL*, pages 223–236. ACM Press, 2010.

34 Steffen Jost, Pedro B. Vasconcelos, Mário Florido, and Kevin Hammond. Type-based cost analysis for lazy functional languages. *Journal of Automated Reasoning*, 59(1):87–120, 2017.

35 Wolfram Kahl. Basic pattern matching calculi: a fresh view on matching failure. In *FLOPS*, volume 2998 of *LNCS*, pages 276–290. Springer, 2004.

36 Delia Kesner. Reasoning about call-by-need by means of types. In *FoSSaCS*, volume 9634 of *LNCS*, pages 424–441. Springer, 2016.

37 Delia Kesner, Carlos Lombardi, and Alejandro Ríos. Standardisation for constructor based pattern calculi. In *HOR*, volume 49, page 58–72, 2011.

38 Delia Kesner and Daniel Ventura. Quantitative types for the linear substitution calculus. In *IFIP TCS*, volume 8705 of *LNCS*, pages 296–310. Springer, 2014.

39 Delia Kesner and Daniel Ventura. A resource aware computational interpretation for Herbelin's syntax. In *ICTAC*, volume 9399 of *LNCS*, pages 388–403. Springer, 2015.

40 Delia Kesner and Pierre Vial. Types as resources for classical natural deduction. In *FSCD*, volume 84 of *LIPIcs*, pages 24:1–24:17. Schloss Dagstuhl, 2017.

41 Delia Kesner and Pierre Vial. Extracting exact bounds from typing in a classical framework. 25th International Conference on Types for Proofs and Programs, 2019.

42 Delia Kesner and Pierre Vial. Consuming and persistent types for classical logic. In *LICS*. IEEE Computer Society, 2020.

43 Assaf Kfoury. A linearization of the lambda-calculus and consequences. *Journal of Logic and Computation*, 10(3):411–436, 2000.

44 Jean Louis Krivine. *Lambda-Calculus, Types and Models*. Masson, Paris, and Ellis Horwood, Hemel Hempstead, 1993.

45 Ugo Dal Lago and Marco Gaboardi. Linear dependent types and relative completeness. *Logical Methods in Computer Science*, 8(4), 2011.

46 Ugo Dal Lago and Barbara Petit. Linear dependent types in a call-by-value scenario. *Science of Computer Programming*, 84:77–100, 2014.

47 C.-H. Luke Ong. Quantitative semantics of the lambda calculus: Some generalisations of the relational model. In *LICS*, pages 1–12. IEEE Computer Society, 2017.

48 Luca Paolini, Mauro Piccolo, and Simona Ronchi Della Rocca. Essential and relational models. *Mathematical Structures in Computer Science*, 27(5):626–650, 2017.

49 Álvaro J. Rebón Portillo, Kevin Hammond, Hans-Wolfgang Loidl, and Pedro B. Vasconcelos. Cost analysis using automatic size and time inference. In *IFL*, volume 2670 of *LNCS*, pages 232–248. Springer, 2002.

50 RaML. Resource Aware ML. URL: http://raml.co.

51 Franz Siglmüller. Type-based resource analysis on haskell. In *DICE-FOPARA*, volume 298 of *EPTCS*, pages 47–60, 2019.

52 Hugo R. Simões, Kevin Hammond, Mário Florido, and Pedro B. Vasconcelos. Using intersection types for cost-analysis of higher-order polymorphic functional programs. In *TYPES*, volume 4502 of *LNCS*, pages 221–236. Springer, 2006.

53 Hugo R. Simões, Pedro B. Vasconcelos, Mário Florido, Steffen Jost, and Kevin Hammond. Automatic amortised analysis of dynamic memory allocation for lazy functional programs. In *ICFP*, pages 165–176. ACM Press, 2012.

54 Pedro B. Vasconcelos and Kevin Hammond. Inferring cost equations for recursive, polymorphic and higher-order functional programs. In *IFL*, volume 3145 of *LNCS*, pages 86–101. Springer, 2003.

55 Philip Wadler. The essence of functional programming. In *POPL*, pages 1–14. ACM Press, 1992.

56 Joe B. Wells. The essence of principal typings. In *ICALP*, volume 2380 of *LNCS*, pages 913–925. Springer, 2002.

A The $\mathscr{U}$ Typing System

▶ **Lemma 4** (Relevance). *Let* $\Phi \triangleright \Gamma \vdash t : \sigma$. *Then,* $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(t)$.

Proof. Let $\Phi \triangleright \Gamma \vdash t : \sigma$. By straightforward induction on Φ. Note that $\Gamma = (\Gamma \setminus\!\setminus \mathtt{var}(p)); \Gamma|_p$ in both (`abs`) and (`match`) rules. ◀

▶ **Lemma 22** (Substitution for System $\mathscr{U}$). *If* $\Phi_t \triangleright \Gamma; x : \mathcal{A} \vdash t : \sigma$, *and* $\Phi_u \triangleright \Delta \vdash u : \mathcal{A}$, *then there exists* $\Phi_{t\{x\backslash u\}} \triangleright \Gamma \wedge \Delta \vdash t\{x\backslash u\} : \sigma$ *such that* $\mathtt{sz}\left(\Phi_{t\{x\backslash u\}}\right) = \mathtt{sz}\left(\Phi_t\right) + \mathtt{sz}\left(\Phi_u\right) - |\mathcal{A}|$.

Proof. We generalise the statement as follows: Let $\Phi_u \triangleright \Delta \vdash u : \mathcal{A}$.

- If $\Phi_t \triangleright \Gamma; x : \mathcal{A} \vdash t : \sigma$, then there exists $\Phi_{t\{x\backslash u\}} \triangleright \Gamma \wedge \Delta \vdash t\{x\backslash u\} : \sigma$.
- If $\Phi_t \triangleright \Gamma; x : \mathcal{A} \vdash t : \mathcal{B}$, then there exists $\Phi_{t\{x\backslash u\}} \triangleright \Gamma \wedge \Delta \vdash t\{x\backslash u\} : \mathcal{B}$.

In both cases $\mathtt{sz}\left(\Phi_{t\{x\backslash u\}}\right) = \mathtt{sz}\left(\Phi_t\right) + \mathtt{sz}\left(\Phi_u\right) - |\mathcal{A}|$.

The proof then follows by induction on Φ_t.

- If Φ_t is (`ax`), then we consider two cases:
  - $t = x$: then $\Phi_x \triangleright x : [\sigma] \vdash x : \sigma$ and $\Phi_u \triangleright \Delta \vdash u : [\sigma]$, which is a consequence of $\Delta \vdash u : \sigma$. Then $x\{x\backslash u\} = u$, and we trivially obtain $\Phi_{t\{x\backslash u\}} \triangleright \Delta \vdash u : \sigma$. We have $\mathtt{sz}\left(\Phi_{t\{x\backslash u\}}\right) = 1 + \mathtt{sz}\left(\Phi_u\right) - 1$ as expected.
  - $t = y$: then $\Phi_y \triangleright y : [\sigma]; x : [\,] \vdash y : \sigma$ and $\Phi_u \triangleright \emptyset \vdash u : [\,]$ by the (`many`) rule. Then $y\{x\backslash u\} = y$, and we trivially obtain $\Phi_{t\{x\backslash u\}} \triangleright y : [\sigma] \vdash y : \sigma$. We have $\mathtt{sz}\left(\Phi_{t\{x\backslash u\}}\right) = 1 + 0 - 0$ as expected.
- If Φ_t ends with (`many`), then it has premises of the form $(\Phi^i_t \triangleright \Gamma^i; x : \mathcal{A}^i \vdash t : \sigma^i)_{i\in I}$, where $\Gamma = \wedge_{i\in I}\Gamma^i$, $\mathcal{A} = \wedge_{i\in I}\mathcal{A}^i$ and $\mathcal{B} = [\sigma^i]_{i\in I}$. The derivation Φ_u can also be decomposed into subderivations $(\Phi^i_u \triangleright \Delta^i \vdash u : \mathcal{A}^i)_{i\in I}$ where $\Delta = \wedge_{i\in I}\Delta^i$. The i.h. gives the derivations $(\Phi^i_{t\{x\backslash u\}} \triangleright \Gamma^i \wedge \Delta^i \vdash t\{x\backslash u\} : \sigma^i)_{i\in I}$. Then we apply rule (`many`) to get $\Phi_{t\{x\backslash u\}} \triangleright \Gamma \wedge \Delta \vdash t\{x\backslash u\} : \mathcal{B}$. The statement about $\mathtt{sz}\,(\_)$ works as expected by the i.h.
- If Φ_t ends with (`abs`), so that $t = \lambda p.t'$ then, without loss of generality, one can always assume that $(\mathtt{fv}(u) \cup \{x\}) \cap \mathtt{var}(p) = \emptyset$. The result will follow easily by induction and relevance of the typing system. The statement about $\mathtt{sz}\,(\_)$ works as expected by the i.h.
- If Φ_t ends with (`app`), so that $t = t'u'$, then $\Phi_{t'u'}$ is of the form

$$\frac{\Phi_{t'} \triangleright \Gamma_{t'}; x : \mathcal{A}_{t'} \vdash t' : \mathcal{B} \to \sigma \qquad \Phi_{u'} \triangleright \Gamma_{u'}; x : \mathcal{A}_{u'} \vdash u' : \mathcal{B}}{\Gamma_{t'} \wedge \Gamma_{u'}; x : \mathcal{A}_{t'} \wedge \mathcal{A}_{u'} \vdash t'u' : \sigma}$$

Also, $\Phi_u \triangleright \Delta \vdash u : \mathcal{A}$ is a consequence of $(\Delta_k \vdash u : \sigma_k)_{k\in K}$, with $\mathcal{A} = [\sigma_k]_{k\in K}$ and $\Delta = \wedge_{k\in K}\Delta_k$. Note that $\mathcal{A} = \mathcal{A}_{t'} \wedge \mathcal{A}_{u'} = [\sigma_i]_{i\in K_{t'}} \wedge [\sigma_i]_{i\in K_{u'}}$, with $K = K_{t'} \uplus K_{u'}$, from which one can obtain both $\Delta_{t'} \vdash u : \mathcal{A}_{t'}$ and $\Delta_{u'} \vdash u : \mathcal{A}_{u'}$, through the (`many`) rule. By

the i.h. we then have $\Gamma_{t'} \wedge \Delta_{t'} \vdash t'\{x\backslash u\} : \mathcal{B} \to \sigma$ and $\Gamma_{u'} \wedge \Delta_{u'} \vdash u'\{x\backslash u\} : \mathcal{B}$. Finally, $\Gamma_{t'} \wedge \Gamma_{u'} \wedge \Delta_{t'} \wedge \Delta_{u'} \vdash (t'\{x\backslash u\})(u'\{x\backslash u\}) : \sigma$ by the $\mathtt{app}$ rule. The statement about $\mathtt{sz}\,(\_)$ works as expected by the i.h.

- If Φ_t ends with ($\mathtt{pair}$) or ($\mathtt{pair_p}$), so that $t = \langle t', u' \rangle$, then the result is obtained by induction following the same reasoning used in rule $\mathtt{app}$. The statement about $\mathtt{sz}\,(\_)$ works as expected by the i.h.
- If Φ_t ends with ($\mathtt{match}$), so that $t = t'[p\backslash u']$, then the proof is similar to the application case since $t'[p\backslash u']\{x\backslash u\} = (t'\{x\backslash u\})[p\backslash u'\{x\backslash u\}]$ and we can assume that $(\mathtt{fv}(u) \cup \{x\}) \cap \mathtt{var}(p) = \emptyset$. The statement about $\mathtt{sz}\,(\_)$ works as expected by the i.h. ◀

▶ **Lemma 23** (Anti-Substitution for System $\mathscr{U}$). *Let $\Phi \triangleright \Gamma \vdash t\{x\backslash u\} : \sigma$. Then, there exist derivations Φ_t, Φ_u, contexts Γ_t, Γ_u, and multitype $\mathcal{A}$ such that $\Phi_t \triangleright \Gamma_t; x : \mathcal{A} \vdash t : \sigma$, $\Phi_u \triangleright \Gamma_u \vdash u : \mathcal{A}$ and $\Gamma = \Gamma_t \wedge \Gamma_u$. Moreover, $\mathtt{sz}\,(\Phi) = \mathtt{sz}\,(\Phi_t) + \mathtt{sz}\,(\Phi_u) - |\mathcal{A}|$.*

Proof. As in the case of the substitution lemma, the proof follows by generalising the property for the two cases where the type derivation Φ assigns a type or a multiset type:

- Let $\Phi \triangleright \Gamma \vdash t\{x\backslash u\} : \sigma$. Then, there exist derivations Φ_t, Φ_u, contexts Γ_t, Γ_u, and multitype $\mathcal{A}$ such that $\Phi_t \triangleright \Gamma_t; x : \mathcal{A} \vdash t : \sigma$, $\Phi_u \triangleright \Gamma_u \vdash u : \mathcal{A}$ and $\Gamma = \Gamma_t \wedge \Gamma_u$.
- Let $\Phi \triangleright \Gamma \vdash t\{x\backslash u\} : \mathcal{B}$. Then, there exist derivations Φ_t, Φ_u, contexts Γ_t, Γ_u, and multitype $\mathcal{A}$ such that $\Phi_t \triangleright \Gamma_t; x : \mathcal{A} \vdash t : \mathcal{B}$, $\Phi_u \triangleright \Gamma_u \vdash u : \mathcal{A}$ and $\Gamma = \Gamma_t \wedge \Gamma_u$.

In both cases $\mathtt{sz}\,(\Phi) = \mathtt{sz}\,(\Phi_t) + \mathtt{sz}\,(\Phi_u) - |\mathcal{A}|$ holds.

We will reason by induction on Φ and cases analysis on t. For all the rules (except $\mathtt{many}$), we will have the trivial case $t\{x\backslash u\}$, where $t = x$, in which case $t\{x\backslash u\} = u$, for which we have a derivation $\Phi \triangleright \Gamma \vdash u : \sigma$. Therefore $\Phi_t \triangleright x : [\sigma] \vdash x : \sigma$ and $\Phi_u \triangleright \Gamma \vdash u : [\sigma]$ is obtained from Φ using the ($\mathtt{many}$) rule. We conclude since $\mathtt{sz}\,(\Phi) = 1 + \mathtt{sz}\,(\Phi_u) - 1$. We now reason on the different cases assuming that $t \neq x$.

