Quantitative Global Memory

Sandra Alves[1,⋆], Delia Kesner[2,3], and Miguel Ramos[4] [⋆⋆]

[1] CRACS/INESC-TEC, DCC, Faculty of Science, University of Porto
[2] Université Paris Cité, CNRS, IRIF
[3] Institut Universitaire de France
[4] LIACC, DCC, Faculty of Science, University of Porto

Abstract. We show that recent approaches of static analysis based on quantitative typing systems can be extended to programming languages with global state. More precisely, we define a call-by-value language equipped with operations to access a global memory, together with a semantic model based on a (tight) multi-type system that captures exact measures of time and space related to evaluation of programs. We show that the type system is quantitatively sound and complete with respect to the original operational semantics of the language.

1 Introduction

The aim of this paper is to extend *quantitative* techniques of *static analysis* based on *multi-types* to programs with *effects*.

Effectful Programs. Programming languages produce different kind of *effects* (observable interactions with the environment), such as handling exceptions, read/write from a global memory outside its own scope, using a database or a file, performing non-deterministic choices, or using sample probabilistic functions. The degree to which these side effects are used depends on each programming paradigm [21] (imperative programming makes use of them while declarative programming does not). In general, avoiding the use of side effects facilitates the formal verification of programs, thus allowing to (statically) ensure their correctness. Thus for example, the functional language Haskell eliminates side effects by replacing them with *monadic* actions, a clean approach which continues to attract growing attention. Indeed, rather than writing a function that returns a raw type, an effectful function returns a raw type inside a useful wrapper – where that wrapper is a monad [31]. This approach allows programming languages to combine the qualities of both the imperative and declarative worlds: programs produce effects, but these are encoded in such a way that formal verifications can be performed very conveniently.

[⋆] Supported by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project LA/P/0063/2020.

[⋆⋆] Supported by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, through the individual research grant 2021.04731.BD.

Quantitative Properties. We address quantitative properties of programs with effects using *multi-types*, which originate in the theory of *intersection* type systems. They extend simple types with a new constructor $\cap$ in such a way that a program t is typable with $\sigma \cap \tau$ if t is typable with both types σ and τ independently. Intersection types were originally introduced as *models* capturing computational properties of functional programming in a broader sense [12]. For example, termination of different evaluations strategies can be characterized by typability in some appropriate intersection type system: a program t is terminating if and only if t is typable. Originally, intersection enjoys associativity, commutativity, and in particular idempotency (*i.e.* $\sigma \cap \sigma = \sigma$). By changing to a *non-idempotent* intersection constructor, one naturally comes to represent types by multisets, which is why they are called multi-types. Just like their idempotent precursors, multi-types still allow for a characterization of several operational properties of programs, but they also grant a substantial improvement: they provide quantitative measures about these properties. For example, it is still possible to prove that a program is terminating if and only if it is typable, but now an *upper bound* or *exact measure* for the time needed for its evaluation length can be derived from the typing derivation of the program. This shift of perspective, from idempotent to non-idempotent types, goes beyond lowering the logical complexity of the proof: the quantitative information provided by typing derivations in the non-idempotent setting unveils crucial quantitative relations between typing (static) and reduction (dynamic) of programs.

Upper Bounds and Exact Split Measures. Multi-types are extensively used to reason about programming languages from a quantitative point of view, as pioneered by de Carvalho [10,11]. For example, they are able to provide *upper bounds*, in the sense that the evaluation length of a program t *plus* the size of its result (called *normal form*) can be bounded by the size of the type derivation of t. A major drawback of this approach, however, is that the size of normal forms can be exponentially bigger than the length of the evaluation reaching those normal forms. This means that bounding the sum of these two integers at the same time is too rough, and not very relevant from a quantitative point of view. Fortunately, it is possible to extract better measures from a multi-type system. A crucial point to obtain *exact measures*, instead of upper bounds, is to consider minimal type derivations, called *tight*. Moreover, using appropriate refined tight systems it is also possible to obtain *independent* measures (called *split* exact measures) for *length* and for *size*. More precisely, the quantitative typing systems are now equipped with constants and counters, together with an appropriate notion of tightness, which encodes minimality of type derivations. For any tight type derivation Φ of a program t with counters b and d, it is now possible to show that t evaluates to a normal form of size d in exactly b steps. Therefore, the type system is not only *sound*, *i.e.* it is able to *guess* the number of steps to normal form as well as the size of this normal form, but the opposite direction providing *completeness* of the approach also holds.

Contribution. The focus of this paper is on effectful computations such as reading and writing on a global memory. Taking inspiration from the monadic approach adopted in [14], we design a tight quantitative type system that provides split exact measures. More precisely, our system is not only capable of discriminating between length of evaluation to normal form and size of the normal form, but the measure corresponding to the length of the evaluation is split into two different integers: the first one corresponds to the length of standard computation (β-reduction) and the second one to the number of memory accesses. We show that the system is sound *i.e.* for any tight type derivation Φ of t ending with counters (b, m, d), the term t is normalisable by performing b evaluation steps and m memory accesses, yielding a normal form having size d. The opposite direction, giving completeness of the model, is also proved.

In order to gradually present the material, we first develop the technique for a weak (open) call-by-value (CBV) calculus, which can be seen as a contribution per se, and then we encapsulate these preliminary ideas in the general framework of the language with global state.

Summary. Sec. 2 illustrates the technique on a weak (open) CBV calculus. We then lift the technique to the λ-calculus with global state in Sec. 3 by following the same methodology. More precisely, Sec. 3.1 introduces the λ_{gs}-calculus, Sec. 3.2 defines a quantitative type system $\mathcal{P}$. Soundness and completeness of $\mathcal{P}$ w.r.t. λ_{gs} are proved in Sec. 3.3. We conclude and discuss related work in Sec. 4. Due to space limitation we cannot include all proofs, but they are available in [5].

Preliminary General Notations. We start with some general notations. Given a (one-step) reduction relation $\rightarrow_{\mathcal{R}}$, $\twoheadrightarrow_{\mathcal{R}}$ denotes the reflexive-transitive closure of $\rightarrow_{\mathcal{R}}$. We write $t \twoheadrightarrow^b u$ for a reduction sequence from t to u of length b. A term t is said to be (1) in **$\mathcal{R}$-normal form** (written $t \not\rightarrow_{\mathcal{R}}$) iff there is no u such that $t \rightarrow_{\mathcal{R}} u$, (2) **$\mathcal{R}$-weakly normalizing** (written $t \in \mathcal{WN}(\mathcal{R})$) iff there is some $\mathcal{R}$-nf u such that $t \twoheadrightarrow_{\mathcal{R}} u$, (3) **$\mathcal{R}$-strongly normalizing** (written $t \in \mathcal{SN}(\mathcal{R})$) iff there is no infinite $\mathcal{R}$-reduction sequence starting at t. $\mathcal{R}$ is weakly (resp. strongly) normalizing iff every term is $\mathcal{R}$-weakly (resp. $\mathcal{R}$-strongly) normalizing.

2 Weak Open CBV

In this section we first introduce the technique of tight typing on a simple language without effects, the weak open CBV. Sec. 2.1 defines the syntax and operational semantics of the language, Sec. 2.2 presents the tight typing system $\mathcal{O}$ and discusses soundness and completeness of $\mathcal{O}$ w.r.t. the CBV language.

2.1 Syntax and Operational Semantics

Weak open CBV is based on two principles: reduction is *weak* (not performed inside abstractions), and terms are *open* (may contain free variables). **Value**, **terms** and **weak contexts** are given by the following grammars, respectively:

$$v, w ::= x \mid \lambda x.t \qquad t, u, p ::= v \mid tu \qquad \mathcal{W} ::= \Box \mid \mathcal{W}t \mid t\mathcal{W}$$

We write $\mathtt{Val}$ for the set of all values. Notation $\mathtt{I}$ denotes the identity function $\lambda z.z$. The sets of **free** and **bound** variables of terms and the notion of

α-conversion are defined as usual. A term t is said to be **closed** if t does not contain any free variable, and **open** otherwise. The **size of a term** t, denoted $|t|$, is given by: $|x| = |\lambda x.t| = 0$; and $|tu| = 1 + |t| + |u|$.

We now introduce the operational semantics of our language, which models the core behavior of programming languages such as OCaml, where CBV evaluation is *weak*. Indeed, the **deterministic reduction relation** (written $\rightarrow$), is given by the following rules:

$$\frac{}{(\lambda x.t)v \rightarrow t\{x\backslash v\}}\ (\beta_{\mathtt{v}}) \qquad \frac{t \rightarrow t'}{tu \rightarrow t'u}\ (\mathtt{appL}) \qquad \frac{t \not\rightarrow \quad u \rightarrow u'}{tu \rightarrow tu'}\ (\mathtt{appR})$$

Terms in $\rightarrow$-normal form can be characterized by the following grammars: $\mathtt{no} ::= \mathtt{Val} \mid \mathtt{ne}$ and $\mathtt{ne} ::= x\ \mathtt{no} \mid \mathtt{no}\ \mathtt{ne} \mid \mathtt{ne}\ \mathtt{no}$.

Proposition 1. *Let t be a term. Then $t \in \mathtt{no}$ iff $t \not\rightarrow$ $\mathtt{no}$.*

In closed CBV [28] (only reducing closed terms), abstractions are the only normal forms, but in open CBV, the following terms turn out to be also acceptable normal forms: xy, $x(\lambda y.y(\lambda z.z))$ and $(\lambda x.x)(y(\lambda z.z))$.

2.2 A Quantitative Type System for the Weak Open CBV

The *untyped* λ-calculus can be interpreted as a *typed* calculus with a single type D, where $D = D \Rightarrow D$ [30]. Applying Girard's [19] *boring* CBV translation of intuitionistic logic into linear logic, we get $D = !D \multimap !D$ [1]. Type system $\mathcal{O}$ is built having this equation in mind.

The **set of types** is given by the following grammar:

(Tight Constants)	$\mathtt{tt} ::= \mathtt{v} \mid \mathtt{a} \mid \mathtt{n}$
(Value Types)	$\sigma ::= \mathtt{v} \mid \mathtt{a} \mid \mathcal{M} \mid \mathcal{M} \Rightarrow \tau$
(Multi-Types)	$\mathcal{M} ::= [\sigma_i]_{i \in I}$ where I is a finite set
(Types)	$\tau ::= \mathtt{n} \mid \sigma$

Tight types are minimal types assigned to terms reducing to normal forms ($\mathtt{v}$ for variables, $\mathtt{a}$ for abstractions, and $\mathtt{n}$ for neutral terms). Given an arbitrary tight type $\mathtt{tt}_0$, we write $\overline{\mathtt{tt}_0}$ to denote all the other tight types in $\mathtt{tt}$ different from $\mathtt{tt}_0$. Multi-types are multisets of types. A **(typing) environment**, written Γ, Δ, is a function from variables to multi-types, assigning the empty multi-type $[\,]$ to all but a finite set of variables. The domain of Γ is $\mathtt{dom}(\Gamma) := \{x \mid \Gamma(x) \neq [\,]\}$. The **union** of environments, written $\Gamma + \Delta$, is defined by $(\Gamma + \Delta)(x) = \Gamma(x) \sqcup \Delta(x)$, where $\sqcup$ denotes **multiset union**. An example is $(x : [\sigma_1], y : [\sigma_2]) + (x : [\sigma_1], z : [\sigma_2]) = (x : [\sigma_1, \sigma_1], y : [\sigma_2], z : [\sigma_2])$. This notion is extended to a finite union of environments, written $+_{i \in I}\Gamma_i$ (the empty environment is obtained when $I = \emptyset$). We write $\Gamma \setminus\!\!\setminus x$ for the environment $(\Gamma \setminus\!\!\setminus x)(x) = [\,]$ and $(\Gamma \setminus\!\!\setminus x)(y) = \Gamma(y)$ if $y \neq x$ and we write $\Gamma; x : \mathcal{M}$ for $\Gamma + (x : \mathcal{M})$, when $x \notin \mathtt{dom}(\Gamma)$. Notice that Γ and $\Gamma; x : [\,]$ are the same environment.