- If Φ is (ax) then $\Phi \triangleright y : [\sigma] \vdash y : \sigma$ and, since $t \neq x$, $t = y \neq x$. Then we take $\mathcal{A} = [\,]$, $\Phi_t \triangleright y : [\sigma]; x : [\,] \vdash y : \sigma$, and $\Phi_u \triangleright \emptyset \vdash u : [\,]$ from rule ($\mathtt{many}$). We conclude since $\mathtt{sz}\,(\Phi) = 1 + 0 - 0$.
- If Φ ends with ($\mathtt{many}$), then $\Phi \triangleright \wedge_{k \in K} \Gamma_k \vdash t\{x\backslash u\} : [\sigma_k]_{k \in K}$ follows from the derivation $\Phi^k \triangleright \Gamma_k \vdash t\{x\backslash u\} : \sigma_k$, for each $k \in K$. By the i.h. there exist Φ^k_t, Φ^k_u, contexts Γ^k_t, Γ^k_u and multitype $\mathcal{A}_k$, such that $\Phi^k_t \triangleright \Gamma^k_t; x : \mathcal{A}_k \vdash t : \sigma_k$, $\Phi^k_u \triangleright \Gamma^k_u \vdash u : \mathcal{A}_k$, $\Gamma_k = \Gamma^k_t \wedge \Gamma^k_u$. Taking $\mathcal{A} = \wedge_{k \in K} \mathcal{A}_k$ and using rule $\mathtt{many}$ we get $\wedge_{k \in K} \Gamma^k_t; x : \mathcal{A} \vdash t : [\sigma_k]_{k \in K}$. From the premises of Φ^k_u for $k \in K$, applying the $\mathtt{many}$ rule, we get $\wedge_{k \in K} \Gamma^k_u \vdash u : \mathcal{A}$. Note that $\Gamma = \wedge_{k \in K} \Gamma_k = (\wedge_{k \in K} \Gamma^k_t) \wedge (\wedge_{k \in K} \Gamma^k_u)$. The statement about $\mathtt{sz}\,(\_)$ works as expected by the i.h.
- If Φ ends with ($\mathtt{abs}$), then $t = \lambda p.t'$, therefore $\Phi \triangleright \Gamma \setminus\!\setminus \mathtt{var}(p) \vdash \lambda p.(t'\{x\backslash u\}) : \mathcal{B} \to \sigma$ follows from $\Phi' \triangleright \Gamma \vdash t'\{x\backslash u\} : \sigma$ and $\Pi_p \triangleright \Gamma|_p \Vdash p : \mathcal{B}$. Note that, one can always assume that $\mathtt{var}(p) \cap \mathtt{fv}(u) = \emptyset$ and $x \notin \mathtt{var}(p)$. By the i.h., $\Phi_{t'} \triangleright \Gamma_{t'}; x : \mathcal{A} \vdash t' : \sigma$, $\Phi_u \triangleright \Gamma_u \vdash u : \mathcal{A}$, with $\Gamma = \Gamma_{t'} \wedge \Gamma_u$. Then using $\mathtt{abs}$ we get $\Phi_t \triangleright \Gamma_{t'} \setminus\!\setminus \mathtt{var}(p); x : \mathcal{A} \vdash \lambda p.t' : \mathcal{B} \to \sigma$. Note that $(\Gamma_{t'}; x : \mathcal{A}) \setminus\!\setminus \mathtt{var}(p) = \Gamma_{t'} \setminus\!\setminus \mathtt{var}(p); x : \mathcal{A}$ and $\Gamma \setminus\!\setminus \mathtt{var}(p) = (\Gamma_{t'} \setminus\!\setminus \mathtt{var}(p)) \wedge \Gamma_u$. The statement about $\mathtt{sz}\,(\_)$ works as expected by the i.h.
- The remaining cases for ($\mathtt{app}$), ($\mathtt{pair}$) and ($\mathtt{match}$) also hold by the i.h. and do not present any special difficulty. ◀

▶ **Lemma 5.** Let $\Phi \triangleright \Gamma \vdash t : \sigma$. Then,

1. (Upper Subject Reduction). $t \to_{\mathtt{p}} t'$ implies there is $\Phi' \triangleright \Gamma \vdash t' : \sigma$ s.t. $\mathtt{sz}\,(\Phi) \geq \mathtt{sz}\,(\Phi')$, and $t \to_{\mathtt{h}} t'$ implies there is $\Phi' \triangleright \Gamma \vdash t' : \sigma$ s.t. $\mathtt{sz}\,(\Phi) > \mathtt{sz}\,(\Phi')$.
2. (Upper Subject Expansion). $t' \to_{\mathtt{p}} t$ implies there is $\Phi' \triangleright \Gamma \vdash t' : \sigma$ such that $\mathtt{sz}\,(\Phi') \geq \mathtt{sz}\,(\Phi)$ and $t' \to_{\mathtt{h}} t$ implies there is $\Phi' \triangleright \Gamma \vdash t' : \sigma$ such that $\mathtt{sz}\,(\Phi') > \mathtt{sz}\,(\Phi)$.

Proof. Let $\Phi \triangleright \Gamma \vdash t : \sigma$.

1. By induction on $\to_{\mathtt{p}}$ (resp. $\to_{\mathtt{h}}$) and the substitution property (Lem. 22). The first three cases represent the base cases for both reductions, where the size relation is strict.
   - $t = \mathtt{L}[\![\lambda p.v]\!]u \to_{\mathtt{p/h}} \mathtt{L}[\![v[p\backslash u]]\!] = t'$. The proof is by induction on the list $\mathtt{L}$. We only show the case of the empty list as the other one is straightforward. The typing derivation Φ is necessarily of the form
   $$\dfrac{\dfrac{\Gamma_v \vdash v : \sigma \quad \Gamma_v|_p \Vdash p : \mathcal{A}}{\Gamma_v \between \mathtt{var}(p) \vdash \lambda p.v : \mathcal{A} \to \sigma} \quad \Gamma_u \vdash u : \mathcal{A}}{\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u \vdash (\lambda p.v)u : \sigma}$$
   We then construct the following derivation Φ':
   $$\dfrac{\Gamma_v \vdash v : \sigma \quad \Gamma_v|_p \Vdash p : \mathcal{A} \quad \Gamma_u \vdash u : \mathcal{A}}{\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u \vdash v[p\backslash u] : \sigma}$$
   Moreover, $\mathtt{sz}(\Phi) = \mathtt{sz}(\Phi') + 1$.
   - $t = v[x\backslash u] \to_{\mathtt{p}} v\{x\backslash u\} = t'$. Then Φ has two term premises $\Phi_v \triangleright \Gamma_v; x : \mathcal{A} \vdash v : \sigma$, $\Phi_u \triangleright \Gamma_u \vdash u : \mathcal{A}$, and one pattern premise $\Pi_x \triangleright x : \mathcal{A} \Vdash x : \mathcal{A}$, where $\Gamma = \Gamma_v \wedge \Gamma_u$ and $\mathtt{sz}(\Phi) = \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Phi_u) + \mathtt{sz}(\Pi_x) + 1$. Lem. 22 then gives a derivation Φ' ending with $\Gamma_v \wedge \Gamma_u \vdash v\{x\backslash u\} : \sigma$, where $|\mathcal{A}| \geq 0$ and $\mathtt{sz}(\Pi_x) = 1$ imply
   $$\mathtt{sz}(\Phi') = \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Phi_u) - |\mathcal{A}| < \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Phi_u) + \mathtt{sz}(\Pi_x) < \mathtt{sz}(\Phi)$$
   When $t = v[x\backslash u] \to_{\mathtt{h}} v\{x\backslash u\} = t'$, where $v \not\to_{\mathtt{h}}$, the same results hold.
   - $t = v[\langle p_1, p_2\rangle \backslash \mathtt{L}[\![\langle u_1, u_2\rangle]\!]] \to_{\mathtt{p}} \mathtt{L}[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!] = t'$. Let us write $p = \langle p_1, p_2\rangle$ and $u = \langle u_1, u_2\rangle$. The typing derivation Φ is necessarily of the form
   $$\dfrac{\Phi_v \triangleright \Gamma_v \vdash v : \sigma \quad \Pi_p \triangleright \Gamma_v|_p \Vdash \langle p_1, p_2\rangle : \mathcal{A} \quad \Phi_u \triangleright \Gamma_u \vdash \mathtt{L}[\![\langle u_1, u_2\rangle]\!] : \mathcal{A}}{\Gamma_v \between \mathtt{var}(\langle p_1, p_2\rangle) \wedge \Gamma_u \vdash v[\langle p_1, p_2\rangle \backslash \mathtt{L}[\![\langle u_1, u_2\rangle]\!]] : \sigma}$$
   Moreover, $\mathcal{A} = [\times(\mathcal{A}_1, \mathcal{A}_2)]$ and $\mathtt{sz}(\Phi) = \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Pi_p) + \mathtt{sz}(\Phi_u) + 1$.
   Then Π_p is of the form:
   $$\dfrac{\Pi_{p_1} \triangleright \Gamma_v|_{p_1} \Vdash p_1 : \mathcal{A}_1 \quad \Pi_{p_2} \triangleright \Gamma_v|_{p_2} \Vdash p_2 : \mathcal{A}_2 \quad p_1 \# p_2}{\Gamma_v|_p \Vdash \langle p_1, p_2\rangle : [\times(\mathcal{A}_1, \mathcal{A}_2)]}$$
   and $\mathtt{sz}(\Pi_p) = \mathtt{sz}(\Pi_{p_1}) + \mathtt{sz}(\Pi_{p_2}) + 1$
   The proof is then by induction on the list $\mathtt{L}$.
   - For $\mathtt{L} = \Box$ we have Φ_u of the form:
   $$\dfrac{\dfrac{\Phi_{u_1} \triangleright \Gamma_{u_1} \vdash u_1 : \mathcal{A}_1 \quad \Phi_{u_2} \triangleright \Gamma_{u_2} \vdash u_2 : \mathcal{A}_2}{\Gamma_u \vdash \langle u_1, u_2\rangle : \times(\mathcal{A}_1, \mathcal{A}_2)}}{\Gamma_u \vdash \langle u_1, u_2\rangle : \mathcal{A}}$$
   where $\Gamma_u = \Gamma_{u_1} \wedge \Gamma_{u_2}$ and $\mathtt{sz}(\Phi_u) = \mathtt{sz}(\Phi_{u_1}) + \mathtt{sz}(\Phi_{u_2}) + 1$. We first construct the following derivation:
   $$\dfrac{\Phi_v \triangleright \Gamma_v \vdash v : \sigma \quad \Pi_{p_1} \triangleright \Gamma_v|_{p_1} \Vdash p_1 : \mathcal{A}_1 \quad \Phi_{u_1} \triangleright \Gamma_{u_1} \vdash u_1 : \mathcal{A}_1}{\Gamma_v \between \mathtt{var}(p_1) \wedge \Gamma_{u_1} \vdash v[p_1\backslash u_1] : \sigma}$$
   By using Lem. 4 and α-conversion, we construct a derivation Φ' with conclusion $\Gamma_v \between \mathtt{var}(p_1) \between \mathtt{var}(p_2) \wedge \Gamma_{u_1} \wedge \Gamma_{u_2} \vdash v[p_1\backslash u_1][p_2\backslash u_2] : \sigma$. Note that $p_1 \# p_2$ implies $\Gamma_v \between \mathtt{var}(\langle p_1, p_2\rangle) = \Gamma_v \between \mathtt{var}(p_1) \between \mathtt{var}(p_2)$. Thus, we finally obtain $\mathtt{sz}(\Phi') = \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Pi_p) + \mathtt{sz}(\Phi_u) < \mathtt{sz}(\Phi)$.

- Let $\mathtt{L} = \mathtt{L}'[q\backslash s]$. Then Φ_u is necessarily of the following form:

$$\dfrac{\dfrac{\Phi_{\mathtt{L}'} \triangleright \Delta_u \vdash \mathtt{L}'[\![\langle u_1, u_2\rangle]\!] : \times(\mathcal{A}_1, \mathcal{A}_2) \qquad \Pi_q \triangleright \Delta_u|_q \Vdash q : \mathcal{B} \qquad \Phi_s \triangleright \Delta_s \vdash s : \mathcal{B}}{\Gamma_u \vdash \mathtt{L}'[\![\langle u_1, u_2\rangle]\!][q\backslash s] : \times(\mathcal{A}_1, \mathcal{A}_2)}}{\Gamma_u \vdash \mathtt{L}'[\![\langle u_1, u_2\rangle]\!][q\backslash s] : \mathcal{A}}$$

  where $\Gamma_u = \Delta_u \between \mathtt{var}(q) \wedge \Delta_s$.
  We will apply the i.h. on the reduction step $v[p\backslash \mathtt{L}'[\![u]\!]] \to_{\mathtt{p}} \mathtt{L}'[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!]$, in particular we type the left-hand side term with the following derivation Ψ_1:

$$\dfrac{\Phi_v \qquad \Pi_p \qquad \dfrac{\Phi_{\mathtt{L}'} \triangleright \Delta_u \vdash \mathtt{L}'[\![\langle u_1, u_2\rangle]\!] : \times(\mathcal{A}_1, \mathcal{A}_2)}{\Delta_u \vdash \mathtt{L}'[\![\langle u_1, u_2\rangle]\!] : \mathcal{A}}}{\Gamma_v \between \mathtt{var}(p) \wedge \Delta_u \vdash v[\langle p_1, p_2\rangle\backslash \mathtt{L}'[\![\langle u_1, u_2\rangle]\!]] : \sigma}$$

  The i.h. gives a derivation $\Psi_2 \triangleright \Gamma_v \between \mathtt{var}(p) \wedge \Delta_u \vdash \mathtt{L}'[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!] : \sigma$ verifying $\mathtt{sz}(\Psi_2) < \mathtt{sz}(\Psi_1)$. Let $\Lambda = \Gamma_v \between \mathtt{var}(p) \wedge \Delta_u$. We conclude with the following derivation Φ':

$$\dfrac{\Psi_2 \qquad \Pi_q \triangleright \Delta_u|_q \Vdash q : \mathcal{B} \qquad \Phi_s \triangleright \Delta_s \vdash s : \mathcal{B}}{\Lambda \between \mathtt{var}(q) \wedge \Delta_s \vdash \mathtt{L}'[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!][q\backslash s] : \sigma}$$

  Indeed, we first remark that $\Lambda|_q = \Delta_u|_q$ holds by relevance and α-conversion. Secondly, $\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u = \Gamma_v \between \mathtt{var}(p) \wedge (\Delta_u \between \mathtt{var}(q)) \wedge \Delta_s = \Lambda \between \mathtt{var}(q) \wedge \Delta_s$ also holds by Lem. 4 and α-conversion. Last, we have

$$\begin{array}{rll} \mathtt{sz}(\Phi') = & \mathtt{sz}(\Psi_2) + \mathtt{sz}(\Pi_q) + \mathtt{sz}(\Phi_s) + 1 & < \\ & \mathtt{sz}(\Psi_1) + \mathtt{sz}(\Pi_q) + \mathtt{sz}(\Phi_s) + 1 & = \\ & \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Pi_p) + \mathtt{sz}(\Phi_{\mathtt{L}'}) + 1 + \mathtt{sz}(\Pi_q) + \mathtt{sz}(\Phi_s) + 1 & = \\ & \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Pi_p) + \mathtt{sz}(\Phi_u) + 1 & = \mathtt{sz}(\Phi) \end{array}$$

  When $t = v[\langle p_1, p_2\rangle\backslash \mathtt{L}[\![\langle u_1, u_2\rangle]\!]] \to_{\mathtt{h}} \mathtt{L}[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!] = t'$, where $v \not\to_{\mathtt{h}}$, the same results hold.
- Most of the inductive cases are straightforward. We only detail here two interesting cases.
  - $t = v[p\backslash u] \to_{\mathtt{p}} v[p\backslash u'] = t'$, where $u \to_{\mathtt{p}} u'$. The proof holds here by the i.h. In particular, when $p = x$ and $x \notin \mathtt{fv}(v)$, then by relevance we have x of type $[\,]$ as well as u of type $[\,]$. This means that both u and u' are typed by a (`many`) rule with no premise, and in that case we get $\mathtt{sz}(\Phi) = \mathtt{sz}(\Phi')$.
  - $t = v[p\backslash u] \to_{\mathtt{h}} v[p\backslash u'] = t'$, where $v \not\to_{\mathtt{h}}$ and $p \neq x$ and $u \to_{\mathtt{h}} u'$. By construction there are typing subderivations $\Phi_v \triangleright \Gamma_v \vdash v : \sigma$, $\Pi_p \triangleright \Gamma_v|_p \Vdash p : \mathcal{A}$ and $\Phi_u \triangleright \Gamma_u \vdash u : \mathcal{A}$ such that $\Gamma = \Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u$. Since p is not a variable then Π_p ends with rule $\mathtt{pat}_{\times}$. In which case $\mathcal{A}$ contains exactly one type, let us say $\mathcal{A} = [\sigma_u]$. Then Φ_u has the following form

$$\dfrac{\Gamma_u \vdash u : \sigma_u}{\Phi_u \triangleright \Gamma_u \vdash u : [\sigma_u]}$$

    The i.h. applied to the premise of Φ_u gives a derivation $\Gamma_u \vdash u' : \sigma_u$ and having the expected size relation. To conclude we build a type derivation Φ' for $v[p\backslash u']$ having the expected size relation.

2. By induction on $\to_{\mathtt{p}}$ (resp. $\to_{\mathtt{h}}$) and the anti-substitution property (Lem. 23).
   - $t' = \mathtt{L}[\![\lambda p.v]\!]u \to_{\mathtt{p/h}} \mathtt{L}[\![v[p\backslash u]]\!] = t$. The proof is by induction on the list $\mathtt{L}$. We consider the case $\mathtt{L} = \Box$, since the other case follows straightforward by i.h. The typing derivation Φ is necessarily of the form:

$$\dfrac{\Gamma_v \vdash v : \sigma \quad \Gamma_v|_p \Vdash p : \mathcal{A} \quad \Gamma_u \vdash u : \mathcal{A}}{\Gamma_v \setminus\!\setminus \mathtt{var}(p) \wedge \Gamma_u \vdash v[p\backslash u] : \sigma}$$

     We then construct the following derivation Φ':

$$\dfrac{\dfrac{\Gamma_v \vdash v : \sigma \quad \Gamma_v|_p \Vdash p : \mathcal{A}}{\Gamma_v \setminus\!\setminus \mathtt{var}(p) \vdash \lambda p.v : \mathcal{A} \to \sigma} \quad \Gamma_u \vdash u : \mathcal{A}}{\Gamma_v \setminus\!\setminus \mathtt{var}(p) \wedge \Gamma_u \vdash (\lambda p.v)u : \sigma}$$

     Moreover, $\mathtt{sz}(\Phi') = \mathtt{sz}(\Phi) + 1$.
   - $t' = v[x\backslash u] \to_{\mathtt{p}} v\{x\backslash u\} = t$. Then by Lem. 23, there exist derivations Φ_v, Φ_u, contexts Γ_v, Γ_u and a multitype $\mathcal{A}$, such that $\Phi_v \triangleright \Gamma_v; x : \mathcal{A} \vdash v : \sigma$, $\Phi_u \triangleright \Gamma_u \vdash u : \mathcal{A}$, $\Gamma = \Gamma_v \wedge \Gamma_u$, and $\mathtt{sz}(\Phi) = \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Phi_u) - |\mathcal{A}|$. Furthermore, one has $\Pi_x \triangleright x : \mathcal{A} \Vdash x : \mathcal{A}$ Then one can construct the following derivation Φ'.

$$\dfrac{\Gamma_v; x : \mathcal{A} \vdash v : \sigma \quad x : \mathcal{A} \Vdash x : \mathcal{A} \quad \Gamma_u \vdash u : \mathcal{A}}{\Gamma_v \wedge \Gamma_u \vdash v[x\backslash u] : \sigma}$$

     Furthermore, $\mathtt{sz}(\Phi') = \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Pi_x) + \mathtt{sz}(\Phi_u) > \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Phi_u) - |A|$, since $|\mathcal{A}| \geq 0$ and $\mathtt{sz}(\Pi_x) = 1$. The same result holds for $t = v[x\backslash u] \to_{\mathtt{h}} v\{x\backslash u\} = t'$, where $v \not\to_{\mathtt{h}}$.
   - $t' = v[\langle p_1, p_2\rangle \backslash \mathtt{L}[\![\langle u_1, u_2\rangle]\!]] \to_{\mathtt{p}} \mathtt{L}[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!] = t$. Let us write $p = \langle p_1, p_2\rangle$ and $u = \langle u_1, u_2\rangle$. The proof is by induction on the list $\mathtt{L}$.
     - $\mathtt{L} = \Box$, then the typing derivation Φ is necessarily of the form:

$$\dfrac{\dfrac{\Gamma_v \vdash v : \sigma \quad \Gamma_v|_{p_1} \Vdash p_1 : \mathcal{A}_1 \quad \Gamma_1 \vdash u_1 : \mathcal{A}_1}{(\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \wedge \Gamma_1 \vdash v[p_1\backslash u_1] : \sigma} \quad ((\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \wedge \Gamma_1)|_{p_2} \Vdash p_2 : \mathcal{A}_2 \quad \Gamma_2 \vdash u_2 : \mathcal{A}_2}{(((\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \wedge \Gamma_1) \setminus\!\setminus \mathtt{var}(p_2)) \wedge \Gamma_2 \vdash v[p_1\backslash u_1][p_2\backslash u_2] : \sigma}$$

       where $\Gamma = (((\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \wedge \Gamma_1) \setminus\!\setminus \mathtt{var}(p_2)) \wedge \Gamma_2$. Moreover, the following equality holds $(((\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \wedge \Gamma_1) \setminus\!\setminus \mathtt{var}(p_2)) = ((\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \setminus\!\setminus \mathtt{var}(p_2) \wedge \Gamma_1 \setminus\!\setminus \mathtt{var}(p_2))$, since $(\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \setminus\!\setminus \mathtt{var}(p_2) = \Gamma_v \setminus\!\setminus \mathtt{var}(p)$ and $\Gamma_1 \setminus\!\setminus \mathtt{var}(p_2) =_{L.4} \Gamma_1$. Similarly, $((\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \wedge \Gamma_1)|_{p_2} =_{L.4} (\Gamma_v \setminus\!\setminus \mathtt{var}(p_1))|_{p_2}$ and, by linearity of patterns, we have $(\Gamma_v \setminus\!\setminus \mathtt{var}(p_1))|_{p_2} = \Gamma_v|_{p_2}$. Hence, we conclude with the following derivation Φ':

$$\dfrac{\Gamma_v \vdash v : \sigma \qquad \dfrac{\Gamma_v|_{p_1} \Vdash p_1 : \mathcal{A}_1 \quad \Gamma_v|_{p_2} \Vdash p_2 : \mathcal{A}_2}{\Gamma_v|_p \Vdash p : [\times(\mathcal{A}_1, \mathcal{A}_2)]} \qquad \dfrac{\Gamma_1 \vdash u_1 : \mathcal{A}_1 \quad \Gamma_2 \vdash u_2 : \mathcal{A}_2}{\Gamma_1 \wedge \Gamma_2 \vdash u : \times(\mathcal{A}_1, \mathcal{A}_2)}}{\Gamma_1 \wedge \Gamma_2 \vdash u : [\times(\mathcal{A}_1, \mathcal{A}_2)]}}{(\Gamma_v \setminus\!\setminus \mathtt{var}(p)) \wedge (\Gamma_1 \wedge \Gamma_2) \vdash v[p\backslash u] : \sigma}$$

       Furthermore,

$$\begin{array}{rll} \mathtt{sz}(\Phi) = & \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Pi_{p_1}) + \mathtt{sz}(\Phi_{u_1}) + 1 + \mathtt{sz}(\Pi_{p_2}) + \mathtt{sz}(\Phi_{u_2}) + 1 & = \\ & \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Pi_{p_1}) + \mathtt{sz}(\Pi_{p_2}) + 1 + \mathtt{sz}(\Phi_{u_1}) + \mathtt{sz}(\Phi_{u_2}) + 1 & < \\ & \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Pi_p) + \mathtt{sz}(\Phi_u) + 1 & = \mathtt{sz}(\Phi') \end{array}$$

- If $\mathtt{L} = \mathtt{L}'[q\backslash s]$, then $t' = v[\langle p_1, p_2\rangle\backslash \mathtt{L}'[q\backslash s][\![\langle u_1, u_2\rangle]\!]] \to_{\mathtt{p}} \mathtt{L}'[q\backslash s][\![v[p_1\backslash u_1][p_2\backslash u_2]]\!] = \mathtt{L}'[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!][q\backslash s] = t$, and Φ is of the form:

$$\frac{\Phi_{\mathtt{L}'} \triangleright \Gamma_{\mathtt{L}'} \vdash \mathtt{L}'[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!] : \sigma \qquad \Pi_q \triangleright \Gamma_{\mathtt{L}'}|_q \Vdash q : \mathcal{A} \qquad \Phi_s \triangleright \Gamma_s \vdash s : \mathcal{A}}{\Gamma_{\mathtt{L}'} \between \mathtt{var}(q) \wedge \Gamma_s \vdash \mathtt{L}'[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!][q\backslash s] : \sigma}$$

From $v[\langle p_1, p_2\rangle\backslash \mathtt{L}'[\![\langle u_1, u_2\rangle]\!]] \to_{\mathtt{p}} \mathtt{L}'[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!]$ and $\Phi_{\mathtt{L}'}$ by the i.h. one gets $\Phi'_{\mathtt{L}'} \triangleright \Gamma_{\mathtt{L}'} \vdash v[\langle p_1, p_2\rangle\backslash \mathtt{L}'[\![\langle u_1, u_2\rangle]\!]] : \sigma$ with $\mathtt{sz}(\Phi'_{\mathtt{L}'}) > \mathtt{sz}(\Phi_{\mathtt{L}'})$. Furthermore $\Phi'_{\mathtt{L}'}$ is necessarily of the form:

$$\frac{\Phi_v \triangleright \Gamma_v \vdash v : \sigma \qquad \Pi_p \triangleright \Gamma_v|_p \Vdash p : [\times(\mathcal{A}_1, \mathcal{A}_2)] \qquad \dfrac{\Phi_u \triangleright \Gamma_u \vdash \mathtt{L}'[\![u]\!] : \times(\mathcal{A}_1, \mathcal{A}_2)}{\Gamma_u \vdash \mathtt{L}'[\![u]\!] : [\times(\mathcal{A}_1, \mathcal{A}_2)]}}{\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u \vdash v[p\backslash \mathtt{L}'[\![u]\!]] : \sigma}$$

Then one can construct the following derivation Φ'_u:

$$\frac{\dfrac{\Gamma_u \vdash \mathtt{L}'[\![u]\!] : \times(\mathcal{A}_1, \mathcal{A}_2) \qquad \Gamma_u|_q \Vdash q : \mathcal{A} \qquad \Phi_s}{\Gamma_u \between \mathtt{var}(q) \wedge \Gamma_s \vdash \mathtt{L}'[q\backslash s][\![u]\!] : \times(\mathcal{A}_1, \mathcal{A}_2)}}{\Gamma_u \between \mathtt{var}(q) \wedge \Gamma_s \vdash \mathtt{L}'[q\backslash s][\![u]\!] : [\times(\mathcal{A}_1, \mathcal{A}_2)]}$$

From which we build Φ':

$$\frac{\Gamma_v \vdash v : \sigma \qquad \Gamma_v|_p \Vdash p : [\times(\mathcal{A}_1, \mathcal{A}_2)] \qquad \Gamma_u \between \mathtt{var}(q) \wedge \Gamma_s \vdash \mathtt{L}'[q\backslash s][\![u]\!] : [\times(\mathcal{A}_1, \mathcal{A}_2)]}{\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u \between \mathtt{var}(q) \wedge \Gamma_s \vdash v[p\backslash \mathtt{L}'[q\backslash s][\![u]\!]] : \sigma}$$

With $\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u \between \mathtt{var}(q) \wedge \Gamma_s = (\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u) \between \mathtt{var}(q) \wedge \Gamma_s = \Gamma$. Furthermore

$$\begin{array}{lll} \mathtt{sz}(\Phi) = & \mathtt{sz}(\Phi_{\mathtt{L}'}) + \mathtt{sz}(\Pi_q) + \mathtt{sz}(\Phi_s) + 1 & <_{i.h.} \\ & \mathtt{sz}(\Phi'_{\mathtt{L}'}) + \mathtt{sz}(\Pi_q) + \mathtt{sz}(\Phi_s) + 1 & = \\ & \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Pi_p) + \mathtt{sz}(\Phi_u) + \mathtt{sz}(\Pi_q) + \mathtt{sz}(\Phi_s) + 1 + 1 & = \\ & \mathtt{sz}(\Phi_v) + \mathtt{sz}(\Pi_p) + \mathtt{sz}(\Phi'_u) + 1 & = \mathtt{sz}(\Phi') \end{array}$$

The same result holds for $t' = v[\langle p_1, p_2\rangle\backslash \mathtt{L}[\![\langle u_1, u_2\rangle]\!]] \to_{\mathtt{p}} \mathtt{L}[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!] = t$, where $v \not\to_{\mathtt{h}}$.