A **judgement** has the form $\Gamma \vdash^{(b,s)} t : \tau$, where b, s are two integers. The **typing system** $\mathcal{O}$ is defined by the rules in Fig. 1. We write $\triangleright \Gamma \vdash^{(b,s)} t : \tau$ if there is a (tree) **type derivation** of the judgement $\Gamma \vdash^{(b,s)} t : \tau$ using the rules of system $\mathcal{O}$. The term t is **$\mathcal{O}$-typable** (we may omit the name $\mathcal{O}$) iff there is an environment Γ, a type τ and counters (b,s) such that $\triangleright \Gamma \vdash^{(b,s)} t : \tau$. We use letters $\Phi, \Psi, \ldots$ to name type derivations, by writing for example $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \tau$. Notice that in rule $(\mathtt{ax})$ of Fig. 1 variables can only be assigned value types σ

$$\frac{}{x : [\sigma] \vdash^{(0,0)} x : \sigma}\ (\mathtt{ax}) \qquad \frac{\Gamma \vdash^{(b,s)} t : \tau}{\Gamma \setminus\!\!\setminus x \vdash^{(b,s)} \lambda x.t : \Gamma(x) \Rightarrow \tau}\ (\lambda)$$

$$\frac{\Gamma \vdash^{(b,s)} t : \mathcal{M} \Rightarrow \tau \qquad \Delta \vdash^{(b',s')} u : \mathcal{M}}{\Gamma + \Delta \vdash^{(1+b+b',s+s')} tu : \tau}\ (\mathtt{@}) \qquad \frac{(\Gamma_i \vdash^{(b_i,s_i)} v : \sigma_i)_{i \in I}}{+_{i\in I}\Gamma_i \vdash^{(+_{i\in I} b_i, +_{i \in I} s_i)} v : [\sigma_i]_{i\in I}}\ (\mathtt{m})$$

$$\frac{}{\vdash^{(0,0)} \lambda x.t : \mathtt{a}}\ (\lambda_{\mathtt{p}})$$

$$\frac{\Gamma \vdash^{(b,s)} t : \bar{\mathtt{a}} \qquad \Delta \vdash^{(b',s')} u : \mathtt{tt}}{\Gamma + \Delta \vdash^{(b+b',1+s+s')} tu : \mathtt{n}}\ (\mathtt{@}_{\mathtt{p1}}) \qquad \frac{\Gamma \vdash^{(b,s)} t : \mathtt{tt} \qquad \Delta \vdash^{(b',s')} u : \mathtt{n}}{\Gamma + \Delta \vdash^{(b+b',1+s+s')} tu : \mathtt{n}}\ (\mathtt{@}_{\mathtt{p2}})$$

Fig. 1. Typing Rules of system $\mathcal{O}$

(in particular no type $\mathtt{n}$): this is because they can only be substituted by values. Due to this fact, multi-types only contain value types. Regarding typing rules $(\mathtt{ax})$, (λ), $(\mathtt{@})$, and $(\mathtt{m})$, they are the usual rules for non-idempotent intersection types [8]. Rules $(\lambda_{\mathtt{p}})$, $(\mathtt{@}_{\mathtt{p1}})$, and $(\mathtt{@}_{\mathtt{p2}})$ are used to type *persistent* symbols, *i.e.* symbols that are not going to be *consumed* during evaluation. More specifically, rule $(\lambda_{\mathtt{p}})$ types abstractions (with type $\mathtt{a}$) that are normal regardless of the typability of its body. Rule $(\mathtt{@}_{\mathtt{p1}})$ types applications that will never reduce to an abstraction on the left (thus of any tight type that is not $\mathtt{a}$, *i.e.* $\bar{\mathtt{a}}$), while any term reducing to a normal form is allowed on the right (of tight type $\mathtt{tt}$). Rule $(\mathtt{@}_{\mathtt{p2}})$ also types applications, but when values will never be obtained on the right (only neutral terms of type $\mathtt{n}$). Rule $(\mathtt{ax})$ is also used to type persistent variables, in particular when $\sigma \in \{\mathtt{v}, \mathtt{a}\}$.

A **type** τ is **tight** if $\tau \in \mathtt{tt}$. We write $\mathtt{tight}(\mathcal{M})$, if every $\sigma \in \mathcal{M}$ is tight. A **type environment** Γ is **tight** if it assigns tight multi-types to all variables. A **type derivation** $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \tau$ is **tight** if Γ and τ are both tight.

Example 1. Let $t = (\lambda x.x(yz))(\lambda z.z)$. Let Φ be the following typing derivation:

$$\dfrac{\dfrac{\dfrac{}{x : [\mathtt{a}] \vdash^{(0,0)} x : \mathtt{a}}\ (\mathtt{ax}) \qquad \dfrac{\dfrac{}{y : [\mathtt{v}] \vdash^{(0,0)} y : \mathtt{v}}\ (\mathtt{ax}) \qquad \dfrac{}{z : [\mathtt{v}] \vdash^{(0,0)} z : \mathtt{v}}\ (\mathtt{ax})}{y : [\mathtt{v}], z : [\mathtt{v}] \vdash^{(0,1)} yz : \mathtt{n}}\ (\mathtt{@}_{\mathtt{p1}})}{x : [\mathtt{a}], y : [\mathtt{v}], z : [\mathtt{v}] \vdash^{(0,2)} x(yz) : \mathtt{n}}\ (\mathtt{@}_{\mathtt{p2}})}{y : [\mathtt{v}], z : [\mathtt{v}] \vdash^{(0,2)} \lambda x.x(yz) : [\mathtt{a}] \Rightarrow \mathtt{n}}\ (\lambda)$$

Then, we can build the following tight typing derivation Φ_t for t:

$$\dfrac{\Phi \quad \dfrac{\dfrac{}{\vdash^{(0,0)} \lambda z.z : \mathtt{a}}\ (\lambda_{\mathtt{p}})}{\vdash^{(0,0)} \lambda z.z : [\mathtt{a}]}\ (\mathtt{m})}{y : [\mathtt{v}], z : [\mathtt{v}] \vdash^{(1,2)} (\lambda x.x(yz))(\lambda z.z) : \mathtt{n}}\ (\mathtt{@})$$

The type system $\mathcal{O}$ can be shown to be *sound* and *complete* w.r.t. the operational semantics $\to$. Soundness means that not only a *tightly* typable term t is terminating, but also that the *tight* type derivation of t gives exact and split measures concerning the reduction sequence from t to normal form. More precisely, if $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \tau$ is tight, then there exists $u \in \mathtt{no}$ such that $t \twoheadrightarrow^{b} u$ with $|u| = s$. Dually for *completeness.* Because we are going to show this kind of properties for the more sophisticated language with global state (Sec. 3.3), we do not give here technical details of them. However, we highlight these properties on our previous example. Consider again term t in Ex. 1 and its tight derivation Φ_t with counters $(b, s) = (1, 2)$. Counter b is different from 0, so $t \notin \mathtt{no}$, but t normalizes in one β_v-step ($b = 1$) to a normal form having size $s = 2 = |(\lambda z.z)(yz)|$.

3 A λ-Calculus with Global State

Based on the preliminary presentation of Sec. 2, we now introduce a λ-calculus with global state following a CBV strategy. Sec. 3.1 defines the syntax and operational semantics of the λ-calculus with global state. Sec. 3.2 presents the tight typing system $\mathcal{P}$, and Sec. 3.3 shows soundness and completeness.

3.1 Syntax and Operational Semantics

Values, **terms**, **states** and **configurations** of $\lambda_{\mathtt{gs}}$ are defined resp. as follows:

$$\begin{array}{rlrl} v, w ::= & x \mid \lambda x.t & t, u, p ::= & v \mid vt \mid \mathtt{get}_l(\lambda x.t) \mid \mathtt{set}_l(v, t) \\ s, q ::= & \epsilon \mid \mathtt{upd}_l(v, t) & c ::= & (t, s) \end{array}$$

Notice that applications are restricted to the form vt. This, combined with the use of a deterministic reduction strategy based on weak contexts, ensures that composition of effects is well behaved. Indeed, this kind of restriction is usual in computational calculi [27,29,14,17].

The size function is extended to states and configurations: $|s| := 0$, and $|(t, s)| := |t|$. The update constructor is commutative in the following sense:

$$\mathtt{upd}_l(v, \mathtt{upd}_{l'}(w, s)) \equiv_{\mathtt{c}} \mathtt{upd}_{l'}(w, \mathtt{upd}_l(v, s)) \text{ if } l \neq l'$$

We denote by $\equiv$ the equivalence relation generated by the axiom $\equiv_{\mathtt{c}}$. We write $l \in \mathtt{dom}(s)$, if $s \equiv \mathtt{upd}_l(v, q)$, for some value v and store q. Moreover, these v and q are *unique.* For example, if $l_1 \neq l_2$, then $s_1 = \mathtt{upd}_{l_1}(v_1, \mathtt{upd}_{l_2}(v_2, q)) \equiv \mathtt{upd}_{l_2}(v_2, \mathtt{upd}_{l_1}(v_1, q)) = s_2$, but $\mathtt{upd}_{l_1}(v_1, \mathtt{upd}_{l_1}(v_2, s)) \not\equiv \mathtt{upd}_{l_1}(v_2, \mathtt{upd}_{l_1}(s,))$. As a consequence, whenever we want to access the content of a particular location in a state, we can simply assume that the location is at the top of the state.

The operational semantics of the $\lambda_{\texttt{gs}}$-calculus is given on configurations. The **deterministic reduction relation** $\rightarrow$ is defined to be the union of the rules $\rightarrow_{\texttt{r}}$ ($\texttt{r} \in \{\beta_v, \texttt{g}, \texttt{s}\}$) below. We write $(t, s) \twoheadrightarrow^{(b,m)} (u, q)$ if (t, s) reduces to (u, q) in b β_v-steps and m $\texttt{g}/\texttt{s}$-steps.

$$\frac{}{((\lambda x.t)v, s) \rightarrow_{\beta_v} (t\{x\backslash v\}, s)}\ (\beta_{\mathtt{v}}) \qquad \frac{s \equiv \texttt{upd}_l(v, q)}{(\texttt{get}_l(\lambda x.t), s) \rightarrow_{\texttt{g}} (t\{x\backslash v\}, s)}\ (\texttt{get})$$

$$\frac{(t, s) \rightarrow_{\texttt{r}} (u, q) \qquad \texttt{r} \in \{\beta_v, \texttt{g}, \texttt{s}\}}{(vt, s) \rightarrow_{\texttt{r}} (vu, q)}\ (\texttt{appR}) \qquad \frac{}{(\texttt{set}_l(v, t), s) \rightarrow_{\texttt{s}} (t, \texttt{upd}_l(v, s))}\ (\texttt{set})$$

Example 2. Consider the configuration $c_0 = ((\lambda x.\texttt{get}_l(\lambda y.yx))(\texttt{set}_l(\texttt{I}, z)), \epsilon)$. Then we can reach an irreducible configuration as follows:

$$\begin{array}{l} ((\lambda x.\texttt{get}_l(\lambda y.yx))(\texttt{set}_l(\texttt{I}, z)), \epsilon) \rightarrow_{\texttt{g}} ((\lambda x.\texttt{get}_l(\lambda y.yx))z, \texttt{upd}_l(\texttt{I}, \epsilon)) \\ \rightarrow_{\beta_v} (\texttt{get}_l(\lambda y.yz), \texttt{upd}_l(\texttt{I}, \epsilon)) \rightarrow_{\texttt{g}} (\texttt{I}z, \texttt{upd}_l(\texttt{I}, \epsilon)) \rightarrow_{\beta_v} (z, \texttt{upd}_l(\texttt{I}, \epsilon)) \end{array}$$

A configuration (t, s) is said to be **blocked** if either $t = \texttt{get}_l(\lambda x.u)$ and $l \notin \texttt{dom}(s)$; or $t = vu$ and (u, s) is blocked. A configuration is **unblocked** if it is not blocked. As an example, $(\texttt{get}_l(\lambda x.x), \epsilon)$ is obviously blocked. As a consequence, the following configuration reduces to a blocked one: $((\lambda y.y\ \texttt{get}_l(\lambda x.x))z, \epsilon) \rightarrow (z\ \texttt{get}_l(\lambda x.x), \epsilon)$. This suggest a notion of **final configuration**: (t, s) is **final** if either (t, s) is blocked; or $t \in \texttt{no}$, where **neutral** and **normal** terms are given resp. by the grammars $\texttt{ne} ::= x\ \texttt{no} \mid (\lambda x.t)\ \texttt{ne}$ and $\texttt{no} ::= \texttt{Val} \mid \texttt{ne}$.

Proposition 2. *Let (t, s) be a configuration. Then (t, s) is final iff $(t, s) \not\rightarrow$.*

Notice that when (t, s) is an unblocked final configuration, then $t \in \texttt{no}$. These are the configurations captured by the typing system $\mathcal{P}$ in Sec. 3.2. Consider the final configurations $c_0 = (\texttt{get}_l(\lambda x.x), \epsilon)$, $c_1 = (z\ \texttt{get}_l(\lambda x.x), \epsilon)$, $c_2 = (y, s)$ and $c_3 = ((\lambda x.x)(yz), s)$. Then c_0 and c_1 are blocked, while c_2 and c_3 are unblocked.

3.2 A Quantitative Type System for the $\lambda_{\texttt{gs}}$-calculus

We now introduce the quantitative type system $\mathcal{P}$ for $\lambda_{\texttt{gs}}$. To deal with global states, we extend the language of types with the notions of state, configuration and monadic types. To do this, we translate linear arrow types according to Moggi's [27] CBV interpretation of reflexive objects in the category of λ_c-models: $D = !D \multimap !D$ becomes $D = !D \multimap T(!D)$, where T a functor. Type system $\mathcal{P}$ was built having this equation in mind, similarly to what was done in [18].