- Most of the inductive cases are straightforward. We only detail here two interesting cases.
  - $t' = v[p\backslash u'] \to_{\mathtt{p}} v[p\backslash u] = t$, where $u' \to_{\mathtt{p}} u$. The proof holds by the i.h. In particular, when $p = x$ and $x \notin \mathtt{fv}(v)$, then by relevance we have x of type $[\,]$ as well as u of type $[\,]$. This means that u, u' are typed by a ($\mathtt{many}$) rule with no premise, and in that case we get $\mathtt{sz}(\Phi) = \mathtt{sz}(\Phi')$.
  - $t' = v[p\backslash u'] \to_{\mathtt{h}} v[p\backslash u] = t$, where $v \not\to_{\mathtt{h}}$ and $p \neq x$ and $u' \to_{\mathtt{h}} u$. By construction there are subderivations $\Phi_v \triangleright \Gamma_v \vdash v : \sigma$, $\Pi_p \triangleright \Gamma_v|_p \Vdash p : \mathcal{A}$ and $\Phi_u \triangleright \Gamma_u \vdash u : \mathcal{A}$ for some multiset $\mathcal{A}$ and $\Gamma = (\Gamma_v \between \mathtt{var}(p)) \wedge \Gamma_u$. Since p is not a variable then Π_p ends with rule ($\mathtt{pat}_\times$), in which case $\mathcal{A}$ contains only one type, let us say $\mathcal{A} = [\sigma_u]$. Then Φ_u has the following form:

$$\Phi_u \triangleright \frac{\Gamma_u \vdash u : \sigma_u}{\Gamma_u \vdash u' : [\sigma_u]}$$

The i.h. applied to the premise of Φ_u gives a derivation $\Gamma_u \vdash u' : \sigma_u$. Therefore, we construct the following derivation Φ':

$$\frac{\Gamma_v \vdash v : \sigma \qquad \Gamma_v|_p \Vdash p : [\sigma_u] \qquad \Gamma_u \vdash u' : \sigma_u}{\Gamma_v \setminus\!\setminus \mathtt{var}(p) \wedge \Gamma_u \vdash v[p\backslash u'] : \sigma}$$

Furthermore,

$$\begin{array}{rll} \mathtt{sz}\,(\Phi) = & \mathtt{sz}\,(\Phi_v) + \mathtt{sz}\,(\Pi_p) + \mathtt{sz}\,(\Phi_u) + 1 & <_{i.h.} \\ & \mathtt{sz}\,(\Phi_v) + \mathtt{sz}\,(\Pi_p) + \mathtt{sz}\,(\Phi_{u'}) + 1 & = \mathtt{sz}\,(\Phi') \end{array}$$

◀

▶ **Lemma 6** (Clash-Free). *Let $\Phi \triangleright \Gamma \vdash t : \sigma$. Then t is (head) clash-free.*

Proof. Let $\Phi \triangleright \Gamma \vdash t : \sigma$. By induction on $\mathtt{sz}\,(\Phi)$, using the syntax-directed aspect of system $\mathscr{U}$.

- The base case for rule (ax) is trivial.
- The cases for rules (many) and (abs) are straightforward from the i.h.
- The case for (pair) is also straightforward since every pair is (head) clash-free.
- Let us consider the case for (match), where Φ has the following form:

$$\frac{\Gamma_t \vdash t : \sigma \qquad \Gamma_t|_p \Vdash p : \mathcal{A} \qquad \Delta \vdash u : \mathcal{A}}{(\Gamma_t \setminus\!\setminus \mathtt{var}(p)) \wedge \Delta \vdash t[p\backslash u] : \sigma}\,(\mathtt{match})$$

  - If $t \to_{\mathtt{h}} t'$ for some t', so that $t[p\backslash u] \to_{\mathtt{h}} t'[p\backslash u]$, then the size of the typing derivation of $t'[p\backslash u]$ is smaller than that of Φ by Upper Subject Reduction. The i.h. then gives $t'[p\backslash u]$ (head) clash-free and thus $t[p\backslash u]$ is (head) clash-free.
  - If $t \not\to_{\mathtt{h}}$ then there are two cases.
    - If p is a variable x, then $t[x\backslash u] \to_{\mathtt{h}} t\{x\backslash u\}$ and by Upper Subject Reduction $t\{x\backslash u\}$ has a type derivation strictly smaller than that of Φ, thus by the i.h. $t\{x\backslash u\}$ is (head) clash-free and so is $t[x\backslash u]$.
    - Otherwise p is a pair, so that $\mathcal{A} \neq [\,]$ (i.e. $\mathcal{A} = [\times(\mathcal{A}_1, \mathcal{A}_2)]$).
      - If $u \to_{\mathtt{h}} u'$ then $t[p\backslash u] \to_{\mathtt{h}} t[p\backslash u']$, so that the size of the typing derivation of $t[p\backslash u']$ is smaller than that of Φ by Upper Subject Reduction. The i.h. then gives $t[p\backslash u']$ (head) clash-free and thus $t[p\backslash u]$ is (head) clash-free.
      - If $u \not\to_{\mathtt{h}}$ then $t[p\backslash u]$ is a head-normal form. In order to guarantee that $t[p\backslash u]$ is (head) clash-free note that u cannot be of the form $\mathtt{L}[\![\lambda q.v]\!]$, which can only be typed with a multiset of functional types.
- Let us consider the case for rule (app), where Φ has the following form

$$\frac{\Phi_u \triangleright \Gamma_u \vdash u : \mathcal{A} \to \sigma \qquad \Phi_v \triangleright \Gamma_v \vdash v : \mathcal{A}}{\Gamma_u \wedge \Gamma_v \vdash u\, v : \sigma}$$

  Note that u cannot be of the form $\mathtt{L}[\![\langle u_1, u_2\rangle]\!]$ because it is typed with a functional type, thus it is either $\mathtt{L}[\![x]\!]$ or $\mathtt{L}[\![\lambda p.u']\!]$.
  If u is $\mathtt{L}[\![x]\!]$, then u is (head) clash-free by the i.h. and thus uv is necessarily (head) clash-free.
  If u is $\mathtt{L}[\![\lambda p.u']\!]$ then $t = \mathtt{L}[\![\lambda p.u']\!]v \to_{\mathtt{h}} \mathtt{L}[\![u'[p\backslash v]]\!] = t'$ and the size of the type derivation of t' is strictly smaller than the size of Φ by Upper Subject Reduction. The i.h. gives t' (head) clash-free, and thus t is also (head) clash-free. ◀

B Soundness of System $\mathscr{E}$

▶ **Lemma 12** (Tight Spreading). *Let $t \in \mathcal{N}$. Let $\Phi \rhd \Gamma \vdash^{(b,e,m,f)} t : \sigma$ be a typing derivation such that $\mathit{tight}(\Gamma)$. Then σ is tight and the last rule of Φ does not belong to $\{\mathit{app}, \mathit{abs}, \mathtt{abs_p}, \mathtt{pair}, \mathtt{pair_p}\}$.*

Proof. First note that, since $t \in \mathcal{N}$, then t is not an abstraction nor a pair, therefore one cannot apply any of the rules $\{\mathtt{abs}, \mathtt{abs_p}, \mathtt{pair}, \mathtt{pair_p}, \mathtt{emptypair}\}$. We now examine the remaining rules.

- $t = x$. Then Φ is an axiom $x : [\mathtt{t}] \vdash^{(0,0,0,0)} x : \mathtt{t}$ so the property trivially holds.
- $t = uv$, with $u \in \mathcal{N}$. Then Φ has a (left) subderivation $\Phi_u \rhd \Gamma_u \vdash^{(b,e,m,f)} u : \sigma_u$, and since $\Gamma_u \subseteq \Gamma$, then Γ_u is necessarily tight. Therefore, by the i.h., $\sigma_u = \bullet_{\mathcal{N}}$, from which follows that $\sigma = \bullet_{\mathcal{N}}$ by applying rule $(\mathtt{app_p})$. Note that one cannot apply rule $(\mathtt{app})$ to type uv, since t would have to be an arrow type, which contradicts the i.h.
- $t = u[p \backslash v]$, with $u \in \mathcal{N}, v \in \mathcal{N}$. Then Φ follows from $\Phi_u \rhd \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \sigma$, $\Phi_p \rhd \Gamma_u|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{A}$ and $\Phi_v \rhd \Delta \vdash^{(b_v,e_v,m_v,f_v)} v : \mathcal{A}$, where $\Gamma = (\Gamma_u \setminus\!\setminus \mathtt{var}(p)) \wedge \Delta$. Since $(\Gamma_u \setminus\!\setminus \mathtt{var}(p)) \wedge \Delta$ is tight, then Δ is tight. By the i.h. on v one gets a derivation $\Delta \vdash^{(b_v,e_v,m_v,f_v)} v : \bullet_{\mathcal{N}}$ so that $\Delta \vdash^{(b_v,e_v,m_v,f_v)} v : [\bullet_{\mathcal{N}}]$ follows from $\mathtt{many}$ and $\Gamma_u|_p \Vdash^{(0,0,1)} p : [\bullet_{\mathcal{N}}]$ necessarily follows from rule $(\mathtt{pat_p})$. This implies $\Gamma_u|_p$ is tight, therefore Γ_u is tight. Since $u \in \mathcal{N}$ the i.h. gives $\sigma \in \mathtt{t}$ as expected. ◀

▶ **Lemma 13** (Canonical Forms and Minimal Counters). *Let $\Phi \rhd \Gamma \vdash^{(b,e,m,f)} t : \sigma$ be a tight derivation. Then $t \in \mathcal{M}$ if and only if $b = e = m = 0$.*

Proof. By induction on $\Phi \rhd \Gamma \vdash^{(b,e,m,f)} t : \sigma$, where Φ is tight (right-to-left implication), and by induction on $t \in \mathcal{M}$ (left-to-right implication). The latter is presented below.

- $t = \lambda p.u$, with $u \in \mathcal{M}$. Then Φ cannot end with rule $(\mathtt{abs})$ because σ is tight. The last rule of Φ is necessarily $(\mathtt{abs_p})$. The i.h. then applies and gives $b = e = m = 0$ and $f - 1 = |u|$. We conclude since $f = |u| + 1 = |t|$.
- $t = \langle t_1, t_2 \rangle$. Then Φ necessarily ends with rule $(\mathtt{pair_p})$ and the counters are as required.
- $t = u[p \backslash v]$, with $u \in \mathcal{M}, v \in \mathcal{N}$. Then Φ ends with rule $(\mathtt{match})$, so that u (resp. v) is typable with some context Γ_u (resp. Γ_v), where $\Gamma = (\Gamma_u \setminus\!\setminus \mathtt{var}(p)) \wedge \Gamma_v$. Let us consider the type $\mathcal{A}$ of u in the premise of rule $(\mathtt{match})$. Since Γ_v is tight and $v \in \mathcal{N}$, then Lem. 12 guarantees that every type of v in $\mathcal{A}$ is tight, and every counter typing v is of the form $(0, 0, 0, |v|)$. This same multitype $\mathcal{A}$ types the pattern p, so that there are in principle two cases:
  - Either p is a variable typable with rule $(\mathtt{pat_v})$, but then $t \notin \mathcal{M}$ since t is still reducible. Contradiction.
  - Or p is typable with rule $(\mathtt{pat_p})$, so that its counter is $(0, 0, 1)$, its type is $[\bullet_{\mathcal{N}}]$ and its context is $\Gamma_u|_p$ necessarily tight by definition of rule $(\mathtt{pat_p})$.

  Since $\Gamma_u \setminus\!\setminus \mathtt{var}(p)$ is tight by hypothesis, then the whole context Γ_u is tight. We can then apply the i.h. to u and obtain counters for u of the form $b_u = e_u = m_u = 0$ and $f_u = |u|$. On the other side, since the type of p is $[\bullet_{\mathcal{N}}]$ (rule $\mathtt{pat_p}$), there is only one premise to type v, which is necessarily of the form $\Delta \vdash^{(0,0,0,|v|)} v : \bullet_{\mathcal{N}}$. We then conclude that the counters typing $u[p \backslash v]$ are $b = e = m = 0$ and $f = f_u + f_v + 1 = |u| + |v| + 1 = |t|$, as required.
- $t \in \mathcal{N}$. We have three different cases.
  - $t = x$. This case is straightforward.

- $t = uv$, with $u \in \mathcal{N}$. Since Φ is tight, then Γ is tight and we can apply Lem. 12. Then Φ necessarily ends with rule ($\mathtt{app_p}$). The i.h. then applies to the premise typing u, thus giving counters $b = e = m = 0$ and $f - 1 = |u|$. We conclude since $f = |u| + 1 = |t|$.
- $t = u[p\backslash v]$, with $u \in \mathcal{N}, v \in \mathcal{N}$. This case is similar to the third case. ◀

▶ **Lemma 14** (Substitution for System $\mathscr{E}$). *If* $\Phi_t \triangleright \Gamma; x : \mathcal{A} \vdash^{(b_t,e_t,m_t,f_t)} t : \sigma$, *and* $\Phi_u \triangleright \Delta \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$, *then there exists* $\Phi_{t\{x\backslash u\}} \triangleright \Gamma \wedge \Delta \vdash^{(b_t+b_u,e_t+e_u,m_t+m_u,f_t+f_u)} t\{x\backslash u\} : \sigma$.

Proof. We generalise the statement as follows: Let $\Phi_u \triangleright \Delta \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$.

- If $\Phi_t \triangleright \Gamma; x : \mathcal{A} \vdash^{(b_t,e_t,m_t,f_t)} t : \sigma$, then there exists $\Phi_{t\{x\backslash u\}} \triangleright \Gamma \wedge \Delta \vdash^{(b_t+b_u,e_t+e_u,m_t+m_u,f_t+f_u)} t\{x\backslash u\} : \sigma$.
- If $\Phi_t \triangleright \Gamma; x : \mathcal{A} \vdash^{(b_t,e_t,m_t,f_t)} t : \mathcal{B}$, then there exists $\Phi_{t\{x\backslash u\}} \triangleright \Gamma \wedge \Delta \vdash^{(b_t+b_u,e_t+e_u,m_t+m_u,f_t+f_u)} t\{x\backslash u\} : \mathcal{B}$.

The proof then follows by induction on Φ_t.

- If Φ_t is (ax), then we consider two cases:
  - $t = x$: then $\Phi_x \triangleright x : [\sigma'] \vdash^{(0,0,0,0)} x : \sigma'$ and $\Phi_u \triangleright \Delta \vdash^{(b_u,e_u,m_u,f_u)} u : [\sigma']$, which is a consequence of $\Delta \vdash^{(b_u,e_u,m_u,f_u)} u : \sigma'$. Then $x\{x\backslash u\} = u$, and we trivially obtain $\Phi_{t\{x\backslash u\}} \triangleright \Delta \vdash^{(0+b_u,0+e_u,0+m_u,0+f_u)} u : \sigma'$.
  - $t = y$: then $\Phi_y \triangleright y : [\sigma]; x : [\,] \vdash^{(0,0,0,0)} y : \sigma$ and $\Phi_u \triangleright \emptyset \vdash^{(0,0,0,0)} u : [\,]$. Then $y\{x\backslash u\} = y$, and we trivially obtain $\Phi_{t\{x\backslash u\}} \triangleright y : [\sigma] \vdash^{(0+0,0+0,0+0,0+0)} y : \sigma$.
- If Φ_t ends with ($\mathtt{many}$), then it has premises $(\Phi^i_t \triangleright \Gamma^i; x : \mathcal{A}^i \vdash^{(b^i_t,e^i_t,m^i_t,f^i_t)} t : \sigma^i)_{i\in I}$, where $\Gamma = \wedge_{i\in I}\Gamma^i$, $\mathcal{A} = \wedge_{i\in I}\mathcal{A}^i$, $b_t = +_{i\in I}b^i_t$, $e_t = +_{i\in I}e^i_t$, $m_t = +_{i\in I}m^i_t$, $f_t = +_{i\in I}f^i_t$ and $\mathcal{B} = [\sigma^i]_{i\in I}$. The derivation Φ_u can also be decomposed into several subderivations $(\Phi^i_u \triangleright \Delta^i \vdash^{(b^i_u,e^i_u,m^i_u,f^i_u)} u : \mathcal{A}^i)_{i\in I}$, where $b_u = +_{i\in I}b^i_u$, $e_u = +_{i\in I}e^i_u$, $m_u = +_{i\in I}m^i_u$, $f_u = +_{i\in I}f^i_u$, $\Delta = \wedge_{i\in I}\Delta^i$. We can apply the i.h. and we thus obtain derivations $(\Phi^i_{t\{x\backslash u\}} \triangleright \Gamma^i \wedge \Delta^i \vdash^{(b^i_t+b^i_u,e^i_t+e^i_u,m^i_t+m^i_u,f^i_t+f^i_u)} t\{x\backslash u\} : \sigma^i)_{i\in I}$. Then we apply rule ($\mathtt{many}$) to get $\Phi_{t\{x\backslash u\}} \triangleright \Gamma \wedge \Delta \vdash^{(b_t+b_u,e_t+e_u,m_t+m_u,f_t+f_u)} t\{x\backslash u\} : \mathcal{B}$.
- If Φ_t ends with ($\mathtt{app}$), so that $t = t'u'$, then

$$\Phi_{t'u'} \triangleright \Gamma_{t'} \wedge \Gamma_{u'}; x : \mathcal{A}_{t'} \wedge \mathcal{A}_{u'} \vdash^{(b_{t'}+b_{u'},e_{t'}+e_{u'},m_{t'}+m_{u'},f_{t'}+f_{u'})} t'u' : \sigma,$$

which follows from the two term premises $\Gamma_{t'}; x : \mathcal{A}_{t'} \vdash^{(b_{t'},e_{t'},m_{t'},f_{t'})} t' : \mathcal{B} \to \sigma$ and $\Gamma_{u'}; x : \mathcal{A}_{u'} \vdash^{(b_{u'},e_{u'},m_{u'},f_{u'})} u' : \mathcal{B}$. Also, $\Phi_u \triangleright \Delta \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$ is a consequence of $(\Delta_k \vdash^{(b^k_u,e^k_u,m^k_u,f^k_u)} u : \sigma_k)_{k\in K}$, with $\mathcal{A} = [\sigma_k]_{k\in K}$, $\Delta = \wedge_{k\in K}\Delta_k$ and $b_u = +_{k\in K}b^k_u$, $e_u = +_{k\in K}e^k_u$, $m_u = +_{k\in K}m^k_u$ and $f_u = +_{k\in K}f^k_u$. Note on the other hand that $\mathcal{A} = \mathcal{A}_{t'} \wedge \mathcal{A}_{u'} = [\sigma_i]_{i\in K_{t'}} \wedge [\sigma_i]_{i\in K_{u'}}$, with $K = K_{t'} \uplus K_{u'}$, from which one can obtain (using the $\mathtt{many}$ rule):

  - $\Delta_{t'} \vdash^{(B_{t'},E_{t'},M_{t'},F_{t'})} u : \mathcal{A}_{t'}$
  - $\Delta_{u'} \vdash^{(B_{u'},E_{u'},M_{u'},F_{u'})} u : \mathcal{A}_{u'}$

  where $b_u = B_{t'} + B_{u'} = (+_{i\in K_{t'}}b^i_u) + (+_{i\in K_{u'}}b^i_u)$, $e_u = E_{t'} + E_{u'} = (+_{i\in K_{t'}}e^i_u) + (+_{i\in K_{u'}}e^i_u)$, $m_u = M_{t'} + M_{u'} = (+_{i\in K_{t'}}m^i_u) + (+_{i\in K_{u'}}m^i_u)$, $f_u = F_{t'} + F_{u'} = (+_{i\in K_{t'}}f^i_u) + (+_{i\in K_{u'}}f^i_u)$. By the i.h. we have:

$$\Gamma_{t'} \wedge \Delta_{t'} \vdash^{(b_{t'}+B_{t'},e_{t'}+E_{t'},m_{t'}+M_{t'},f_{t'}+F_{t'})} t'\{x\backslash u\} : \mathcal{B} \to \sigma$$

$$\Gamma_{u'} \wedge \Delta_{u'} \vdash^{(b_{u'}+B_{u'},e_{u'}+E_{u'},m_{u'}+M_{u'},f_{u'}+F_{u'})} u'\{x\backslash u\} : \mathcal{B}$$

Finally, applying the app rule we obtain:

$$\Gamma_{t'} \wedge \Gamma_{u'} \wedge \Delta_{t'} \wedge \Delta_{u'} \vdash^{(b,e,m,f)} (t'\{x\backslash u\})(u'\{x\backslash u\}) : \sigma$$

with $b = b_{t'} + b_{u'} + b_u$, $e = e_{t'} + e_{u'} + e_u$, $m = m_{t'} + m_{u'} + m_u$ and $f = f_{t'} + f_{u'} + f_u$, as expected.

- If Φ_t ends with (abs), ($\mathtt{abs_p}$), ($\mathtt{app_p}$) or (match) the result follows from the i.h. by assuming α-conversion whenever necessary.
- If Φ_t ends with (pair) or ($\mathtt{pair_p}$), so that $t = \langle t', u'\rangle$, then we have two cases. The case for $\mathtt{pair_p}$, follows from Φ_t being of the form $x : [\,] \vdash^{(0,0,0,1)} \langle t', u'\rangle : \bullet_{\mathcal{M}}$, which implies $\Phi_u \triangleright \vdash^{(0,0,0,0)} u : [\,]$. Therefore $\emptyset \vdash^{(0+0,0+0,0+0,1+0)} \langle t'\{x\backslash u\}, u'\{x\backslash u\}\rangle : \bullet_{\mathcal{M}}$ holds. The case for pair follows by induction following the same reasoning used in rule app. ◀

▶ **Lemma 15** (Exact Subject Reduction). *If $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \sigma$, and $t \to_{\mathtt{h}} t'$ is an $\mathtt{s}$-step, with $\mathtt{s} \in \{\mathtt{b}, \mathtt{e}, \mathtt{m}\}$, then $\Phi' \triangleright \Gamma \vdash^{(b',e',m',f)} t' : \sigma$, where*

- *$\mathtt{s} = \mathtt{b}$ implies $b' = b - 1$, $e' = e$, $m' = m$.*
- *$\mathtt{s} = \mathtt{e}$ implies $b' = b$, $e' = e - 1$, $m' = m$.*
- *$\mathtt{s} = \mathtt{m}$ implies $b' = b$, $e' = e$, $m' = m - 1$.*

Proof. By induction on $\to_{\mathtt{h}}$.

- $t = \mathtt{L}[\![\lambda p.v]\!]u \to_{\mathtt{h}} \mathtt{L}[\![v[p\backslash u]]\!] = t'$. The proof is by induction on the list context L. We only show the case of the empty list as the other one is straightforward. The typing derivation Φ is necessarily of the form

$$\dfrac{\dfrac{\Gamma_v \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma \qquad \Gamma_v|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{A}}{\Gamma_v \lVert\!\rVert\, \mathtt{var}(p) \vdash^{(b_v+1,e_v+e_p,m_v+m_p,f_v+f_p)} \lambda p.v : \mathcal{A} \to \sigma} \qquad \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}}{\Gamma_v \lVert\!\rVert\, \mathtt{var}(p) \wedge \Gamma_u \vdash^{(b_v+1+b_u,e_v+e_p+e_u,m_v+m_p+m_u,f_v+f_p+f_u)} (\lambda p.v)u : \sigma}$$

  We then construct the following derivation Φ':

$$\dfrac{\Gamma_v \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma \qquad \Gamma_v|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{A} \qquad \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}}{\Gamma_v \lVert\!\rVert\, \mathtt{var}(p) \wedge \Gamma_u \vdash^{(b_v+b_u,e_v+e_p+e_u,m_v+m_u+m_p,f_v+f_p+f_u)} v[p\backslash u] : \sigma}$$

  The counters are as expected because the first one has decremented by 1.
- $t = v[x\backslash u] \to_{\mathtt{h}} v\{x\backslash u\} = t'$, where $v \not\to_{\mathtt{h}}$. Then Φ has two premises $\Gamma_v; x : \mathcal{A} \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma$ and $\Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$, where $\Gamma = \Gamma_v \wedge \Gamma_u$, $b = b_v + b_u$, $e = e_v + e_u + 1$, $m = m_v + m_u + 0$, and $f = f_v + f_u + 0$.
  Lem. 14 then gives a derivation ending with $\Gamma_v \wedge \Gamma_u \vdash^{(b_v+b_u,e_v+e_u,m_v+m_u,f_v+f_u)} v\{x\backslash u\} : \sigma$. The context, type, and counters are as expected.
- $t = v[\langle p_1, p_2\rangle \backslash \mathtt{L}[\![\langle u_1, u_2\rangle]\!]] \to_{\mathtt{h}} \mathtt{L}[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!] = t'$, where $v \not\to_{\mathtt{h}}$.
  Let $p = \langle p_1, p_2\rangle$ and $u = \langle u_1, u_2\rangle$. The typing derivation Φ is necessarily of the form

$$\dfrac{\Gamma_v \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma \qquad \Gamma_v|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{A} \qquad \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} \mathtt{L}[\![u]\!] : \mathcal{A}}{\Gamma_v \lVert\!\rVert\, \mathtt{var}(p) \wedge \Gamma_u \vdash^{(b_v+b_u,e_v+e_u+e_p,m_v+m_u+m_p,f_v+f_u+f_p)} v[\langle p_1, p_2\rangle \backslash \mathtt{L}[\![u]\!]] : \sigma}$$

  Then $\mathcal{A} = [\times(\mathcal{A}_1, \mathcal{A}_2)]$, for some multitypes $\mathcal{A}_1$ and $\mathcal{A}_2$, and so the pattern $\langle p_1, p_2\rangle$ is typable as follows:

$$\dfrac{\Gamma_v|_{p_1} \Vdash^{(e_1,m_1,f_1)} p_1 : \mathcal{A}_1 \qquad \Gamma_v|_{p_2} \Vdash^{(e_2,m_2,f_2)} p_2 : \mathcal{A}_2}{\Gamma_v|_p \Vdash^{(e_1+e_2,1+m_1+m_2,f_1+f_2)} \langle p_1, p_2\rangle : [\times(\mathcal{A}_1, \mathcal{A}_2)]}$$

  where $e_p = e_1 + e_2$, $m_p = 1 + m_1 + m_2$ and $f_p = f_1 + f_2$. The proof then follows by induction on list L:

- For $\mathtt{L} = \Box$ we have term u typable as follows:

$$\dfrac{\dfrac{\Gamma_1 \vdash^{(b'_1,e'_1,m'_1,f'_1)} u_1 : \mathcal{A}_1 \qquad \Gamma_2 \vdash^{(b'_2,e'_2,m'_2,f'_2)} u_2 : \mathcal{A}_2}{\Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} \langle u_1, u_2\rangle : \times(\mathcal{A}_1,\mathcal{A}_2)}}{\Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} \langle u_1, u_2\rangle : \mathcal{A}}$$

where $\Gamma_u = \Gamma_1 \wedge \Gamma_2$ and $(b_u, e_u, m_u, f_u) = (b'_1 + b'_2, e'_1 + e'_2, m'_1 + m'_2, f'_1 + f'_2)$.
We first construct the following derivation:

$$\dfrac{\Gamma_v \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma \qquad \Gamma_v|_{p_1} \Vdash^{(e_1,m_1,f_1)} p_1 : \mathcal{A}_1 \qquad \Gamma_1 \vdash^{(b'_1,e'_1,m'_1,f'_1)} u_1 : \mathcal{A}_1}{\Gamma_v \between \mathtt{var}(p_1) \wedge \Gamma_1 \vdash^{(b_v+b'_1,e_v+e'_1+e_1,m_v+m'_1+m_1,f_v+f'_1+f_1)} v[p_1\backslash u_1] : \sigma}$$

By using relevance and α-conversion to assume freshness of bound variables, we can construct a derivation with conclusion

$$\Gamma_v \between \mathtt{var}(p_1) \between \mathtt{var}(p_2) \wedge \Gamma_u \vdash^{(b',e',m',f)} v[p_1\backslash u_1][p_2\backslash u_2] : \sigma$$

where $(b', e', m', f) = (b_v + b_u, e_v + e_u + e_1 + e_2, m_v + m_u + m_1 + m_2, f_v + f_u + f_1 + f_2)$.
In order to conclude we remark the following facts:

* $\Gamma_v \between \mathtt{var}(\langle p_1, p_2\rangle) = \Gamma_v \between \mathtt{var}(p_1) \between \mathtt{var}(p_2)$
* $b_v + b_u = b_v + b'_1 + b'_2$
* $e_v + e_u + e_p = e_v + e'_1 + e'_2 + e_1 + e_2$
* $m_v + m_u + m_p = m_v + m'_1 + m'_2 + 1 + m_1 + m_2$
* $f_v + f_u + f_p = f_v + f'_1 + f'_2 + f_1 + f_2$

Then the context, type and counters are as expected.