The **set of types** is given by the following grammar:

(Tight Constants)	$\texttt{tt} ::= \texttt{v} \mid \texttt{a} \mid \texttt{n}$
(Value Types)	$\sigma ::= \texttt{v} \mid \texttt{a} \mid \mathcal{M} \mid \mathcal{M} \Rightarrow \delta$
(Multi-types)	$\mathcal{M} ::= [\sigma_i]_{i \in I}$ where I is a finite set
(Types)	$\tau ::= \texttt{n} \mid \sigma$
(State Types)	$\mathcal{S} ::= \{(l_i : \mathcal{M}_i)\}_{i \in I}$ where all l_i are distinct
(Configuration Types)	$\kappa ::= \tau \times \mathcal{S}$
(Monadic Types)	$\delta ::= \mathcal{S} \gg \kappa$

In system $\mathcal{P}$, the minimal types to be assigned to normal forms distinguish between variables ($\mathtt{v}$), abstractions ($\mathtt{a}$), and neutral terms ($\mathtt{n}$). A **multi-type** is a multi-set of value types. A **state type** is a partial function mapping labels to (possibly empty) multi-types. A **configuration type** is a product type, where the first component is a type and the second is a state type. A **monadic type** associates a state type to a configuration type. We use the notation $\mathcal{T}$ to denote a value type or a monadic type. **Typing environments** and operations over types are defined in the same way as in system $\mathcal{O}$.

The **domain** of a state type $\mathcal{S}$ is the set of all its labels, *i.e.* $\mathtt{dom}(\mathcal{S}) := \{l \mid (l : \mathcal{M}) \in \mathcal{S}\}$. Also, when $l \in \mathtt{dom}(\mathcal{S})$, *i.e.* $(l : \mathcal{M}) \in \mathcal{S}$, we write $\mathcal{S}(l)$ to denote $\mathcal{M}$. The **union of state types** is defined as follows:

$$\begin{array}{c}(\mathcal{S} \Cup \mathcal{S}')(l) = \text{if } (l : \mathcal{M}) \in \mathcal{S} \text{ then (if } (l : \mathcal{M}') \in \mathcal{S}' \text{ then } \mathcal{M} \sqcup \mathcal{M}' \text{ else } \mathcal{M})\\ \text{else (if } (l : \mathcal{M}') \in \mathcal{S}' \text{ then } \mathcal{M}' \text{ else } \mathtt{undefined})\end{array}$$

Example 3. Let $\mathcal{S} = \{(l_1 : [\sigma_1, \sigma_2]), (l_2 : [\sigma_1])\} \Cup \{(l_2 : [\sigma_1, \sigma_2]), (l_3 : [\sigma_3])\}$. Then, $\mathcal{S}(l_1) = [\sigma_1, \sigma_2]$, $\mathcal{S}(l_2) = [\sigma_1, \sigma_1, \sigma_2]$, $\mathcal{S}(l_3) = [\sigma_3]$, and $\mathcal{S}(l) = \mathtt{undefined}$, assuming $l \neq l_i$, for $i \in \{1, 2, 3\}$.

Remark 1. Notice that $\mathtt{dom}(\mathcal{S} \Cup \mathcal{S}') = \mathtt{dom}(\mathcal{S}) \cup \mathtt{dom}(\mathcal{S}')$. Also $\{(l : [\,])\} \Cup \mathcal{S} \neq \mathcal{S}$, if $l \notin \mathtt{dom}(\mathcal{S})$, while $x : [\,]; \Gamma = \Gamma$. Indeed, typing environments are total functions, where variables mapped to $[\,]$ do not occur in typed programs. In contrast, states are partial functions, where labels mapped to $[\,]$ correspond to positions in memory that are accessed (by get or set), but ignored/discarded by the typed program. We use $\{(l : \mathcal{M})\}; \mathcal{S}$ for $\{(l : \mathcal{M})\} \Cup \mathcal{S}$ if $l \notin \mathtt{dom}(\mathcal{S})$.

A **term type judgement** (resp. **state type judgment** and **configuration type judgment**) has the form $\Gamma \vdash^{(b,m,d)} t : \mathcal{T}$ (resp. $\Gamma \vdash^{(b,m,d)} s : \mathcal{S}$ and $\Gamma \vdash^{(b,m,d)} (t, s) : \kappa$) where b, m, d are three integers. The **typing system** $\mathcal{P}$ is defined by the rules in Fig. 2. We write $\triangleright \mathcal{J}$ if there is a type derivation of the judgement $\mathcal{J}$ using the rules of system $\mathcal{P}$. The term t (resp. state s, configuration (t, s)) is $\mathcal{P}$**-typable** iff there is an environment Γ, a type $\mathcal{T}$ (resp. $\mathcal{S}$, κ) and counters (b, m, d) such that $\triangleright \Gamma \vdash^{(b,m,d)} t : \mathcal{T}$ (resp. $\triangleright \Gamma \vdash^{(b,m,d)} s : \mathcal{S}$, $\triangleright \Gamma \vdash^{(b,m,d)} (t, s) : \kappa$). As before, we use letters $\Phi, \Psi, \ldots$ to name type derivations.

Rules ($\mathtt{ax}$), (λ), ($\mathtt{m}$), and ($\mathtt{@}$) are essentially the same as in Fig. 1, but with types lifted to monadic types (*i.e. decorated* with state types). Rule ($\mathtt{@}$) assumes a value type associated to a value v on the left premise and a monadic type associated to a term t on the right premise. To type the application vt, it is necessary to match both the value type $\mathcal{M}$ inside the type of t with the input value type of v, and the output state type $\mathcal{S}'$ of t with the input state type of v. Rule ($\uparrow$) is used to lift multi-types (the type of values) to monadic types. Rules ($\mathtt{get}$) and ($\mathtt{set}$) are used to type operations over the state. While there was just one single typing rule in system $\mathcal{O}$ (Sec. 2.2) to type both consuming and persistent variables, we now need to add an explicit persistent rule ($\mathtt{ax_p}$) to type variables with lifted type $\mathcal{S} \gg (\overline{\mathtt{n}} \times \mathcal{S})$. Rule ($\mathtt{emp}$) types empty states, rule ($\mathtt{upd}$) types states, and ($\mathtt{conf}$) types configurations.

A **type** τ is **tight**, if $\tau \in \mathtt{tt}$. We write $\mathtt{tight}(\mathcal{M})$ if every $\sigma \in \mathcal{M}$ is tight. A **state type** $\mathcal{S}$ is **tight** if $\mathtt{tight}(\mathcal{S}(l))$ holds for all $l \in \mathtt{dom}(\mathcal{S})$. A **configuration**

type $\tau \times \mathcal{S}$ is **tight**, if τ and $\mathcal{S}$ are tight. A monadic type $\mathcal{S} \gg \kappa$ is **tight**, if κ is tight. The notion of tightness of type derivations is defined in the same way as in system $\mathcal{O}$, *i.e.* a **type derivation** Φ is **tight** if the type environment and the type of the conclusion of Φ are tight.

Example 4. Consider configuration c_0 from Ex. 2. Let $\delta_0 = \emptyset \gg (\mathtt{v} \times \emptyset)$, $\mathcal{M} = [[\mathtt{v}] \Rightarrow \delta_0]$, and Φ be the following typing derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{y : \mathcal{M} \vdash^{(0,0,0)} y : [\mathtt{v}] \Rightarrow \delta_0}\,(\mathtt{ax}) \qquad \dfrac{\dfrac{\dfrac{}{x : [\mathtt{v}] \vdash^{(0,0,0)} x : \mathtt{v}}\,(\mathtt{ax})}{x : [\mathtt{v}] \vdash^{(0,0,0)} x : [\mathtt{v}]}\,(\mathtt{m})}{x : [\mathtt{v}] \vdash^{(0,0,0)} x : \emptyset \gg ([\mathtt{v}] \times \emptyset)}\,(\uparrow)}{y : \mathcal{M}, x : [\mathtt{v}] \vdash^{(1,0,0)} yx : \delta_0}\,(\mathtt{@})}{x : [\mathtt{v}] \vdash^{(1,1,0)} \mathtt{get}_l(\lambda y.yx) : \{(l : \mathcal{M})\} \gg (\mathtt{v} \times \emptyset)}\,(\mathtt{get})}{\vdash^{(1,1,0)} \lambda x.\mathtt{get}_l(\lambda y.yx) : [\mathtt{v}] \Rightarrow (\{(l : \mathcal{M})\} \gg (\mathtt{v} \times \emptyset))}\,(\lambda)$$

And Φ' be the following typing derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : [\mathtt{v}] \vdash^{(0,0,0)} x : \delta_0}\,(\mathtt{ax_p})}{\vdash^{(0,0,0)} \mathtt{I} : [\mathtt{v}] \Rightarrow \delta_0}\,(\lambda)}{\vdash^{(0,0,0)} \mathtt{I} : \mathcal{M}}\,(\mathtt{m}) \qquad \dfrac{\dfrac{\dfrac{}{z : [\mathtt{v}] \vdash^{(0,0,0)} z : \mathtt{v}}\,(\mathtt{ax})}{z : [\mathtt{v}] \vdash^{(0,0,0)} z : [\mathtt{v}]}\,(\mathtt{m})}{z : [\mathtt{v}] \vdash^{(0,0,0)} z : \{(l : \mathcal{M})\} \gg ([\mathtt{v}] \times \{(l : \mathcal{M})\})}\,(\uparrow)}{z : [\mathtt{v}] \vdash^{(0,1,0)} \mathtt{set}_l(\mathtt{I}, z) : \emptyset \gg ([\mathtt{v}] \times \{(l : \mathcal{M})\})}\,(\mathtt{set})$$

Then we can build the following tight typing derivation Φ_c for c:

$$\dfrac{\dfrac{\Phi \qquad \Phi'}{z : [\mathtt{v}] \vdash^{(1,2,0)} (\lambda x.\mathtt{get}_l(\lambda y.yx))(\mathtt{set}_l(\mathtt{I}, z)) : \delta_0}\,(\mathtt{@}) \qquad \dfrac{}{z : [\mathtt{v}] \vdash^{(0,0,0)} \epsilon : \emptyset}\,(\mathtt{emp})}{z : [\mathtt{v}] \vdash^{(1,2,0)} ((\lambda x.\mathtt{get}_l(\lambda y.yx))(\mathtt{set}_l(\mathtt{I}, z)), \epsilon) : \mathtt{v} \times \emptyset}\,(\mathtt{conf})$$

We will come back to this example at the end of Sec. 3.3.

3.3 Soundness and Completeness

In this section we show the main properties of the type system $\mathcal{P}$ with respect to the operational semantics of the λ-calculus with global state. The properties of type system $\mathcal{P}$ are similar to the ones for $\mathcal{O}$, but now with respect to configurations instead of terms. *Soundness* does not only state that a (tightly) typable configuration (t, s) is terminating, but also gives exact (and split) measures concerning the reduction sequence from (t, s) to a final form. *Completeness* guarantees that a terminating configuration (t, s) is tightly typable, where the measures of the associated reduction sequence of (t, s) to final form are reflected in the counters of the resulting type derivation of (t, s). This is the first work providing a model for a language with global memory being able to count the number of memory accesses.

We start by noting that type system $\mathcal{P}$ does not type blocked configurations, which is exactly the notion that we want to capture.

Rules for Terms

$$\frac{}{x : [\sigma] \vdash^{(0,0,0)} x : \sigma}\ (\mathtt{ax}) \qquad \frac{\Gamma \vdash^{(b,m,d)} v : \mathcal{M}}{\Gamma \vdash^{(b,m,d)} v : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S})}\ (\uparrow)$$

$$\frac{\Gamma \vdash^{(b,m,d)} t : \mathcal{S} \gg \kappa}{\Gamma \setminus\!\!\setminus x \vdash^{(b,m,d)} \lambda x.t : \Gamma(x) \Rightarrow (\mathcal{S} \gg \kappa)}\ (\lambda) \qquad \frac{(\Gamma_i \vdash^{(b_i,m_i,d_i)} v : \sigma_i)_{i\in I}}{+_{i\in I}\Gamma_i \vdash^{(+_{i\in I}b_i, +_{i\in I}m_i, +_{i\in I}d_i)} v : [\sigma_i]_{i\in I}}\ (\mathtt{m})$$

$$\frac{\Gamma \vdash^{(b,m,d)} v : \mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa) \qquad \Delta \vdash^{(b',m',d')} t : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}')}{\Gamma + \Delta \vdash^{(1+b+b', m+m', d+d')} vt : \mathcal{S} \gg \kappa}\ (\mathtt{@})$$

$$\frac{\Gamma \vdash^{(b,m,d)} t : \mathcal{S} \gg \kappa}{\Gamma \setminus\!\!\setminus x \vdash^{(b,1+m,d)} \mathtt{get}_l(\lambda x.t) : \{(l : \Gamma(x))\} \uplus \mathcal{S} \gg \kappa}\ (\mathtt{get})$$

$$\frac{\Gamma \vdash^{(b,m,d)} v : \mathcal{M} \qquad \Delta \vdash^{(b',m',d')} t : \{(l : \mathcal{M})\}; \mathcal{S} \gg \kappa}{\Gamma + \Delta \vdash^{(b+b', 1+m+m', d+d')} \mathtt{set}_l(v,t) : \mathcal{S} \gg \kappa}\ (\mathtt{set})$$

$$\frac{}{x : [\overline{\mathtt{n}}] \vdash^{(0,0,0)} x : \mathcal{S} \gg (\overline{\mathtt{n}} \times \mathcal{S})}\ (\mathtt{ax}_{\mathtt{p}}) \qquad \frac{}{\vdash^{(0,0,0)} \lambda x.t : \mathcal{S} \gg (\mathtt{a} \times \mathcal{S})}\ (\lambda_{\mathtt{p}})$$

$$\frac{\Gamma \vdash^{(b,m,d)} t : \mathcal{S} \gg (\mathtt{tt} \times \mathcal{S}')}{(x : [\mathtt{v}]) + \Gamma \vdash^{(b,m,1+d)} xt : \mathcal{S} \gg (\mathtt{n} \times \mathcal{S}')}\ (\mathtt{@}_{\mathtt{p1}}) \qquad \frac{\Gamma \vdash^{(b,m,d)} u : \mathcal{S} \gg (\mathtt{n} \times \mathcal{S}')}{\Gamma \vdash^{(b,m,1+d)} (\lambda x.t)u : \mathcal{S} \gg (\mathtt{n} \times \mathcal{S}')}\ (\mathtt{@}_{\mathtt{p2}})$$

Rules for States

$$\frac{}{\vdash^{(0,0,0)} \epsilon : \emptyset}\ (\mathtt{emp}) \qquad \frac{\Gamma \vdash^{(b,m,d)} v : \mathcal{M} \qquad \Delta \vdash^{(b',m',d')} s : \mathcal{S}}{\Gamma + \Delta \vdash^{(b+b', m+m', d+d')} \mathtt{upd}_l(v,s) : \{(l : \mathcal{M})\}; \mathcal{S}}\ (\mathtt{upd})$$

Rule for Configurations

$$\frac{\Gamma \vdash^{(b,m,d)} t : \mathcal{S} \gg \kappa \qquad \Delta \vdash^{(b',m',d')} s : \mathcal{S}}{\Gamma + \Delta \vdash^{(b+b', m+m', d+d')} (t,s) : \kappa}\ (\mathtt{conf})$$

Fig. 2. Typing rules for $\lambda_{\mathtt{gs}}$.

Proposition 3. *If $\Phi \rhd \Gamma \vdash^{(b,m,d)} (t,s) : \kappa$, then (t,s) is unblocked.*

We also show that counters capture the notion of normal form correctly, both for terms and states.

Lemma 1.

1. *Let $\Phi \rhd \Gamma \vdash^{(0,0,d)} t : \delta$ be tight. Then, (1) $t \in \mathtt{no}$ and (2) $d = |t|$.*
2. *Let $\Phi \rhd \Delta \vdash^{(0,0,d)} s : \mathcal{S}$ be tight. Then $d = 0$.*

In fact, we can show the following stronger property with respect to the counters for the number of β_v- and $\mathtt{g}/\mathtt{s}$-steps.

Lemma 2. *Let $\Phi \rhd \Gamma \vdash^{(b,m,d)} t : \delta$ be tight. Then, $b = m = 0$ iff $t \in \mathtt{no}$.*

The following property is essential for tight type systems [2], and it shows that tightness of types spreads throughout type derivations of neutral terms, just as long as the environments are tight.

Lemma 3 (Tight Spreading). *Let $\Phi \triangleright \Gamma \vdash^{(b,m,d)} t : \mathcal{S} \gg (\tau \times \mathcal{S}')$, such that Γ is tight. If $t \in \mathtt{ne}$, then $\tau \in \mathtt{tt}$.*

The two following properties ensure tight typability of final configurations. For that we need to be able to *tightly* type any state, as well as any normal form. In fact, normal forms do not depend on a particular state since they are irreducible, so we can type them with any state type.

Lemma 4 (Typability of States and Normal Forms).

1. *Let s be a state. Then, there exists $\Phi \triangleright \vdash^{(0,0,0)} s : \mathcal{S}$ tight.*
2. *Let $t \in \mathtt{no}$. Then for any tight $\mathcal{S}$ there exists $\Phi \triangleright \Gamma \vdash^{(0,0,d)} t : \mathcal{S} \gg (\mathtt{tt} \times \mathcal{S})$ tight s.t. $d = |t|$.*

Finally, we state the usual basic properties.

Lemma 5 (Substitution and Anti-Substitution).

1. **(Substitution)** *If $\Phi_t \triangleright \Gamma_t; x : \mathcal{M} \vdash^{(b_t,m_t,d_t)} t : \delta$ and $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M}$, then $\Phi_{t\{x\backslash v\}} \triangleright \Gamma_t + \Gamma_v \vdash^{(b_t+b_v,m_t+m_v,d_t+d_v)} t\{x\backslash v\} : \delta$.*
2. **(Anti-Substitution)** *If $\Phi_{t\{x\backslash v\}} \triangleright \Gamma_{t\{x\backslash v\}} \vdash^{(b,m,d)} t\{x\backslash v\} : \delta$, then $\Phi_t \triangleright \Gamma_t; x : \mathcal{M} \vdash^{(b_t,m_t,d_t)} t : \delta$ and $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M}$, such that $\Gamma_{t\{x\backslash v\}} = \Gamma_t + \Gamma_v$, $b = b_t + b_v$, $m = m_t + m_v$, and $d = d_t + d_v$.*

Lemma 6 (Split Exact Subject Reduction and Expansion).

1. **(Subject Reduction)** *Let $(t,s) \to_{\mathtt{r}} (u,q)$. If $\Phi \triangleright \Gamma \vdash^{(b,m,d)} (t,s) : \kappa$ is tight, then $\Phi' \triangleright \Gamma \vdash^{(b',m',d)} (u,q) : \kappa$, where $\mathtt{r} = \beta$ implies $b' = b - 1$ and $m' = m$, while $\mathtt{r} \in \{\mathtt{g},\mathtt{s}\}$ implies $b' = b$ and $m' = m - 1$.*
2. **(Subject Expansion)** *Let $(t,s) \to_{\mathtt{r}} (u,q)$. If $\Phi' \triangleright \Gamma \vdash^{(b',m',d)} (u,q) : \kappa$ is tight, then $\Phi \triangleright \Gamma \vdash^{(b,m,d)} (t,s) : \kappa$, where $\mathtt{r} = \beta$ implies $b' = b - 1$ and $m' = m$, while $\mathtt{r} \in \{\mathtt{g},\mathtt{s}\}$ implies $b' = b$ and $m' = m - 1$.*

Soundness (resp. completeness) is based on exact subject reduction (resp. expansion) respectively, in turn based on the previous substitution (resp. anti-substitution) lemma.

Theorem 1 (Quantitative Soundness and Completeness).

1. **(Soundness)** *If $\Phi \triangleright \Gamma \vdash^{(b,m,d)} (t,s) : \kappa$ tight, then there exists (u,q) such that $u \in \mathtt{no}$ and $(t,s) \twoheadrightarrow^{(b,m)} (u,q)$ with b β-steps, m $\mathtt{g}/\mathtt{s}$-steps, and $|(u,q)| = d$.*
2. **(Completeness)** *If $(t,s) \twoheadrightarrow^{(b,m,d)} (u,q)$ and $u \in \mathtt{no}$, then there exists $\Phi \triangleright \Gamma \vdash^{(b,m,|(u,q)|)} (t,s) : \kappa$ tight.*

Example 5. Consider again configuration c_0 from Ex. 2 and its associated tight derivation Φ_{c_0}. The first two counters of Φ_c are different from 0: this means that c is not a final configuration, but normalizes in one β_v-step ($b = 1$) and two $\mathtt{g}/\mathtt{s}$-steps ($m = 2$), to a final configuration having size $d = 0 = |z| = |(z, \mathtt{upd}_l(I, \epsilon))|$.

4 Conclusion and Related Work

This paper provides a foundational step into the development of quantitative models for programming languages with effects. We focus on a simple language with global memory access capabilities. Due to the inherent lack of confluence in such framework we fix a particular evaluation strategy following a (weak) CBV approach. We provide a type system for our language that is able to (both) extract and discriminate between (exact) measures for the length of evaluation, number of memory accesses and size of normal forms. This study provides a valuable insight into time and space analysis of languages with global memory.

In future work we would like to explore effectful computations involving global memory in a more general framework being able to capture different models of computation, such as the CBPV [25] or the bang calculus [7]. Furthermore, we would like to apply our quantitative techniques to other effects that can be found in programming languages, such as non-termination, exceptions, non-determinism, I/O.

Related Work. Several papers proposed quantitative approaches for different notions of CBV (without effects). But none of them exploits the idea of exact *and* split tight typing. Indeed, the first non-idempotent intersection type system for Plotkin's CBV is [16], where reduction is allowed under abstractions, and terms are considered to be closed. This work was further extended to [9], where commutation rules are added to the calculus. None of these contributions extracts quantitative bounds from the type derivations. A calculus for open CBV is proposed in [3], where *fireball* –normal forms– can be either erased or duplicated. Quantitative results are obtained, but no split measures. Other similar approaches appear in [20]. A logical characterization of CBV solvability is given in [4], the resulting non-impotent system gives quantitative information of the *solvable* associated reduction relation. A similar notion of solvability for CBV for generalized applications was studied in [23], together with a logical characterization provided by a quantitative system. Other non-idempotent systems for CBV were proposed [26,22], but they are defective in the sense that they do not enjoy subject reduction and expansion. Split measures for (strong) open CBV are developed in [24].

In [15], a system with universally quantified intersection and reference types is introduced for a language belonging to the ML-family. However, intersections are idempotent and only (qualitative) soundness is proved.

Concerning (exact) quantitative models for programming languages with global state the state of the art is still underexplored. Some sound but not complete approaches are given in [6,13], and quantitative results are not provided. Our work is inspired by a recent idempotent (thus only qualitative and not quantitative) model for CBV with global memory proposed by [14]. This work was further extended in [18] to a more generic framework of algebraic effectful computation, still the model does not provide any quantitative information about the evaluation of programs and the size of their results.

References

1. Accattoli, B.: Proof nets and the call-by-value λ-calculus. Theor. Comput. Sci. **606**, 2–24 (2015). https://doi.org/10.1016/j.tcs.2015.08.006, https://doi.org/10.1016/j.tcs.2015.08.006
2. Accattoli, B., Graham-Lengrand, S., Kesner, D.: Tight typings and split bounds, fully developed. J. Funct. Program. **30**(e14), 1–101 (2020). https://doi.org/10.1017/S095679682000012X, https://doi.org/10.1017/S095679682000012X
3. Accattoli, B., Guerrieri, G.: Types of fireballs. In: 16th Asian Symposium Programming Languages and Systems (APLAS), 2018, Wellington, New Zealand. Lecture Notes in Computer Science, vol. 11275, pp. 45–66. Springer (2018). https://doi.org/10.1007/978-3-030-02768-1_3, https://doi.org/10.1007/978-3-030-02768-1_3
4. Accattoli, B., Guerrieri, G.: The theory of call-by-value solvability. Proc. ACM Program. Lang. **6**(ICFP), 855–885 (2022). https://doi.org/10.1145/3547652, https://doi.org/10.1145/3547652
5. Alves, S., Kesner, D., Ramos, M.: Quantitative global memory (full version) (2023), https://boilnkettle.github.io/assets/papers/quantitative-global-memory-full.pdf
6. Benton, N., Kennedy, A., Beringer, L., Hofmann, M.: Relational semantics for effect-based program transformations: higher-order store. In: 11th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, (POPL), 2009, Coimbra, Portugal. pp. 301–312. ACM (2009). https://doi.org/10.1145/1599410.1599447, https://doi.org/10.1145/1599410.1599447
7. Bucciarelli, A., Kesner, D., Ríos, A., Viso, A.: The bang calculus revisited. In: Functional and Logic Programming - 15th International Symposium, FLOPS 2020, Akita, Japan, September 14-16, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12073, pp. 13–32. Springer (2020). https://doi.org/10.1007/978-3-030-59025-3_2, https://doi.org/10.1007/978-3-030-59025-3_2
8. Bucciarelli, A., Kesner, D., Ventura, D.: Non-idempotent intersection types for the lambda-calculus. Log. J. (IGPL) **25**(4), 431–464 (2017). https://doi.org/10.1093/jigpal/jzx018, https://doi.org/10.1093/jigpal/jzx018
9. Carraro, A., Guerrieri, G.: A semantical and operational account of call-by-value solvability. In: 17th International Conference on Foundations of Software Science and Computation Structures, (FOSSACS), 2014, Grenoble, France. Lecture Notes in Computer Science, vol. 8412, pp. 103–118. Springer (2014). https://doi.org/10.1007/978-3-642-54830-7_7, https://doi.org/10.1007/978-3-642-54830-7_7
10. de Carvalho, D.: Sémantiques de la logique linéaire et temps de calcul. These de doctorat, Université Aix-Marseille II (2007)
11. de Carvalho, D.: Execution time of lambda-terms via denotational semantics and intersection types. CoRR **abs/0905.4251** (2009), http://arxiv.org/abs/0905.4251
12. Coppo, M., Dezani-Ciancaglini, M.: A new type assignment for lambda-terms. Archiv für Math. Logik **19**, 139–156 (1978)
13. Davies, R., Pfenning, F.: Intersection types and computational effects. In: Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP), 2000, Montreal, Canada. pp. 198–208. ACM (2000). https://doi.org/10.1145/351240.351259, https://doi.org/10.1145/351240.351259