- Let $\mathtt{L} = \mathtt{L}'[q\backslash s]$. Then Φ_u is necessarily of the following form:

$$\dfrac{\dfrac{\Delta_u \vdash^{(b'_u,e'_u,m'_u,f'_u)} \mathtt{L}'\llbracket u\rrbracket : \times(\mathcal{A}_1,\mathcal{A}_2) \qquad \Delta_u|_q \Vdash^{(e_q,m_q,f_q)} q : \mathcal{B} \qquad \Delta_s \vdash^{(b_s,e_s,m_s,f_s)} s : \mathcal{B}}{\Gamma_u \vdash^{(b'_u+b_s,e'_u+e_s+e_q,m'_u+m_s+m_q,f'_u+f_s+f_q)} \mathtt{L}'\llbracket u\rrbracket[q\backslash s] : \times(\mathcal{A}_1,\mathcal{A}_2)}}{\Gamma_u \vdash^{(b'_u+b_s,e'_u+e_s+e_q,m'_u+m_s+m_q,f'_u+f_s+f_q)} \mathtt{L}'\llbracket u\rrbracket[q\backslash s] : \mathcal{A}}$$

where $\Gamma_u = \Delta_u \between \mathtt{var}(q) \wedge \Delta_s$, $b_u = b'_u + b_s$, $e_u = e'_u + e_s + e_q$, $m_u = m'_u + m_s + m_q$ and $f_u = f'_u + f_s + f_q$.
We will apply the i.h. on the reduction step $v[p\backslash \mathtt{L}'\llbracket u\rrbracket] \to_{\mathtt{p}} \mathtt{L}'\llbracket v[p_1\backslash u_1][p_2\backslash u_2]\rrbracket$, in particular we type the left-hand side term with the following derivation Ψ_1:

$$\dfrac{\Gamma_v \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma \qquad \Gamma_v|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{A} \qquad \dfrac{\Delta_u \vdash^{(b'_u,e'_u,m'_u,f'_u)} \mathtt{L}'\llbracket u\rrbracket : \times(\mathcal{A}_1,\mathcal{A}_2)}{\Delta_u \vdash^{(b'_u,e'_u,m'_u,f'_u)} \mathtt{L}'\llbracket u\rrbracket : \mathcal{A}}}{\Gamma_v \between \mathtt{var}(p) \wedge \Delta_u \vdash^{(b_v+b'_u,e_v+e'_u+e_p,m_v+m'_u+m_p,f_v+f'_u+f_p)} v[p\backslash \mathtt{L}'\llbracket u\rrbracket] : \sigma}$$

where $b_1 = b_v + b'_u$, $e_1 = e_v + e'_u + e_p$, $m_1 = m_v + m'_u + m_p$ and $f_1 = f_v + f'_u + f_p$. The i.h. gives a derivation $\Psi_2 \triangleright \Gamma_v \between \mathtt{var}(p) \wedge \Delta_u \vdash^{(b_2,e_2,m_2,f_2)} \mathtt{L}'\llbracket v[p_1\backslash u_1][p_2\backslash u_2]\rrbracket : \sigma$ where $b_2 = b_1$, $e_2 = e_1$, $m_2 = m_1 - 1$ and $f_2 = f_1$. Let $\Lambda = \Gamma_v \between \mathtt{var}(p) \wedge \Delta_u$. We conclude with the following derivation Φ':

$$\dfrac{\Psi_2 \qquad \Delta_u|_q \Vdash^{(e_q,m_q,f_q)} q : \mathcal{B} \qquad \Delta_s \vdash^{(b_s,e_s,m_s,f_s)} s : \mathcal{B}}{\Lambda \between \mathtt{var}(q) \wedge \Delta_s \vdash^{(b_2+b_s,e_2+e_s+e_q,m_2+m_s+m_q,f_2+f_s+f_q)} \mathtt{L}'\llbracket v[p_1\backslash u_1][p_2\backslash u_2]\rrbracket[q\backslash s] : \sigma}$$

Indeed, we first remark that $\Lambda|_q = \Delta_u|_q$ holds by relevance and α-conversion. Secondly, $\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u = \Gamma_v \between \mathtt{var}(p) \wedge (\Delta_u \between \mathtt{var}(q)) \wedge \Delta_s = \Lambda \between \mathtt{var}(q) \wedge \Delta_s$ also holds by relevance and α-conversion. Last, we conclude with the following remarks:

  - $b' = b_2 + b_s = b_1 + b_s = b_v + b_u = b$
  - $e' = e_2 + e_s + e_q = e_1 + e_s + e_q = e_v + e_u + e_p = e$
  - $m' = m_2 + m_s + m_q = m_1 - 1 + m_s + m_q = m_v + m_u + m_p - 1 = m - 1$
  - $f' = f_2 + f_s + f_q = f_1 + f_s + f_q = f_v + f_u + f_p = f$

- Most of the inductive cases are straightforward, so we only show the interesting one. Let $t = v[p\backslash u] \to_{\mathtt{h}} v[p\backslash u'] = t'$, where $v \not\to_{\mathtt{h}}$ and $p \neq x$ and $u \to_{\mathtt{h}} u'$. By construction there are typing subderivations $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma$, $\Phi_p \triangleright \Gamma_v|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{A}$ and $\Phi_u \triangleright \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$. Since p is not a variable then Φ_p ends with rule $\mathtt{pat}_{\mathtt{p}}$ or $\mathtt{pat}_{\times}$. In both cases $\mathcal{A}$ contains only one type, let us say $\mathcal{A} = [\sigma_u]$. Then Φ_u has the following form

$$\frac{\Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \sigma_u}{\Phi_u \triangleright \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : [\sigma_u]}$$

  The i.h. applied to the premise of Φ_u gives a derivation $\Gamma_u \vdash^{(b'_u,e'_u,m'_u,f_u)} u' : \sigma_u$ and having the expected counters. To conclude we build a type derivation Φ' for $v[p\backslash u']$ having the expected counters. ◀

C Completeness for System $\mathscr{E}$

▶ **Lemma 17** (Canonical Forms and Tight Derivations). *Let $t \in \mathcal{M}$. There exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(0,0,0,|t|)} t : \mathtt{t}$.*

Proof. We generalise the property to the two following statements, proved by structural induction on $t \in \mathcal{N}$, $t \in \mathcal{M}$, respectively, using relevance (Lem. 8).

- If $t \in \mathcal{N}$, then there exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(0,0,0,|t|)} t : \bullet_{\mathcal{N}}$:
  - If $t = x$, then $x{:}[\bullet_{\mathcal{N}}] \vdash^{(0,0,0,0)} x : \bullet_{\mathcal{N}}$ by (ax), where $|x| = 0$.
  - If $t = u\,v$ where $u \in \mathcal{N}$, then $|t| = |u| + 1$ and by i.h. there is a tight derivation $\Phi_u \triangleright \Gamma_u \vdash^{(0,0,0,|u|)} u : \bullet_{\mathcal{N}}$. Then

$$\frac{\Gamma_u \vdash^{(0,0,0,|u|)} u : \bullet_{\mathcal{N}}}{\Gamma_u \vdash^{(0,0,0,|u|+1)} u\,v : \bullet_{\mathcal{N}}}$$

    The result then holds for $\Gamma := \Gamma_u$.
  - If $t = u[\langle p_1, p_2\rangle\backslash v]$ where $u, v \in \mathcal{N}$, then $|t| = |u| + |v| + 1$ and by i.h. there are tight derivations $\Phi_u \triangleright \Gamma_u \vdash^{(0,0,0,|u|)} u : \bullet_{\mathcal{N}}$, $\Phi_v \triangleright \Gamma_v \vdash^{(0,0,0,|v|)} v : \bullet_{\mathcal{N}}$. Then, $\Phi'_v \triangleright \Gamma_v \vdash^{(0,0,0,|v|)} v : [\bullet_{\mathcal{N}}]$ and

$$\frac{\Phi_u \qquad \Gamma_u|_{\langle p_1,p_2\rangle} \Vdash^{(0,0,1)} \langle p_1, p_2\rangle : [\bullet_{\mathcal{N}}] \qquad \Phi'_v}{(\Gamma_u \setminus\!\setminus \mathtt{var}(\langle p_1, p_2\rangle)) \wedge \Gamma_v \vdash^{(0,0,0,|u|+|v|+1)} u[\langle p_1, p_2\rangle\backslash v] : \bullet_{\mathcal{N}}}$$

    The result then holds for $\Gamma := (\Gamma_u \setminus\!\setminus \mathtt{var}(\langle p_1, p_2\rangle)) \wedge \Gamma_v$, since by i.h. $\mathtt{tight}(\Gamma_u)$ and $\mathtt{tight}(\Gamma_v)$ thus $\mathtt{tight}(\Gamma)$.
- If $t \in \mathcal{M}$, then there exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(0,0,0,|t|)} t : \mathtt{t}$.
  - If $t \in \mathcal{N}$ then by the previous item the result holds for $\mathtt{t} := \bullet_{\mathcal{N}}$.
  - If $t = \langle u, v\rangle$ then $|t| = 1$ and $\vdash^{(0,0,0,1)} \langle u, v\rangle : \bullet_{\mathcal{M}}$ by $(\mathtt{pair}_{\mathtt{p}})$. The result then holds for $\Gamma := \emptyset$.

- If $t = \lambda p.u$ where $u \in \mathcal{M}$ then $|t| = |u| + 1$ and, by i.h., there is a tight derivation $\Phi_u \rhd \Gamma_u \vdash^{(0,0,0,|u|)} u : \mathtt{t}$. Then

$$\frac{\Phi_u \rhd \Gamma_u \vdash^{(0,0,0,|u|)} u : \mathtt{t}}{\Gamma_u \setminus\!\setminus \mathtt{var}(p) \vdash^{(0,0,0,|u|+1)} \lambda p.u : \bullet_{\mathcal{M}}}$$

  The result then holds for $\Gamma := \Gamma_u \setminus\!\setminus \mathtt{var}(p)$. Observe that, since by i.h. $\mathtt{tight}(\Gamma_u)$ holds, and $\Gamma_u \setminus\!\setminus \mathtt{var}(p) \subseteq \Gamma_u$ then $\mathtt{tight}(\Gamma_u \setminus\!\setminus \mathtt{var}(p))$ trivially holds.
- If $t = u[\langle p_1, p_2\rangle \backslash v]$ where $u \in \mathcal{M}$ and $v \in \mathcal{N}$ then $|t| = |u| + |v| + 1$. Moreover, by the previous item there is a tight $\Phi_v \rhd \Gamma_v \vdash^{(0,0,0,|v|)} v : \bullet_{\mathcal{N}}$ (so that $\mathtt{tight}(\Gamma_v)$) and, by i.h., there is a tight derivation $\Phi_u \rhd \Gamma_u \vdash^{(0,0,0,|u|)} u : \mathtt{t}$. Then, $\Phi'_v \rhd \Gamma_v \vdash^{(0,0,0,|v|)} v : [\bullet_{\mathcal{N}}]$ and

$$\frac{\Phi_u \qquad (\Gamma_u)|_{\langle p,q\rangle} \Vdash^{(0,0,1)} \langle p, q\rangle : [\bullet_{\mathcal{N}}] \qquad \Phi'_v}{(\Gamma_u \setminus\!\setminus \mathtt{var}(\langle p_1, p_2\rangle)) \wedge \Gamma_v \vdash^{(0,0,0,|u|+|v|+1)} u[\langle p_1, p_2\rangle \backslash v] : \mathtt{t}}$$

  The result then holds for $\Gamma := (\Gamma_u \setminus\!\setminus \mathtt{var}(\langle p_1, p_2\rangle)) \wedge \Gamma_v$, since $\mathtt{tight}(\Gamma_v)$ as remarked, and by i.h. $\mathtt{tight}(\Gamma_u)$, thus $\mathtt{tight}(\Gamma)$. ◀

▶ **Lemma 18** (Anti-Substitution for System $\mathscr{E}$). *Let $\Phi \rhd \Gamma \vdash^{(b,e,m,f)} t\{x\backslash u\} : \sigma$. Then, there exist derivations Φ_t, Φ_u, integers $b_t, b_u, e_t, e_u, m_t, m_u, f_t, f_u$, contexts Γ_t, Γ_u, and multitype $\mathcal{A}$ such that $\Phi_t \rhd \Gamma_t; x : \mathcal{A} \vdash^{(b_t,e_t,m_t,f_t)} t : \sigma$, $\Phi_u \rhd \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$, $b = b_t + b_u$, $e = e_t + e_u$, $m = m_t + m_u$, $f = f_t + f_u$, and $\Gamma = \Gamma_t \wedge \Gamma_u$.*

Proof. As in the case of the substitution lemma, the proof follows by generalising the property for the two cases where the type derivation Φ assigns a type or a multiset type.

- Let $\Phi \rhd \Gamma \vdash^{(b,e,m,f)} t\{x\backslash u\} : \sigma$. Then, there exist derivations Φ_t, Φ_u, integers b_t, b_u, e_t, e_u, m_t, m_u, f_t, f_u, contexts Γ_t, Γ_u, and multitype $\mathcal{A}$ such that $\Phi_t \rhd \Gamma_t; x : \mathcal{A} \vdash^{(b_t,e_t,m_t,f_t)} t : \sigma$, $\Phi_u \rhd \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$, $b = b_t + b_u$, $e = e_t + e_u$, $m = m_t + m_u$, $f = f_t + f_u$, and $\Gamma = \Gamma_t \wedge \Gamma_u$.
- Let $\Phi \rhd \Gamma \vdash^{(b,e,m,f)} t\{x\backslash u\} : \mathcal{B}$. Then, there exist derivations Φ_t, Φ_u, integers b_t, b_u, e_t, e_u, m_t, m_u, f_t, f_u, contexts Γ_t, Γ_u, and multitype $\mathcal{A}$ such that $\Phi_t \rhd \Gamma_t; x : \mathcal{A} \vdash^{(b_t,e_t,m_t,f_t)} t : \mathcal{B}$, $\Phi_u \rhd \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$, $b = b_t + b_u$, $e = e_t + e_u$, $m = m_t + m_u$, $f = f_t + f_u$, and $\Gamma = \Gamma_t \wedge \Gamma_u$.