14. de'Liguoro, U., Treglia, R.: Intersection types for a λ-calculus with global store. In: PPDP 2021: 23rd International Symposium on Principles and Practice of Declarative Programming, (POPL), 2021, Tallinn, Estonia. pp. 5:1–5:11. ACM (2021). https://doi.org/10.1145/3479394.3479400, https://doi.org/10.1145/3479394.3479400
15. Dezani-Ciancaglini, M., Giannini, P., Rocca, S.R.D.: Intersection, universally quantified, and reference types. In: 18th Annual Conference of the EACSL on Computer Science Logic, 23rd international Workshop, (CSL), 2009, Coimbra, Portugal. Proceedings. Lecture Notes in Computer Science, vol. 5771, pp. 209–224. Springer (2009). https://doi.org/10.1007/978-3-642-04027-6_17, https://doi.org/10.1007/978-3-642-04027-6_17
16. Ehrhard, T.: Collapsing non-idempotent intersection types. In: 26th International Workshop/21st Annual Conference of the EACSL on Computer Science Logic, (CSL), 2012, Fontainebleau, France. LIPIcs, vol. 16, pp. 259–273. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2012). https://doi.org/10.4230/LIPIcs.CSL.2012.259, https://doi.org/10.4230/LIPIcs.CSL.2012.259
17. Faggian, C., Guerrieri, G., de'Liguoro, U., Treglia, R.: On reduction and normalization in the computational core. CoRR **abs/2104.10267** (2021), https://arxiv.org/abs/2104.10267
18. Francesco Gavazzo, Gabriele Vanoni, R.T.: On monadic intersection types (2023), submitted
19. Girard, J.: Linear logic. Theor. Comput. Sci. **50**, 1–102 (1987). https://doi.org/10.1016/0304-3975(87)90045-4, https://doi.org/10.1016/0304-3975(87)90045-4
20. Guerrieri, G.: Towards a semantic measure of the execution time in call-by-value lambda-calculus. In: 12th Workshop on Developments in Computational Models and Ninth Workshop on Intersection Types and Related Systems, (DCM/ITRS), 2018, Oxford, UK. EPTCS, vol. 293, pp. 57–72 (2018). https://doi.org/10.4204/EPTCS.293.5, https://doi.org/10.4204/EPTCS.293.5
21. Jones, S.L.P., Wadler, P.: Imperative functional programming. In: Conference Record of the Twentieth Annual (ACM) (SIGPLAN-SIGACT) Symposium on Principles of Programming Languages (POPL), 1993, Charleston, South Carolina, USA. pp. 71–84. ACM Press (1993). https://doi.org/10.1145/158511.158524, https://doi.org/10.1145/158511.158524
22. Kerinec, A., Manzonetto, G., Rocca, S.R.D.: Call-by-value, again! In: 6th International Conference on Formal Structures for Computation and Deduction, (FSCD), 2021, Buenos Aires, Argentina (Virtual Conference). LIPIcs, vol. 195, pp. 7:1–7:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.FSCD.2021.7, https://doi.org/10.4230/LIPIcs.FSCD.2021.7
23. Kesner, D., Peyrot, L.: Solvability for generalized applications. In: 7th International Conference on Formal Structures for Computation and Deduction, (FSCD), 2022, Haifa, Israel. LIPIcs, vol. 228, pp. 18:1–18:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.FSCD.2022.18, https://doi.org/10.4230/LIPIcs.FSCD.2022.18
24. Kesner, D., Viso, A.: Encoding tight typing in a unified framework. In: 30th EACSL Annual Conference on Computer Science Logic, (CSL), 2022, Göttingen, Germany (Virtual Conference). LIPIcs, vol. 216, pp. 27:1–27:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.CSL.2022.27, https://doi.org/10.4230/LIPIcs.CSL.2022.27

25. Levy, P.B.: Call-by-push-value: A subsuming paradigm. In: Typed Lambda Calculi and Applications, 4th International Conference, TLCA'99, L'Aquila, Italy, April 7-9, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1581, pp. 228–242. Springer (1999). `https://doi.org/10.1007/3-540-48959-2\_17`, `https://doi.org/10.1007/3-540-48959-2_17`
26. Manzonetto, G., Pagani, M., Rocca, S.R.D.: New semantical insights into call-by-value λ-calculus. Fundamenta Informaticae **170**(1-3), 241–265 (2019). `https://doi.org/10.3233/fi-2019-1862`, `https://doi.org/10.3233/fi-2019-1862`
27. Moggi, E.: Computational lambda-calculus and monads. In: 4th Annual Symposium on Logic in Computer Science, (LICS), 1989, Pacific Grove, California, USA. pp. 14–23. IEEE Computer Society (1989). `https://doi.org/10.1109/LICS.1989.39155`, `https://doi.org/10.1109/LICS.1989.39155`
28. Plotkin, G.D.: Call-by-name, call-by-value and the lambda-calculus. Theor. Comput. Sci. **1**(2), 125–159 (1975). `https://doi.org/10.1016/0304-3975(75)90017-1`, `https://doi.org/10.1016/0304-3975(75)90017-1`
29. Sabry, A., Wadler, P.: A reflection on call-by-value. ACM Trans. Program. Lang. Syst. **19**(6), 916–941 (1997). `https://doi.org/10.1145/267959.269968`, `https://doi.org/10.1145/267959.269968`
30. Treglia, R.: The computational core: reduction theory and intersection type discipline. Phd thesis, Univ. Torino (2022)
31. Wadler, P.: Monads for functional programming. In: 1st International Spring School on Advanced Functional Programming Techniques on Advanced Functional Programming, (AFP), 1995, Båstad, Sweden, Tutorial Text. Lecture Notes in Computer Science, vol. 925, pp. 24–52. Springer (1995). `https://doi.org/10.1007/3-540-59451-5\_2`, `https://doi.org/10.1007/3-540-59451-5_2`

A Proofs

A.1 Weak Open CBV

General Lemmas

Proposition 4 (Diamond). *The relation $\to_{\beta_v}$ enjoys the diamond property: if $t \to_{\beta_v} t_i$ $(i = 1, 2)$ and $t_1 \neq t_2$, then there exists t_3 such that $t_i \to_{\beta_v} t_3$ $i = 1, 2$.*

Proposition 1. *Let t be a term. Then $t \in \mathtt{no}$ iff $t \not\to \mathtt{no}$.*

Proof. We are going to show this proposition by splitting the original statement into the two following ones:

1. $t \not\to$ and $\neg\mathtt{val}(t)$ iff $t \in \mathtt{ne}$.
2. $t \not\to$ iff $t \in \mathtt{no}$.

The proof now follows by simultaneous induction over both these statements:

⇒) By induction over t:
   1. Let $t \not\to$ and $\neg\mathtt{val}(t)$. We want to show that $t \in \mathtt{ne}$:
      - Case $t = x$ or $t = \lambda x.u$. Then $\neg\mathtt{val}(t)$ does not hold. Therefore, the statement holds vacuously.

- Case $t = up$. Since $up \not\to$, then, in particular, it must be the case that either $\neg\texttt{abs}(u)$ or $\neg\texttt{val}(p)$ must hold, according to rule $(\beta_{\texttt{v}})$:
  - $*$ Assume $\neg\texttt{abs}(u)$ holds. It must be the case that $u \not\to$, according to rule $(\texttt{appL})$. And it also must be the case that $p \not\to$, according to rule $(\texttt{appR})$. Therefore, $p \in \texttt{no}$, by the *i.h.* (Prop. 1.2). Now, we have to consider u, which can be a variable, or not:
    - $\cdot$ Case $u = x$. Then $up \in x\ \texttt{no} \in \texttt{ne}$.
    - $\cdot$ Case u is not a variable. Then $\neg\texttt{val}(u)$ holds. Therefore, we have $u \in \texttt{ne}$, by the *i.h.* (Prop. 1.1). Thus, $up \in \texttt{ne}\ \texttt{no} \in \texttt{ne}$.
  - $*$ Assume $\neg\texttt{val}(p)$ holds. Then it must be the case that $u \not\to$, according to rule $(\texttt{appL})$. And that $p \not\to$, according to rule $(\texttt{appR})$. Therefore, $u \in \texttt{no}$, by the *i.h.* (1.2), and $p \in \texttt{ne}$, by the *i.h.* (Prop. 1.1). Thus, $up \in \texttt{no}\ \texttt{ne} \in \texttt{ne}$.

2. Let $t \not\to$. We want to show that $t \in \texttt{no}$:
   - Case $t \in \texttt{Val}$. Then, clearly $t \in \texttt{no}$.
   - Case $t \notin \texttt{Val}$. Then, $\neg\texttt{val}(t)$ holds. Therefore, $t \in \texttt{ne}$, by Prop. 1.1. Thus, in particular, $t \in \texttt{no}$.

$\Leftarrow$) By induction over $t \in \texttt{no}$:

1. Let $t \in \texttt{ne}$. We want to show that $t \not\to$ and $\neg\texttt{val}(t)$:
   - Case $t = up \in x\ \texttt{no}$. Then $u = x$ and $p \in \texttt{no}$. Since $u = x$, then both rules $(\beta_{\texttt{v}})$ and $(\texttt{appL})$ cannot be applied. Since $p \in \texttt{no}$, then $p \not\to$, by the *i.h.* (Prop. 1.2). Therefore, rule $(\texttt{appR})$ also cannot be applied. Thus, $up \not\to$. And we can conclude, since $\neg\texttt{val}(up)$ clearly holds.
   - Case $t = up \in \texttt{no}\ \texttt{ne}$. Then $u \in \texttt{no}$ and $p \in \texttt{ne}$. Since $u \in \texttt{no}$, then $u \not\to$, by the *i.h.* (Prop. 1.2). Since $p \in \texttt{ne}$, then $p \not\to$ and $\neg\texttt{val}(p)$ holds, by the *i.h.* (Prop. 1.1). Since $\neg\texttt{val}(p)$, then rule $(\beta_{\texttt{v}})$ cannot be applied. Since $u \not\to$ and $p \not\to$, then rules $(\texttt{appL})$ and $(\texttt{appR})$ cannot be applied. Therefore, $up \not\to$. And we can conclude since $\neg\texttt{val}(up)$ clearly holds.
   - Case $t = up \in \texttt{ne}\ \texttt{no}$. Then $u \in \texttt{ne}$ and $p \in \texttt{ne}$. Since $u \in \texttt{ne}$, then $u \not\to$ and $\neg\texttt{val}(u)$ holds, by the *i.h.* (Prop. 1.1). Since $p \in \texttt{no}$, then $p \not\to$, by the *i.h.* (Prop. 1.2). Since $\neg\texttt{val}(u)$, then rule $(\beta_{\texttt{v}})$ cannot be applied. Since $u \not\to$ and $p \not\to$, then rules $(\texttt{appL})$ and $(\texttt{appR})$ cannot be applied. Therefore $up \not\to$. And we can conclude since $\neg\texttt{val}(up)$ clearly holds.
2. Let $t \in \texttt{no}$. We want to show that $t \not\to$:
   - Case $t \in \texttt{Val}$. Then, clearly $t \not\to$.
   - Case $t \notin \texttt{Val}$. Then, $t \in \texttt{ne}$, by definition. Thus, $t \not\to$ holds, by Prop. 1.1.

Lemma 7 (Relevance). *Let* $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \tau$. *Then* $\texttt{dom}(\Gamma) \subseteq \texttt{fv}(t)$.

Proof. The proof following by induction over Φ. Case Φ ends with rule $(\texttt{ax})$ or $(\lambda_{\texttt{p}})$, then Φ is clearly relevant. The other cases following easily from the *i.h.*.

Soundness (Auxiliary Lemmas)

Lemma 8. *Let $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \tau$. If $t \in \mathtt{Val}$, then $\tau \neq \mathtt{n}$.*

Proof. By case analysis on the form of $t \in \mathtt{Val}$:

- Case $t = x$. Then we have to consider two additional cases according to the last rule used in Φ:
  - Case Φ ends with rule $(\mathtt{ax})$, then τ is of the form $\sigma \neq \mathtt{n}$.
  - Case Φ ends with rule $(\mathtt{m})$, then τ is of the form $\mathcal{M} \neq \mathtt{n}$.
- Case $t = \lambda x.t$. Then we have to consider three additional cases according to the last rule used in Φ:
  - Case Φ ends with rule (λ), then τ is of the form $\mathcal{M} \Rightarrow \delta \neq \mathtt{n}$.
  - Case Φ ends with rule $(\mathtt{m})$, then τ is of the form $\mathcal{M} \neq \mathtt{n}$.
  - Case Φ ends with rule $(\lambda_{\mathtt{p}})$, then $\tau = \mathtt{a} \neq \mathtt{n}$.