We will reason by induction on Φ. For all the rules (except $\mathtt{many}$), we will have the trivial case $t\{x\backslash u\}$, where $t = x$, in which case $t\{x\backslash u\} = u$, for which we have a derivation $\Phi \rhd \Gamma \vdash^{(b,e,m,f)} u : \sigma$. Therefore $\Phi_t \rhd x : [\sigma] \vdash^{(0,0,0,0)} x : \sigma$ and $\Phi_u \rhd \Gamma \vdash^{(b,e,m,f)} u : [\sigma]$ is obtained from Φ using the ($\mathtt{many}$) rule. The conditions on the counters hold trivially. We now reason on the different cases assuming that $t \neq x$.

- If Φ is (ax), therefore $\Phi \rhd y : [\sigma] \vdash^{(0,0,0,0)} y : \sigma$. We only consider the case where $t = y$ and $y \neq x$. Then we take $\mathcal{A} = [\,]$, $\Phi_t \rhd y : [\sigma]; x : [\,] \vdash^{(0,0,0,0)} y : \sigma$ and $\Phi_u \rhd \vdash^{(0,0,0,0)} u : [\,]$, using rule ($\mathtt{many}$). The conditions on the counters follow trivially.
- If Φ ends with ($\mathtt{many}$), then $\Phi \rhd \wedge_{k\in K}\Gamma_k \vdash^{(+_{k\in K}b_k,+_{k\in K}e_k,+_{k\in K}m_k,+_{k\in K}f_k)} t\{x\backslash u\} : [\sigma_k]_{k\in K}$ follows from $\Phi^k \rhd \Gamma_k \vdash^{(b_k,e_k,m_k,f_k)} t\{x\backslash u\} : \sigma_k$, for each $k \in K$. By the i.h. there exist Φ^k_t, Φ^k_u, b^k_t, b^k_u, e^k_t, e^k_u, m^k_t, m^k_u, f^k_t, f^k_u, contexts Γ^k_t, Γ^k_u and multitype $\mathcal{A}_k$, such that $\Phi^k_t \rhd \Gamma^k_t; x : \mathcal{A}_k \vdash^{(b^k_t,e^k_t,m^k_t,f^k_t)} t : \sigma_k$, $\Phi^k_u \rhd \Gamma^k_u \vdash^{(b^k_u,e^k_u,m^k_u,f^k_u)} u : \mathcal{A}_k$, $\Gamma_k = \Gamma^k_t \wedge \Gamma^k_u$, $b_k = b^k_t + b^k_u$, $e_k = e^k_t + e^k_u$, $m_k = m^k_t + m^k_u$, $f_k = f^k_t + f^k_u$.
  Taking $\mathcal{A} = \wedge_{k\in K}\mathcal{A}_k$ and using the ($\mathtt{many}$) rule on the derivations $(\Phi^k_t)_{k\in K}$ we get now $\wedge_{k\in K}\Gamma^k_t; x : \mathcal{A} \vdash^{(+_{k\in K}b^k_t,+_{k\in K}e^k_t,+_{k\in K}m^k_t,+_{k\in K}f^k_t)} t : [\sigma_k]_{k\in K}$. From the premises $(\Phi^k_u)_{k\in K}$, by applying again the ($\mathtt{many}$) rule, we get $\wedge_{k\in K}\Gamma^k_u \vdash^{(+_{k\in K}b^k_u,+_{k\in K}e^k_u,+_{k\in K}m^k_u,+_{k\in K}f^k_u)} u : \mathcal{A}$.

Note that $\Gamma = \wedge_{k\in K}\Gamma_k = (\wedge_{k\in K}\Gamma_t^k) \wedge (\wedge_{k\in K}\Gamma_u^k)$, $b = +_{k\in K} b_k = (+_{k\in K} b_t^k) + (+_{k\in K} b_u^k)$, $e = +_{k\in K} e_k = (+_{k\in K} e_t^k) + (+_{k\in K} e_u^k)$, $m = +_{k\in K} m_k = (+_{k\in K} m_t^k) + (+_{k\in K} m_u^k)$ and $f = +_{k\in K} f_k = (+_{k\in K} f_t^k) + (+_{k\in K} f_u^k)$, as expected.

- If Φ ends with (abs), then $t = \lambda p.t'$, therefore

$$\Phi \triangleright \Gamma \setminus\!\setminus \mathtt{var}(p) \vdash^{(b_t+1, e_t+e_p, m_t+m_p, f_t+f_p)} \lambda p.(t'\{x\backslash u\}) : \mathcal{B} \to \sigma$$

follows from $\Gamma \vdash^{(b_t,e_t,m_t,f_t)} t'\{x\backslash u\} : \sigma$ and $\Gamma|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{B}$. On the other side one can always assume that $\mathtt{var}(p) \cap \mathtt{fv}(u) = \emptyset$ and $x \notin \mathtt{var}(p)$. We can then apply the i.h. to obtain $\Phi_{t'} \triangleright \Gamma_{t'}; x : \mathcal{A} \vdash^{(b'_t,e'_t,m'_t,f'_t)} t' : \sigma$, $\Phi_u \triangleright \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$, with $\Gamma = \Gamma_{t'} \wedge \Gamma_u$, $b_t = b_{t'} + b_u$, $e_t = e_{t'} + e_u$, $m_t = m_{t'} + m_u$, $f_t = f_{t'} + f_u$. Then using rule (abs) we get $\Phi_t \triangleright \Gamma_{t'} \setminus\!\setminus \mathtt{var}(p); x : \mathcal{A} \vdash^{(b_{t'}+1, e_{t'}+e_p, m_{t'}+m_p, f_{t'}+f_p)} \lambda p.t' : \mathcal{B} \to \sigma$. And $\Gamma \setminus\!\setminus \mathtt{var}(p) = (\Gamma_{t'} \setminus\!\setminus \mathtt{var}(p)) \wedge \Gamma_u$, $b_t + 1 = b'_t + 1 + b_u$, $e_t = e_{t'} + e_u$, $m_t = m_{t'} + m_u$ and $f_t = f_{t'} + f_u$, as expected.
- If Φ ends with (app) then $t = t'u'$, and the derivation

$$\Phi \triangleright \Gamma \wedge \Delta \vdash^{(b_{t'}+b_{u'}, e_{t'}+e_{u'}, m_{t'}+m_{u'}, f_{t'}+f_{u'})} t'\{x\backslash u\}\, u'\{x\backslash u\} : \sigma$$

follows from $\Gamma \vdash^{(b_{t'},e_{t'},m_{t'},f_{t'})} t'\{x\backslash u\} : \mathcal{B} \to \sigma$ and $\Delta \vdash^{(b_{u'},e_{u'},m_{u'},f_{u'})} u'\{x\backslash u\} : \mathcal{B}$. By the i.h. there exist $\Phi_{t'}, \Phi^u_{t'}$, $b_{t''}$, $b^u_{t'}$, $e_{t''}$, $e^u_{t'}$, $m_{t''}$, $m^u_{t'}$, $f_{t''}$, $f^u_{t'}$, contexts $\Gamma_{t'}$, $\Gamma^u_{t'}$ and multitype $\mathcal{A}_{t'}$, such that

$$\Phi_{t'} \triangleright \Gamma_{t'}; x : \mathcal{A}_{t'} \vdash^{(b_{t''},e_{t''},m_{t''},f_{t''})} t' : \mathcal{B} \to \sigma \qquad \Phi^u_{t'} \triangleright \Gamma^u_{t'} \vdash^{(b^u_{t'},e^u_{t'},m^u_{t'},f^u_{t'})} u : \mathcal{A}_{t'}$$

where $b_{t'} = b_{t''} + b^u_{t'}$, $e_{t'} = e_{t''} + e^u_{t'}$, $m_{t'} = m_{t''} + m^u_{t'}$, $f_{t'} = f_{t''} + f^u_{t'}$ and $\Gamma = \Gamma_{t'} \wedge \Gamma^u_{t'}$. And by the i.h. applied to the second premise of Φ, there exist $\Phi_{u'}, \Phi^u_{u'}$, $b_{u''}$, $b^u_{u'}$, $e_{u''}$, $e^u_{u'}$, $m_{u''}$, $m^u_{u'}$, $f_{u''}$, $f^u_{u'}$, contexts $\Gamma_{u'}$, $\Gamma^u_{u'}$ and multitype $\mathcal{A}_{u'}$, such that

$$\Phi_{u'} \triangleright \Delta_{u'}; x : \mathcal{A}_{u'} \vdash^{(b_{u''},e_{u''},m_{u''},f_{u''})} u' : \mathcal{B} \qquad \Phi^u_{u'} \triangleright \Delta^u_{u'} \vdash^{(b^u_{u'},e^u_{u'},m^u_{u'},f^u_{u'})} u : \mathcal{A}_{u'}$$

where $b_{u'} = b_{u''} + b^u_{u'}$, $e_{u'} = e_{u''} + e^u_{u'}$, $m_{u'} = m_{u''} + m^u_{u'}$, $f_{u'} = f_{u''} + f^u_{u'}$ and $\Delta = \Delta_{u'} \wedge \Delta^u_{u'}$. Now, taking $\mathcal{A} = \mathcal{A}_{t'} \wedge \mathcal{A}_{u'}$, and using the (app) rule, one gets a derivation of the form $\Phi_{t'u'} \triangleright \Gamma_{t'} \wedge \Delta_{u'}; x : \mathcal{A}_{t'} \wedge \mathcal{A}_{u'} \vdash^{(b_{t''}+b_{u''}, e_{t''}+e_{u''}, m_{t''}+m_{u''}, f_{t''}+f_{u''})} t' : \mathcal{B} \to \sigma$ and applying the (many) rule to the premises of $\Phi^u_{t'}$ and $\Phi^u_{u'}$ one gets a derivation of the form $\Phi_u \triangleright \Gamma^u_{t'} \wedge \Delta^u_{u'} \vdash^{(b^u_{t'}+b^u_{u'}, e^u_{t'}+e^u_{u'}, m^u_{t'}+m^u_{u'}, f^u_{t'}+f^u_{u'})} u : \mathcal{A}$. Note that $\Gamma \wedge \Delta = (\Gamma_{t'} \wedge \Gamma^u_{t'}) \wedge (\Delta_{u'} \wedge \Delta^u_{u'}) = (\Gamma_{t'} \wedge \Delta_{u'}) \wedge (\Gamma^u_{t'} \wedge \Delta^u_{u'})$ and $b = b_{t'} + b_{u'} = (b_{t''} + b^u_{t'}) + (b_{u''} + b^u_{u'}) = (b_{t''} + b_{u''}) + (b^u_{t'} + b^u_{u'})$ as expected (the same happens for the remaining counters).
- If Φ ends with ($\mathtt{abs_p}$), ($\mathtt{app_p}$) or (match) the result follows from the inductive hypothesis, as in the previous cases.
- If Φ ends with (pair) or ($\mathtt{pair_p}$), so that $t = \langle t', u'\rangle$, then we have two cases. The case for $\mathtt{pair_p}$, follows from Φ being of the form $\vdash^{(0,0,0,1)} \langle t'\{x\backslash u\}, u'\{x\backslash u\}\rangle : \bullet_{\mathcal{M}}$. We then take $\mathcal{A} = [\,]$, $\Phi_{\langle t',u'\rangle} \triangleright x : [\,] \vdash^{(0,0,0,1)} \langle t', u'\rangle : \bullet_{\mathcal{M}}$ and $\Phi_u \triangleright \vdash^{(0,0,0,0)} u : [\,]$ follows trivially from the (many) rule. Then conditions on counters and contexts hold trivially. The case for (pair) follows by induction using the same reasoning as in rule (app). ◀

▶ **Lemma 19** (Exact Subject Expansion). *If $\Phi' \triangleright \Gamma \vdash^{(b',e',m',f')} t' : \sigma$, and $t \to_{\mathtt{h}} t'$ is an $\mathtt{s}$-step, with $\mathtt{s} \in \{\mathtt{b}, \mathtt{e}, \mathtt{m}\}$, then $\Phi \triangleright \Gamma \vdash^{(b,e,m,f)} t : \sigma$, where*

- *$\mathtt{s} = \mathtt{b}$ implies $b = b' + 1$, $e' = e$, $m' = m$.*
- *$\mathtt{s} = \mathtt{e}$ implies $b' = b$, $e = e' + 1$, $m' = m$.*
- *$\mathtt{s} = \mathtt{m}$ implies $b' = b$, $e' = e$, $m = m' + 1$.*

Proof. By induction on $\to_{\mathtt{h}}$, using Lem. 18.