Lemma 9. *If $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \tau$, such that Γ is tight. If $\tau \in \overline{\mathtt{a}}$, then $\neg\mathtt{abs}(t)$.*

Proof. By induction over Φ:

- Case Φ ends with rule $(\mathtt{ax})$, $(\mathtt{@})$, $(\mathtt{@_{p1}})$, or $(\mathtt{@_{p2}})$, $\neg\mathtt{abs}(t)$ holds by definition.
- Case Φ ends with rule (λ), $(\mathtt{m})$, or $(\lambda_{\mathtt{p}})$, $\tau \notin \overline{\mathtt{a}}$. Therefore, these cases do not apply.

Lemma 10 (Zero Steps and Normal Forms). *Let $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \tau$ be tight. $b = 0$ iff $t \in \mathtt{no}$.*

Proof.

$\Rightarrow$) We want to show that, if $b = 0$, then $t \in \mathtt{no}$. For this, we are going to split the original statement into the two following ones:
1. Let $\Phi \triangleright \Gamma \vdash^{(0,s)} t : \tau$ be tight and $\neg\mathtt{val}(t)$, then $t \in \mathtt{ne}$.
2. Let $\Phi \triangleright \Gamma \vdash^{(0,s)} t : \tau$ be tight, then $t \in \mathtt{no}$.

The proof now follows by simultaneous induction over both these statements:
1. Let $\Phi \triangleright \Gamma \vdash^{(0,s)} t : \tau$ be tight and $\neg\mathtt{val}(t)$:
   - Case Φ ends with rule $(\mathtt{ax})$, (λ), $(\mathtt{m})$, or $(\lambda_{\mathtt{p}})$, then $\mathtt{val}(t)$ holds. Therefore, these cases do not apply.
   - Case Φ ends with rule $(\mathtt{@})$, then $b > 0$. Therefore, this case does not apply.
   - Case Φ ends with rule $(\mathtt{@_{p1}})$, then t is of the form up and Φ is of the following form:

$$\frac{\Phi_u \triangleright \Gamma_u \vdash^{(0,s_u)} u : \overline{\mathtt{a}} \qquad \Phi_p \triangleright \Gamma_p \vdash^{(0,s_p)} p : \mathtt{tt}}{\Gamma_u + \Gamma_p \vdash^{(0,1+s_u+s_p)} up : \mathtt{n}} \ (\mathtt{@_{p1}})$$

where $\tau = \mathtt{n}$, $\Gamma = \Gamma_u + \Gamma_p$ is tight, and $s = 1 + s_u + s_p$. Moreover, Γ_u and Γ_p are tight. By the *i.h.* (Lemma 10.2) over Φ_u and Φ_p, we have that $u, p \in \mathtt{no}$. By Lemma 9, we have that $\neg\mathtt{abs}(u)$. Therefore, either u is a variable or $u \in \mathtt{ne}$ by definition. So, in both cases, we can conclude that $up \in \mathtt{ne}$.

- Case Φ ends with rule ($@_{\mathtt{p2}}$), then t is of the form up and Φ is of the following form:

$$\frac{\Phi_u \triangleright \Gamma_u \vdash^{(0,s_u)} u : \mathtt{tt} \qquad \Phi_p \triangleright \Gamma_p \vdash^{(0,s_p)} p : \mathtt{n}}{\Gamma_u + \Gamma_p \vdash^{(0,1+s_u+s_p)} up : \mathtt{n}} \; (@_{\mathtt{p2}})$$

  where $\tau = \mathtt{n}$, $\Gamma = \Gamma_u + \Gamma_p$, and $s = 1 + s_u + s_p$. Moreover, Γ_u and Γ_p are tight. By the *i.h.* (Lemma 10.2) over Φ_u, we have that $u \in \mathtt{no}$. By applying Lemma 8 to Φ_p, we have that $\neg\mathtt{val}(p)$. By the *i.h.* (Lemma 10.1) over Φ_p, we have that $p \in \mathtt{ne}$. So, in both cases, we can conclude that $up \in \mathtt{ne}$.

2. Let $\Phi \triangleright \Gamma \vdash^{(0,s)} t : \tau$ be tight:
   - Case Φ ends with rule $(\mathtt{ax})$, (λ), or $(\lambda_{\mathtt{p}})$. Then, clearly $t \in \mathtt{Val}$, so we can conclude immediately.
   - Case Φ ends with rule $(\mathtt{m})$, then τ is of the form $\mathcal{M} \notin \mathtt{tt}$. Therefore, this case does not apply.
   - In all the remaining cases $\neg\mathtt{val}(t)$ holds. Then $t \in \mathtt{ne}$, by Lemma 10.1, so $t \in \mathtt{no}$.

$\Leftarrow$) We want to show that, if $t \in \mathtt{no}$, then $b = 0$. The proof follows by induction over $t \in \mathtt{no}$:

1. Case $t \in \mathtt{ne}$. Then we have to consider the following additional cases:
   - Case $t = xp$, such that $p \in \mathtt{no}$. Then there are three additional cases to consider:
     * Case Φ ends with $(@)$, then it must be of the following form:

$$\frac{x : [\mathcal{M} \Rightarrow \tau] \vdash^{(0,0)} x : \mathcal{M} \Rightarrow \tau \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \mathcal{M}}{(x : [\mathcal{M} \Rightarrow \tau]) + \Gamma_p \vdash^{(1+b_p,s_p)} xp : \tau} \; (@)$$

       where $\Gamma = (x : [\mathcal{M} \Rightarrow \tau]) + \Gamma_p$ is tight, $b = 1 + b_p$, and $s = s_p$. But, $[\mathcal{M} \Rightarrow \tau]$ is not tight, since $\mathcal{M} \Rightarrow \tau \notin \mathtt{tt}$. Therefore, this case does apply.
     * Case Φ ends with $(@_{\mathtt{p1}})$, then Φ must be of the following form:

$$\frac{(x : [\mathtt{v}]) \vdash^{(0,0)} x : \mathtt{v} \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,b_p)} p : \mathtt{tt}}{\Gamma_u + \Gamma_p \vdash^{(b_p,1+s_u+s_p)} up : \mathtt{n}} \; (@_{\mathtt{p1}})$$

       where $\tau = \mathtt{n}$, $\Gamma = (x : [\mathtt{v}]) + \Gamma_p$ is tight, $b = b_p$, and $s = 1 + s_u + s_p$. Moreover, Γ_p is tight. By the *i.h.* over Φ_p, we have that $b_p = 0$. So we can conclude with $b = b_u + b_p = 0$.
     * Case Φ ends with $(@_{\mathtt{p2}})$. This case is very similar to the case where Φ ends with rule $(@_{\mathtt{p1}})$.
   - Case $t = up$, such that $u \in \mathtt{no}$ and $p \in \mathtt{ne}$. Then there are three additional cases to consider:
     * Case Φ ends with $(@)$, then it must be of the following form:

$$\frac{\Gamma_u \vdash^{(b_u,s_u)} u : \mathcal{M} \Rightarrow \tau \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \mathcal{M}}{\Gamma_u + \Gamma_p \vdash^{(1+b_u+b_p,s_u+s_p)} up : \tau} \; (@)$$

where $\tau = \tau$, $\Gamma = \Gamma_u + \Gamma_p$ is tight, $b = 1 + b_u + b_p$, and $s = s_u + s_p$. By Lemma 13.2, we have that $\mathcal{M} \in \mathtt{tt}$, which is a contradiction. Therefore, this case does not apply.

  ∗ Case Φ ends with $(\mathtt{@_{p1}})$ or $(\mathtt{@_{p2}})$. These cases are very similar to the corresponding cases when $t = xp$, such that $p \in \mathtt{no}$.

- Case $t = up$, such that $u \in \mathtt{ne}$ and $p \in \mathtt{no}$. Then there are three cases to consider:
  ∗ Case Φ ends with $(\mathtt{@})$, then it must be of the following form:

$$\frac{\Gamma_u \vdash^{(b_u, s_u)} u : \mathcal{M} \Rightarrow \tau \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p, s_p)} p : \mathcal{M}}{\Gamma_u + \Gamma_p \vdash^{(1 + b_u + b_p, s_u + s_p)} up : \tau} \; (\mathtt{@})$$

where $\tau = \tau$, $\Gamma = \Gamma_u + \Gamma_p$ is tight, $b = 1 + b_u + b_p$, and $s = s_u + s_p$. By Lemma 13.2 over $u \in \mathtt{ne}$, we have that $\mathcal{M} \Rightarrow \tau \in \mathtt{tt}$, which is a contradiction. Therefore, this case does not apply.

  ∗ Case Φ ends with $(\mathtt{@_{p1}})$ or $(\mathtt{@_{p2}})$. These cases are very similar to corresponding cases when $t = xp$, such that $p \in \mathtt{no}$, or $t = up$, such that $u \in \mathtt{no}$ and $p \in \mathtt{ne}$.

2. Case $t \in \mathtt{no}$. Then we can consider the two following additional cases:
   - Case $t \in \mathtt{Val}$. Then Φ must end with $(\mathtt{ax})$, (λ), $(\mathtt{m})$, or $(\lambda_{\mathtt{p}})$. With the exception of the case where Φ ends with rule $(\mathtt{m})$, we can conclude $b = 0$ immediately for every other case, by definition. Case Φ ends with rule $(\mathtt{m})$, then τ is of the form $\mathcal{M} \notin \mathtt{tt}$. Therefore, this case does not apply.
   - Case $t \notin \mathtt{Val}$. Then, $t \in \mathtt{ne}$, by definition. Therefore, $b = 0$, by Lemma 10.1.

Lemma 11. *Let $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \tau$ be tight. If $b = 0$ then $s = |t|$.*

Proof. The proof follows by induction over Φ:

– Case Φ ends with rule $(\mathtt{ax})$ or $(\lambda_{\mathtt{p}})$. Then $t \in \mathtt{Val}$ and $s = 0$. So we can conclude with $|t| = 0 = s$.
– Case Φ ends with rule (λ). Then τ is of the form $\Gamma_u(x) \Rightarrow \delta \notin \mathtt{tt}$, so this case does not apply.
– Case Φ ends with rule $(\mathtt{@})$. Then $b > 0$, so this case does not apply.
– Case Φ ends with rule $(\mathtt{m})$. Then τ is of the form $\mathcal{M} \notin \mathtt{tt}$, so this case does not apply.
– Case Φ ends with rule $(\mathtt{@_{p1}})$. Then $t = up$ and Φ must be of the following form:

$$\frac{\Phi_u \triangleright \Gamma_u \vdash^{(0, s_u)} u : \overline{\mathtt{a}} \qquad \Phi_p \triangleright \Gamma_p \vdash^{(0, s_p)} p : \mathtt{tt}}{\Gamma_u + \Gamma_p \vdash^{(0, 1 + s_u + s_p)} up : \mathtt{n}} \; (\mathtt{@_{p1}})$$

where $\tau = \mathtt{n}$, $\Gamma = \Gamma_u + \Gamma_p$, and $s = 1 + s_u + s_p$. Moreover, Γ_u and Γ_p are tight. By the *i.h.* over Φ_u and Φ_p, we have $s_u = |u|$ and $s_p = |p|$. So we can conclude with $s = 1 + |u| + |p| = |up|$.
– Case Φ ends with rule $(\mathtt{@_{p2}})$. This case is very similar to the case where Φ ends with rule $(\mathtt{@_{p1}})$.

Lemma 12 (Split for Values). *Let $\Phi_v \triangleright \Gamma \vdash^{(b,s)} v : \mathcal{M}$, such that $\mathcal{M} = \sqcup_{i\in I}\mathcal{M}_i$. Then, there exist $(\Phi_v^i \triangleright \Gamma_i \vdash^{(b_i,s_i)} v : \mathcal{M}_i)_{i\in I}$, such that $\Gamma = +_{i\in I}\Gamma_i$, $b = +_{i\in I}b_i$, and $s = +_{i\in I}s_i$.*

Proof. We start by noting that Φ_v must end with the rule ($\mathtt{m}$). Therefore, we have $\Gamma = +_{j\in J}\Gamma_j$, $\mathcal{M} = [\sigma_j]_{j\in J}$, $b = +_{j\in J}b_j$, $s = +_{j\in J}s_j$, and $(\Phi_v^j \triangleright \Gamma_j \vdash^{(b_j,s_j)} v : \sigma_j)_{j\in J}$, for some J. Let $\mathcal{M}_i = [\sigma_k]_{k\in K_i}$, for each $i \in I$, such that $J = +_{i\in I}K_i$. Then, by using rule ($\mathtt{m}$), we can build $\Phi_v^i \triangleright \Gamma_i \vdash^{(b_i,s_i)} v : \mathcal{M}_i$, for each $i \in I$, such that $\Gamma_i = +_{k\in K_i}\Gamma_k$, $b_i = +_{k\in K_i}b_k$, and $s_i = +_{k\in K_i}s_k$. So we can conclude with $\Gamma = +_{j\in J}\Gamma_j = +_{i\in I}(+_{k\in K_i}\Gamma_k) = +_{i\in I}\Gamma_i$, $b = +_{j\in J}b_j = +_{i\in I}(+_{k\in K_i}b_k) = +_{i\in I}b_i$, and $s = +_{j\in J}s_j = +_{i\in I}(+_{k\in K_i}s_k) = +_{i\in I}s_i$.

Completeness (Auxiliary Lemmas)

Lemma 13 (Tight Spreading). *Let $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \tau$, such that Γ is tight:*

1. *If $b = 0$ and τ is not an arrow type or a multi-type, then $\tau \in \mathtt{tt}$.*
2. *If $t \in \mathtt{ne}$, then $\tau \in \mathtt{tt}$.*

Proof.

1. We want to show that, if $b = 0$ and τ is not an arrow type or a multiset type, then $\tau \in \mathtt{tt}$. The proof follows by induction over Φ:
   - Case Φ ends with rule ($\mathtt{ax}$), then it is of the following form:

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

     such that $\tau = \sigma$, $\Gamma = x : [\sigma]$, and $s = 0$. If $x : [\sigma]$ is tight, then $\sigma \in \{\mathtt{a}, \mathtt{v}\}$. Therefore, we can conclude with $\sigma \in \{\mathtt{a}, \mathtt{v}\} \subset \mathtt{tt}$.
   - Case Φ ends with rule (λ), then τ is an arrow type. Therefore, this case does not apply.
   - Case Φ ends with rule ($\mathtt{@}$), then $b > 0$. Therefore, this case does not apply.
   - Case Φ ends with rule ($\mathtt{m}$), then τ is a multiset type. Therefore, this case does not apply.
   - Case Φ ends with rule ($\lambda_{\mathtt{p}}$), then $\tau = \mathtt{a} \in \mathtt{tt}$.
   - Case Φ ends with rules ($\mathtt{@}_{\mathtt{p1}}$) or ($\mathtt{@}_{\mathtt{p2}}$), then $\tau = \mathtt{n} \in \mathtt{tt}$.
2. We want to show that, if $t \in \mathtt{ne}$, then $\tau \in \mathtt{tt}$. By induction over $t \in \mathtt{ne}$:
   - Case $t = xp$, such that $p \in \mathtt{no}$. Then we have to consider the following three cases depending on the last rule in Φ:
     - Case Φ ends with rule ($\mathtt{@}$), then it must be of the following form:

       $$\frac{x : [\mathcal{M} \Rightarrow \tau] \vdash^{(0,0)} x : \mathcal{M} \Rightarrow \delta \qquad \Phi_p \triangleright \Gamma_p \vdash^{(1+b_p,s_p)} p : \mathcal{M}}{(x : [\mathcal{M} \Rightarrow \tau]) + \Gamma_p \vdash^{(b_p,s_p)} xp : \delta}\ (\mathtt{@})$$

       where $\Gamma = (x : [\mathcal{M} \Rightarrow \delta]) + \Gamma_p$ is tight, $b = 1 + b_p$, and $s = s_p$. But, $[\mathcal{M} \Rightarrow \delta]$ is not tight, since $\mathcal{M} \Rightarrow \delta \notin \mathtt{tt}$. Therefore, this case does apply.

  - Case Φ ends with rule $(\mathtt{@_{p1}})$ or $(\mathtt{@_{p2}})$. Then $\tau = \mathtt{n} \in \mathtt{tt}$, so we can conclude immediately.
- Case $t = up$, such that $u \in \mathtt{no}$ and $p \in \mathtt{ne}$. Then we have to consider the following three cases depending on the last rule in Φ:
  - Case Φ ends with rule $(\mathtt{@})$, then it must be of the following form:

$$\dfrac{\Phi_u \triangleright \Gamma_u \vdash^{(b_u,s_u)} u : \mathcal{M} \Rightarrow \tau \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \mathcal{M}}{\Gamma_u + \Gamma_p \vdash^{(1+b_u+b_p,s_u+s_p)} up : \tau}\ (\mathtt{@})$$

    where $\Gamma = \Gamma_u + \Gamma_p$ is tight, $b = 1+b_u+b_p$, and $s = s_u+s_p$. Moreover, Γ_p is tight. By the *i.h.* over Φ_p, we have that $\mathcal{M} \in \mathtt{tt}$, which is a contradiction. Therefore, this case does not apply.
  - Case Φ ends with rule $(\mathtt{@_{p1}})$ or $(\mathtt{@_{p2}})$. Then $\tau = \mathtt{n} \in \mathtt{tt}$, so we can conclude immediately.
- Case $t = up$, such that $u \in \mathtt{ne}$ and $p \in \mathtt{no}$. Then we have to consider the following three cases depending on the last rule in Φ:
  - Case Φ ends with rule $(\mathtt{@})$, then it must be of the following form:

$$\dfrac{\Phi_u \triangleright \Gamma_u \vdash^{(b_u,s_u)} u : \mathcal{M} \Rightarrow \tau \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \mathcal{M}}{\Gamma_u + \Gamma_p \vdash^{(1+b_u+b_p,s_u+s_p)} up : \tau}\ (\mathtt{@})$$

    where $\Gamma = \Gamma_u + \Gamma_p$ is tight, $b = 1+b_u+b_p$, and $s = s_u+s_p$. Moreover, Γ_p is tight. By the *i.h.* over Φ_p, we have that $\mathcal{M} \in \mathtt{tt}$, which is a contradiction. Therefore, this case does not apply.
  - Case Φ ends with rule $(\mathtt{@_{p1}})$ or $(\mathtt{@_{p2}})$. Then $\tau = \mathtt{n} \in \mathtt{tt}$, so we can conclude immediately.

Lemma 14 (Typability of Normal Forms). *If $t \in \mathtt{no}$, then there exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \tau$, such that $s = |t|$.*

To show this proposition we are going to need to split the original statement into the two following ones:

1. If $t \in \mathtt{ne}$, then there exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \mathtt{n}$, such that $s = |t|$.
2. If $t \in \mathtt{no}$, then there exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \mathtt{tt}$, such that $s = |t|$.

The proof follows by simultaneous induction over both these statements:

1. Let $t \in \mathtt{ne}$. We want to show that there exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \mathtt{n}$:
   - Case $t = up \in x\ \mathtt{no}$. Then $u = x$ and $p \in \mathtt{no}$. Therefore, there exists a tight derivation $\Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \mathtt{tt}$, by the *i.h.* (Lemma 14.2), such that $|p| = s_p$. Thus, we can build Φ as follows:

$$\dfrac{\dfrac{}{x : [\mathtt{v}] \vdash^{(0,0)} x : \mathtt{v}}\ (\mathtt{ax}) \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \mathtt{tt}}{x : [\mathtt{v}] + \Gamma_p \vdash^{(b_p,1+s_p)} xp : \mathtt{n}}\ (\mathtt{@_{p1}})$$

     And we can conclude with $\Gamma = x : [\mathtt{v}] + \Gamma_p$, $b = b_p$, and $s = 1 + s_p = 1 + |x| + |p| = |xp|$.

– Case $t = up \in \texttt{no ne}$. Then $u \in \texttt{no}$ and $p \in \texttt{ne}$. Therefore, there exists a tight derivation $\Phi_u \triangleright \Gamma_u \vdash^{(b_u,s_u)} u : \texttt{tt}$, such that $|u| = s_u$, by the *i.h.* (Lemma 14.2), and there exists a tight derivation $\Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \texttt{n}$, such that $|p| = s_p$ by the *i.h.* (Lemma 14.1). Thus, we can build Φ as follows:

$$\frac{\Phi_u \triangleright \Gamma_u \vdash^{(b_u,s_u)} u : \texttt{tt} \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \texttt{n}}{\Gamma_u + \Gamma_p \vdash^{(b_u+b_p,1+s_u+s_p)} up : \texttt{n}}\ (\texttt{@}_{\texttt{p2}})$$

And we can conclude with $\Gamma = \Gamma_u + \Gamma_p$, $b = b_u + b_p$, and $s = 1 + s_u + s_p = 1 + |u| + |p| = |up|$.

– Case $t = up \in \texttt{ne no}$. Then $u \in \texttt{ne}$ and $p \in \texttt{no}$. Therefore, there exists a tight derivation $\Phi_u \triangleright \Gamma_u \vdash^{(b_u,s_u)} u : \texttt{n}$, such that $|u| = s_u$, by the *i.h.* (Lemma 14.1), and there exists a tight derivation $\Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \texttt{tt}$, such that $|p| = s_p$, by the *i.h.* (Lemma 14.2). Thus, we can build Φ as follows:

$$\frac{\Phi_u \triangleright \Gamma_u \vdash^{(b_u,s_u)} u : \texttt{n} \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \texttt{tt}}{\Gamma_u + \Gamma_p \vdash^{(b_u+b_p,1+s_u+s_p)} up : \texttt{n}}\ (\texttt{@}_{\texttt{p1}})$$

And we can conclude with $\Gamma = \Gamma_u + \Gamma_p$, $b = b_u + b_p$, and $s = 1 + s_u + s_p = 1 + |u| + |p| = |up|$.

2. Case $t \in \texttt{no}$. We want to show that there exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \texttt{tt}$:
   – Case $t = x$. Then we can build Φ as follows:

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

   by picking $\sigma \in \{\texttt{a}, \texttt{v}\}$. And we can conclude with $\Gamma = \emptyset$, $b = 0$, and $s = 0 = |x|$.
   – Case $t = \lambda x.u$. Then we can build Φ as follows:

$$\frac{}{\vdash^{(0,0)} \lambda x.u : \texttt{a}}\ (\lambda_{\texttt{p}})$$

   And we can conclude with $\Gamma = \emptyset$, $b = 0$, and $s = 0 = |\lambda x.u|$.
   – The remaining cases are for when $t \in \texttt{ne}$, so they are subsumed by previous cases.

Lemma 15 (Merge for Values). *Let $(\Phi_v^i \triangleright \Gamma_i \vdash^{(b_i,s_i)} v : \mathcal{M}_i)_{i \in I}$. Then, there exists $\Phi_v \triangleright \Gamma \vdash^{(b,s)} v : \mathcal{M}$, such that $\Gamma = +_{i \in I} \Gamma_i$, $\mathcal{M} = +_{i \in I} \mathcal{M}_i$, $b = +_{i \in I} b_i$, and $s = +_{i \in I}$.*

Proof. We start by noting that each Φ_v^i must end with the rule $(\texttt{m})$. Therefore, for each $i \in I$, we have $\Gamma_i = +_{k \in K_i} \Gamma_k$, $\mathcal{M}_i = [\sigma_k]_{k \in K_i}$, such that $b_i = +_{k \in K_i} b_k$ and $s_i = +_{k \in K_i} s_k$, and the following derivations $(\Phi_v^k \triangleright \Gamma_k \vdash^{(b_k,s_k)} v : \sigma_k)_{k \in K_i}$. Let $J = +_{i \in I} K_i$ and $\mathcal{M} = [\sigma_j]_{j \in J} = [\sigma_k]_{k \in K_i, i \in I}$. We can use rule $(\texttt{m})$ to build $\Phi_v \triangleright \Gamma \vdash^{(+_{j \in J} b_j, +_{j \in J} s_j)} v : \mathcal{M}$. So we can conclude with $\Gamma = +_{j \in J} \Gamma_j = +_{i \in I}(+_{k \in K_i} \Gamma_k) = +_{i \in I} \Gamma_i$, $b = +_{j \in J} b_j = +_{i \in I}(+_{k \in K_i} b_k) = +_{i \in I} b_i$, and $s = +_{j \in J} s_j = +_{i \in I}(+_{k \in K_i} s_k) = +_{i \in I} s_i$.

Soundness and Completeness (Main Results)

Lemma 16 (Substitution and Anti-Substitution).

1. *Let* $\Phi_t \rhd \Gamma_t; x : \mathcal{M} \vdash^{(b_t,s_t)} t : \tau$ *and* $\Phi_v \rhd \Gamma_v \vdash^{(b_v,s_v)} v : \mathcal{M}$*, then there exists* $\Phi_{t\{x\backslash v\}} \rhd \Gamma_t + \Gamma_v \vdash^{(b_t+b_v,s_t+s_v)} t\{x\backslash v\} : \tau$*.*
2. *Let* $\Phi_{t\{x\backslash v\}} \rhd \Gamma_{t\{x\backslash v\}} \vdash^{(b,s)} t\{x\backslash v\} : \tau$*. Then, there exist* $\Phi_t \rhd \Gamma_t; x : \mathcal{M} \vdash^{(b_t,s_t)} t : \tau$ *and* $\Phi_v \rhd \Gamma_v \vdash^{(b_v,s_v)} v : \mathcal{M}$*, such that* $\Gamma_{t\{x\backslash v\}} = \Gamma_t + \Gamma_v$*,* $b = b_t + b_v$*, and* $s = s_t + s_v$*.*

Proof.