- $t = \mathtt{L}[\![\lambda p.v]\!]u \to_{\mathtt{h}} \mathtt{L}[\![v[p\backslash u]]\!] = t'$. The proof is by induction on the list context $\mathtt{L}$.
  If $\mathtt{L} = \Box$ then by construction there are derivations $\Phi_v \rhd \Gamma_v \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma$ and $\Phi_u \rhd \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$ for some multitype $\mathcal{A}$, and Φ' is of the form

$$\frac{\Phi_v \qquad (\Gamma_v)|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{A} \qquad \Phi_u}{(\Gamma_v \setminus\!\setminus \mathtt{var}(p)) \wedge \Gamma_u \vdash^{(b',e',m',f)} v[p\backslash u] : \sigma}$$

  where $\Gamma = (\Gamma_v \setminus\!\setminus \mathtt{var}(p)) \wedge \Gamma_u$, $b' = b_v + b_u$, $e' = e_v + e_u + e_p$, $m' = m_v + m_u + m_p$ and $f = f_v + f_u + f_p$. Then

$$\Phi \rhd \frac{\dfrac{\Phi_v \qquad (\Gamma_v)|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{A}}{\Gamma_v \setminus\!\setminus \mathtt{var}(p) \vdash^{(b_v+1,e_v+e_p,m_v+m_p,f_v+f_p)} \lambda p.v : \mathcal{A} \to \sigma} \qquad \Phi_u}{(\Gamma_v \setminus\!\setminus \mathtt{var}(p)) \wedge \Gamma_u \vdash^{((b_v+1)+b_u,(e_v+e_p)+e_u,(m_v+m_p)+m_u,(f_v+f_p)+f_u)} (\lambda p.v)u : \sigma}$$

  where $b = (b_v + 1) + b_u = b' + 1$, $e = (e_v + e_p) + e_u = e'$ and $m = (m_v + m_p) + m_u = m'$.
  If $\mathtt{L} \neq \Box$ the proof from the i.h. is straightforward.
- $t = v[x\backslash u] \to_{\mathtt{h}} v\{x\backslash u\} = t'$, where $v \not\to_{\mathtt{h}}$. Then by the anti-substitution property (Lem. 18) there exist derivations Φ_v, Φ_u, integers $b_v, b_u, e_v, e_u, m_v, m_u, f_v, f_u$, contexts Γ_v, Γ_u, and multitype $\mathcal{A}$ such that $\Phi_v \rhd \Gamma_v; x : \mathcal{A} \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma$, $\Phi_u \rhd \Gamma_u \vdash^{(b_u,e_u,m_u,f_u)} u : \mathcal{A}$, $b' = b_v + b_u$, $e' = e_v + e_u$, $m' = m_v + m_u$, $f = f_v + f_u$, and $\Gamma = \Gamma_v \wedge \Gamma_u$. Then,

$$\Phi \rhd \frac{\Phi_v \qquad (\Gamma_v; x : \mathcal{A})|_x \Vdash^{(1,0,0)} x : \mathcal{A} \qquad \Phi_u}{\Gamma_v \wedge \Gamma_u \vdash^{(b_v+b_u,e_v+e_u+1,m_v+m_u+0,f_v+f_u+0)} v[x\backslash u] : \sigma}$$

  where $b = b_v + b_u = b'$, $e = e_v + e_u + 1 = e' + 1$ and $m = m_v + m_u = m'$. Note that $(\Gamma_v; x : \mathcal{A}) \setminus\!\setminus \mathtt{var}(x) = \Gamma_v$.
- $t = v[\langle p_1, p_2\rangle \backslash \mathtt{L}[\![\langle u_1, u_2\rangle]\!]] \to_{\mathtt{h}} \mathtt{L}[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!] = t'$, where $v \not\to_{\mathtt{h}}$. Let us abbreviate $p = \langle p_1, p_2\rangle$ and $u = \langle u_1, u_2\rangle$. The proof is by induction on the list $\mathtt{L}$.
  - $\mathtt{L} = \Box$, then there are $\Phi_v \rhd \Gamma_v \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma$, $\Phi_1 \rhd \Gamma_1 \vdash^{(b_u^1,e_u^1,m_u^1,f_u^1)} u_1 : \mathcal{A}_1$ and $\Phi_2 \rhd \Gamma_2 \vdash^{(b_u^2,e_u^2,m_u^2,f_u^2)} u_2 : \mathcal{A}_2$ where

$$\Phi_{v[p_1\backslash u_1]} \rhd \frac{\Phi_v \qquad \Gamma_v|_{p_1} \Vdash^{(e_1,m_1,f_1)} p_1 : \mathcal{A}_1 \qquad \Phi_1}{(\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \wedge \Gamma_1 \vdash^{(b_v+b_u^1,e_v+e_u^1+e_1,m_v+m_u^1+m_1,f_v+f_u^1+f_1)} v[p_1\backslash u_1] : \sigma}$$

    and

$$\Phi' \rhd \frac{\Phi_{v[p_1\backslash u_1]} \qquad ((\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \wedge \Gamma_1)|_{p_2} \Vdash^{(e_2,m_2,f_2)} p_2 : \mathcal{A}_2 \qquad \Phi_2}{\Gamma \vdash^{(b',e',m',f)} v[p_1\backslash u_1][p_2\backslash u_2] : \sigma}$$

    where $b' = b_v +_{i=1,2} b_u^i$, $e' = e_v +_{i=1,2} e_u^i + e_i$, $m' = m_v +_{i=1,2} m_u^i + m_i$, $f = f_v +_{i=1,2} f_u^i + f_i$ and $\Gamma = (((\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \wedge \Gamma_1) \setminus\!\setminus \mathtt{var}(p_2)) \wedge \Gamma_2$.
    Moreover, $(((\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \wedge \Gamma_1) \setminus\!\setminus \mathtt{var}(p_2)) = ((\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \setminus\!\setminus \mathtt{var}(p_2) \wedge \Gamma_1 \setminus\!\setminus \mathtt{var}(p_2))$, where $(\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \setminus\!\setminus \mathtt{var}(p_2) = \Gamma_v \setminus\!\setminus \mathtt{var}(\langle p_1, p_2\rangle)$ and $\Gamma_1 \setminus\!\setminus \mathtt{var}(p_2) =_{L.\ 8} \Gamma_1$. Similarly, $((\Gamma_v \setminus\!\setminus \mathtt{var}(p_1)) \wedge \Gamma_1)|_{p_2} =^{Lem.\ 8} (\Gamma_v \setminus\!\setminus \mathtt{var}(p_1))|_{p_2}$ and, by linearity of patterns, $(\Gamma_v \setminus\!\setminus \mathtt{var}(p_1))|_{p_2} = \Gamma_v|_{p_2}$. Hence,

$$\Phi_{\langle p_1,p_2\rangle} \rhd \frac{\Gamma_v|_{p_1} \Vdash^{(e_1,m_1,f_1)} p_1 : \mathcal{A}_1 \qquad \Gamma_v|_{p_2} \Vdash^{(e_2,m_2,f_2)} p_2 : \mathcal{A}_2}{\Gamma_v|_{p_1} \wedge \Gamma_v|_{p_2} \Vdash^{(e_1+e_2,1+m_1+m_2,f_1+f_2)} \langle p_1, p_2\rangle : [\times(\mathcal{A}_1, \mathcal{A}_2)]}$$

    where $\Gamma_v|_{p_1} \wedge \Gamma_v|_{p_2} = \Gamma_v|_{\langle p_1,p_2\rangle}$, and $\Phi_{\langle u_1,u_2\rangle} \rhd \Gamma_1 \wedge \Gamma_2 \vdash^{(b_u^1+b_u^2,e_u^1+e_u^2,m_u^1+m_u^2,f_u^1+f_u^2)} \langle u_1, u_2\rangle : [\times(\mathcal{A}_1, \mathcal{A}_2)]$. Therefore,

$$\Phi \triangleright \frac{\Phi_v \qquad \Phi_{\langle p_1,p_2\rangle} \qquad \Phi_{\langle u_1,u_2\rangle}}{(\Gamma_v \between \mathtt{var}(\langle p_1,p_2\rangle)) \wedge (\Gamma_1 \wedge \Gamma_2) \vdash^{(b,e,m,f)} v[\langle p_1,p_2\rangle \backslash \langle u_1,u_2\rangle] : \sigma}$$

where $b = b_v + _{i=1,2} b_u^i = b'$, $e' = e_v + _{i=1,2} e_u^i + e_i = e'$ and $m = 1 + m_v + _{i=1,2} m_u^i + m_i = m' + 1$.

- If $\mathtt{L} = \mathtt{L}'[q\backslash s]$, then Φ' is of the form:

$$\frac{\Gamma_{\mathtt{L}'} \vdash^{(b'_l,e'_l,m'_l,f'_l)} \mathtt{L}'[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!] : \sigma \qquad \Gamma_{\mathtt{L}'}|_q \Vdash^{(e_q,m_q,f_q)} q : \mathcal{A} \qquad \Gamma_s \vdash^{(b_s,e_s,m_s,f_s)} s : \mathcal{A}}{\Gamma_{\mathtt{L}'} \between \mathtt{var}(q) \wedge \Gamma_s \vdash^{(b'_l+b_s,e'_l+e_s+e_q,m'_l+m_s+m_q,f'_l+f_s+f_q)} \mathtt{L}'[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!][q\backslash s] : \sigma}$$

where $b' = b'_l + b_s$, $e' = e'_l + e_s + e_q$, $m' = m'_l + m_s + m_q$, $f' = f'_l + f_s + f_q$ and $\Gamma = \Gamma_{\mathtt{L}'} \between \mathtt{var}(q) \wedge \Gamma_s$. From $v[\langle p_1,p_2\rangle \backslash \mathtt{L}'[\![\langle u_1,u_2\rangle]\!]] \to_{\mathtt{h}} \mathtt{L}'[\![v[p_1\backslash u_1][p_2\backslash u_2]]\!]$ and derivation $\Phi'_{\mathtt{L}'}$ for the leftmost premise by the i.h. one gets $\Phi_{\mathtt{L}'} \triangleright \Gamma_{\mathtt{L}'} \vdash^{(b_l,e_l,m_l,f_l)} v[\langle p_1,p_2\rangle \backslash \mathtt{L}'[\![\langle u_1,u_2\rangle]\!]] : \sigma$ where $b_l = b'_l$, $e_l = e'_l$, $m_l = m'_l + 1$ and $f_l = f'_l$. Furthermore $\Phi_{\mathtt{L}'}$ is necessarily of the form:

$$\frac{\Gamma_v \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma \qquad \Gamma_v|_p \Vdash^{(e_p,m_p,f_p)} p : [\times(\mathcal{A}_1,\mathcal{A}_2)] \qquad \dfrac{\Gamma_u \vdash^{(b'_u,e'_u,m'_u,f'_u)} \mathtt{L}'[\![u]\!] : \times(\mathcal{A}_1,\mathcal{A}_2)}{\Gamma_u \vdash^{(b'_u,e'_u,m'_u,f'_u)} \mathtt{L}'[\![u]\!] : [\times(\mathcal{A}_1,\mathcal{A}_2)]}}{\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u \vdash^{(b_v+b'_u,e_v+e'_u+e_p,m_v+m'_u+m_p,f_v+f'_u+f_p)} v[p\backslash \mathtt{L}'[\![u]\!]] : \sigma}$$

where $b_l = b_v + b'_u$, $e_l = e_v + e'_u + e_p$, $m_l = m_v + m'_u + m_p$, $f_l = f_v + f'_u + f_p$ and $\Gamma_{\mathtt{L}'} = \Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u$. Note that, by relevance and α-conversion we have that $\Gamma_{\mathtt{L}'}|_q = \Gamma_u|_q$. Then one can construct the following derivation Φ_u:

$$\frac{\dfrac{\Gamma_u \vdash^{(b'_u,e'_u,m'_u,f'_u)} \mathtt{L}'[\![u]\!] : \times(\mathcal{A}_1,\mathcal{A}_2) \qquad \Gamma_u|_q \Vdash^{(e_q,m_q,f_q)} q : \mathcal{A} \qquad \Phi_s}{\Gamma_u \between \mathtt{var}(q) \wedge \Gamma_s \vdash^{(b'_u+b_s,e'_u+e_s+e_q,m'_u+m_s+m_q,f'_u+f_s+f_q)} \mathtt{L}'[q\backslash s][\![u]\!] : \times(\mathcal{A}_1,\mathcal{A}_2)}}{\Gamma_u \between \mathtt{var}(q) \wedge \Gamma_s \vdash^{(b'_u+b_s,e'_u+e_s+e_q,m'_u+m_s+m_q,f'_u+f_s+f_q)} \mathtt{L}'[q\backslash s][\![u]\!] : [\times(\mathcal{A}_1,\mathcal{A}_2)]}$$

where $b_u = b'_u + b_s$, $e_u = e'_u + e_s + e_q$, $m_u = m'_u + m_s + m_q$ and $f_u = f'_u + f_s + f_q$. From $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma$ and $\Pi_p \triangleright \Gamma_v|_p \Vdash^{(e_p,m_p,f_p)} p : [\times(\mathcal{A}_1,\mathcal{A}_2)]$ we build Φ:

$$\frac{\Phi_v \qquad \Pi_p \qquad \Gamma_u \between \mathtt{var}(q) \wedge \Gamma_s \vdash^{(b_u,e_u,m_u,f_u)} \mathtt{L}'[q\backslash s][\![u]\!] : [\times(\mathcal{A}_1,\mathcal{A}_2)]}{\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u \between \mathtt{var}(q) \wedge \Gamma_s \vdash^{(b_v+b_u,e_v+e_u+e_p,m_v+m_u+m_p,f_v+f_u+f_p)} v[p\backslash \mathtt{L}'[q\backslash s][\![u]\!]] : \sigma}$$

With $\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u \between \mathtt{var}(q) \wedge \Gamma_s = (\Gamma_v \between \mathtt{var}(p) \wedge \Gamma_u) \between \mathtt{var}(q) \wedge \Gamma_s = \Gamma$. Furthermore,

- $b = b_v + b_u = b_v + b'_u + b_s = b_l + b_s = b'$
- $e = e_v + e_u + e_p = e_v + e'_u + e_s + e_q + e_p = e_l + e_s + e_q = e'$
- $m = m_v + m_u + m_p = m_v + m'_u + m_s + m_q + m_p = m_l + m_s + m_q = m' + 1$
- $f = f_v + f_u + f_p = f_v + f'_u + f_s + f_q + f_p = f_l + f_s + f_q = f'$

- Most of the inductive cases are straightforward, so we only show the interesting one. Let $t = v[p\backslash u] \to_{\mathtt{h}} v[p\backslash u'] = t'$, where $v \not\to_{\mathtt{h}}$ and $p \neq x$ and $u \to_{\mathtt{h}} u'$. By construction there are subderivations $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,e_v,m_v,f_v)} v : \sigma$, $\Gamma_v|_p \Vdash^{(e_p,m_p,f_p)} p : \mathcal{A}$ and $\Phi_{u'} \triangleright \Gamma_{u'} \vdash^{(b_{u'},e_{u'},m_{u'},f_{u'})} u' : \mathcal{A}$ for some multiset $\mathcal{A}$ and $\Gamma = (\Gamma_v \between \mathtt{var}(p)) \wedge \Gamma_{u'}$. Since p is not a variable then Φ_p ends with rule ($\mathtt{pat_p}$) or ($\mathtt{pat}_\times$). In both cases $\mathcal{A}$ contains only one type, let us say $\mathcal{A} = [\sigma_{u'}]$. Then $\Phi_{u'}$ has the following form

$$\Phi_{u'} \triangleright \frac{\Gamma_{u'} \vdash^{(b_{u'},e_{u'},m_{u'},f_{u'})} u' : \sigma_{u'}}{\Gamma_{u'} \vdash^{(b_{u'},e_{u'},m_{u'},f_{u'})} u' : [\sigma_{u'}]}$$

The i.h. applied to the premise of $\Phi_{u'}$ gives a derivation $\Gamma_{u'} \vdash^{(b_u,e_u,m_u,f_u)} u : \sigma_{u'}$ and having the expected counters. To conclude we build a type derivation Φ' for $v[p\backslash u']$ having the expected counters. ◀