1. The proof follows by induction over Φ_t:
   - Case Φ_t ends with rule (ax). Then t must be a variable and we need to consider two cases:
     - Assume $t = y = x$. Then $\Gamma_t = \emptyset$, $\tau = \mathcal{M}$, $t\{x\backslash v\} = v$, $b_t = 0$, and $s_t = 0$. So we can take $\Phi_{t\{x\backslash v\}} = \Phi_v$ and conclude with $\Gamma_t + \Gamma_v = \Gamma_v$, $b_t + b_v = b_v$, and $s_t + s_v = s_v$.
     - Assume $t = y \neq x$. Then $\mathcal{M} = [\,]$, $\Gamma_v = \emptyset$, $t\{x\backslash v\} = t$, $b_v = 0$, and $s_v = 0$. So we can take $\Phi_{t\{x\backslash v\}} = \Phi_t$ and conclude with $\Gamma_t + \Gamma_v = \Gamma_t$, $b_t + b_v = b_t$, and $s_t + s_v = s_t$.
   - Case Φ_t ends with rule (λ). Then t must be of the form $\lambda y.u$ and Φ_t must be of the following form (by α-conversion):

     $$\frac{\Phi_u \rhd \Gamma; x : \mathcal{M} \vdash^{(b_t,s_t)} u : \tau'}{(\Gamma \setminus\!\!\setminus y); x : \mathcal{M} \vdash^{(b_t,s_t)} \lambda y.u : \Gamma(y) \Rightarrow \tau'}\ (\lambda)$$

     where $\tau = \Gamma(y) \Rightarrow \tau'$ and $\Gamma_t = (\Gamma \setminus\!\!\setminus y)$. By the *i.h.*, we have the following derivation $\Phi_{u\{x\backslash v\}} \rhd \Gamma + \Gamma_v \vdash^{(b_t+b_v,s_t+s_v)} u\{x\backslash v\} : \tau$. Therefore, we can construct $\Phi_{t\{x\backslash v\}}$ as follows:

     $$\frac{\Phi_{u\{x\backslash v\}} \rhd \Gamma + \Gamma_v \vdash^{(b_t+b_v,s_t+s_v)} u\{x\backslash v\} : \tau'}{(\Gamma + \Gamma_v) \setminus\!\!\setminus y \vdash^{(b_t+b_v,s_t+s_v)} (\lambda y.u)\{x\backslash v\} : \Gamma(y) \Rightarrow \tau'}\ (\lambda)$$

     And we can conclude with $(\Gamma + \Gamma_v) \setminus\!\!\setminus y = (\Gamma \setminus\!\!\setminus y) + \Gamma_v = \Gamma_t + \Gamma_v$, by α-conversion.
   - Case Φ_t ends with rule (@). Then t must be of the form up and Φ_t must be of the following form:

     $$\frac{\Phi_u \rhd \Gamma; x : \mathcal{M}_1 \vdash^{(b_u,s_u)} u : \mathcal{M}' \Rightarrow \tau \qquad \Phi_p \rhd \Delta; x : \mathcal{M}_2 \vdash^{(b_p,s_p)} p : \mathcal{M}'}{(\Gamma + \Delta); x : \mathcal{M}_1 \sqcup \mathcal{M}_2 \vdash^{(1+b_u+b_p,s_u+s_p)} up : \tau}\ (@)$$

     where $\Gamma_t = (\Gamma + \Delta)$, $\mathcal{M} = \mathcal{M}_1 \sqcup \mathcal{M}_2$, $b_t = 1 + b_u + b_p$, and $s_t = s_u + s_p$. By Lemma 12, we know there exist the following derivations $(\Phi_v^i \rhd \Gamma_v^i \vdash^{(b_i,s_i)} v : \mathcal{M}_i)_{i\in\{1,2\}}$, such that $\Gamma_v = \Gamma_v^1 + \Gamma_v^2$, $b_v = b_1 + b_2$, and $s_v = s_1 + s_2$. By the *i.h.*, we know there exist $\Phi_{u\{x\backslash v\}} \rhd \Gamma + \Gamma_v^1 \vdash^{(b_u+b_1,s_u+s_1)}$

$u\{x\backslash v\} : \mathcal{M}' \Rightarrow \tau$ and $\Phi_{p\{x\backslash v\}} \triangleright \Delta + \Gamma_v^2 \vdash^{(b_p+b_2,s_p+s_2)} p\{x\backslash v\} : \mathcal{M}'$. So we can construct $\Phi_{t\{x\backslash v\}}$ as follows:

$$\frac{\Phi_{u\{x\backslash v\}} \triangleright \Gamma + \Gamma_v^1 \vdash^{(b_u+b_1,s_u+s_1)} u\{x\backslash v\} : \mathcal{M}' \Rightarrow \tau \qquad \Phi_{p\{x\backslash v\}} \triangleright \Delta + \Gamma_v^2 \vdash^{(b_p+b_2,s_p+s_2)} p\{x\backslash v\} : \mathcal{M}'}{(\Gamma + \Delta) + (\Gamma_v^1 + \Gamma_v^2) \vdash^{(1+b_u+b_p+b_1+b_2,s_u+s_p+s_1+s_2)} (up)\{x\backslash v\} : \tau} \ (\mathtt{@})$$

And we can conclude with $\Gamma_t + \Gamma_v = (\Gamma + \Delta) + (\Gamma_v^1 + \Gamma_v^2)$, $b_t + b_v = 1 + b_u + b_p + b_1 + b_2$, and $s_t + s_v = s_u + s_p + s_1 + s_2$.

- Case Φ_t ends with rule $(\mathtt{m})$. Then t must be of the form w and Φ must be of the following form:

$$\frac{(\Phi_w^i \triangleright \Gamma_i; x : \mathcal{M}_i \vdash^{(b_i,s_i)} w : \sigma_i)_{i\in I}}{+_{i\in I}\Gamma_i; x : \sqcup_{i\in I}\mathcal{M}_i \vdash^{(+_{i\in I}b_i,+_{i\in I}s_i)} w : [\sigma_i]_{i\in I}} \ (\mathtt{m})$$

where $\tau = [\sigma_i]_{i\in I}$, $\Gamma_t = +_{i\in I}\Gamma_i$, $b_t = +_{i\in I}b_i$, and $s_t = +_{i\in I}s_i$. By Lemma 12, we have the following derivations $(\Phi_v^i \triangleright \Gamma_v^i \vdash^{(b_v^i,s_v^i)} v : \mathcal{M}_i)_{i\in I}$, such that $\Gamma_v = +_{i\in I}\Gamma_v^i$, $b_v = +_{i\in I}b_v^i$, and $s_v = +_{i\in I}s_v^i$. By the *i.h.* over each Φ_w^i, we have $(\Phi_{w\{x\backslash v\}}^i \triangleright \Gamma_i + \Gamma_v^i \vdash^{(b_i+b_v^i,s_i+s_v^i)} w\{x\backslash v\} : \sigma_i)_{i\in I}$. Therefore, we can construct $\Phi_{t\{x\backslash v\}}$ as follows:

$$\frac{(\Phi_{w\{x\backslash v\}}^i \triangleright \Gamma_i + \Gamma_v^i \vdash^{(b_i+b_v^i,s_i+s_v^i)} w\{x\backslash v\} : \sigma_i)_{i\in I}}{+_{i\in I}(\Gamma_i + \Gamma_v^i) \vdash^{(+_{i\in I}(b_i+b_v^i),+_{i\in I}(s_i+s_v^i))} w\{x\backslash v\} : [\sigma_i]_{i\in I}} \ (\mathtt{m})$$

And we can conclude with $\Gamma_t + \Gamma_v = +_{i\in I}\Gamma_i +_{i\in I} \Gamma_v^i = +_{i\in I}(\Gamma_i + \Gamma_v^i)$, $b_t + b_v = +_{i\in I}b_i +_{i\in I} b_v^i = +_{i\in I}(b_i + b_v^i)$, and $s_t + s_v = +_{i\in I}s_i +_{i\in I} s_v^i = +_{i\in I}(s_i + s_v^i)$.

- Case Φ_t ends with rule $(\lambda_{\mathtt{p}})$. Then t must be of the form $\lambda y.u$, $\Gamma_t = \emptyset$, $\tau = \mathtt{a}$, $\mathcal{M} = [\,]$, $\Gamma_v = \emptyset$, $t\{x\backslash v\} = \lambda y.(u\{x\backslash v\}) = (\lambda y.u)\{x\backslash v\}$, $b_t = b_v = 0$, and $s_t = s_v = 0$. So we can construct $\Phi_{t\{x\backslash v\}}$ as follows:

$$\frac{}{\vdash^{(0,0)} (\lambda y.u)\{x\backslash v\} : \mathtt{a}} \ (\lambda_{\mathtt{p}})$$

And conclude with $\Gamma_t + \Gamma_v = \emptyset$, $b_t + b_v = 0$, and $s_t + s_v = 0$.

- Case Φ_t ends with rule $(\mathtt{@_{p1}})$ or $(\mathtt{@_{p2}})$, the proof is very similar to when Φ_t ends with rule $(\mathtt{@})$.

2. The proof follows by induction over t:
   - Case $t = y$. Then we have to consider two cases:
     - Case $t = y \neq x$. Then, $t\{x\backslash v\} = y$. Let $\Gamma_v = \emptyset$, $\mathcal{M} = [\,]$, $b_v = 0$, and $s_v = 0$. Then, Φ_v is derivable using rule $(\mathtt{m})$. We also take $\Phi_t = \Phi_{t\{x\backslash v\}}$, so that, in particular $\Gamma_t = \Gamma_{t\{x\backslash v\}}$. Then, we conclude with $\Gamma_{t\{x\backslash v\}} = \Gamma_t + \Gamma_v = \Gamma_t$, $b = b_t + b_v = b_t$, and $s = s_t + s_v = s_t$.
     - Case $t = y = x$. Then, $t\{x\backslash v\} = v$. Let $\Gamma_t = \emptyset$, $b_t = 0$, and $s_t = 0$. Now, we have to consider two cases depending on the last rule used in $\Phi_{t\{x\backslash v\}}$:

* Case $\Phi_{t\{x\backslash v\}}$ ends with rule $(\mathtt{ax})$, then $\tau = \sigma$. Let $\Gamma_v = \Gamma_{t\{x\backslash v\}}$, $\mathcal{M} = [\sigma]$, $b_v = b$, and $s_v = s$. Then, we can build derivation Φ_v as follows:
$$\dfrac{\Phi_{t\{x\backslash v\}} \triangleright \Gamma_{t\{x\backslash v\}} \vdash^{(b,s)} v : \sigma}{\Gamma_{t\{x\backslash v\}} \vdash^{(b,s)} v : [\sigma]}\ (\mathtt{m})$$
Let $\Gamma_t = \emptyset$, $b_t = 0$, and $s_t = 0$. Then, $\Phi_t \triangleright x : [\sigma] \vdash^{(0,0)} x : \sigma$ is given by rule $(\mathtt{ax})$. So we can conclude with $\Gamma_{t\{x\backslash v\}} = \Gamma_v = \Gamma_t + \Gamma_v$, $b = b_v = b_t + b_v$, and $s = s_v = s_t + s_v$.
* Case $\Phi_{t\{x\backslash v\}}$ ends with rule $(\mathtt{m})$, then $\tau = [\sigma_i]_{i\in I}$, for some I. Let $\Gamma_t = \emptyset$, and $\mathcal{M} = [\sigma_i]_{i\in I}$. Then, we can build Φ_t as follows:
$$\dfrac{\dfrac{}{(x : [\sigma_i] \vdash^{(0,0)} x : \sigma_i)_{i\in I}}\ (\mathtt{ax})}{x : [\sigma_i]_{i\in I} \vdash^{(0,0)} x : [\sigma_i]_{i\in I}}\ (\mathtt{m})$$
Then, we can take $\Phi_v = \Phi_{t\{x\backslash v\}}$, so that $\Gamma_v = \Gamma_{t\{x\backslash v\}}$, $b_v = b$, and $s_v = s$. And we can conclude $\Gamma_{t\{x\backslash v\}} = \Gamma_v = \Gamma_t + \Gamma_v$, $b = b_v = b_t + b_v$, and $s = s_v = s_t + s_v$.

– Case $t = \lambda y.u$. Then $t\{x\backslash v\} = (\lambda y.u)\{x\backslash v\} = \lambda y.(u\{x\backslash v\})$ and we have to consider three cases:
  • Case $\Phi_{t\{x\backslash v\}}$ ends with rule (λ), then it must be of the following form:
$$\dfrac{\Phi_{u\{x\backslash v\}} \triangleright \Gamma_{u\{x\backslash v\}}; y : \mathcal{M}' \vdash^{(b,s)} u\{x\backslash v\} : \tau'}{\Gamma_{u\{x\backslash v\}} \vdash^{(b,s)} \lambda y.(u\{x\backslash v\}) : \mathcal{M}' \Rightarrow \tau'}\ (\lambda)$$
where $\tau = \mathcal{M}' \Rightarrow \tau'$, and $\Gamma_{t\{x\backslash v\}} = \Gamma_{u\{x\backslash v\}}$. By the *i.h.*, we have the following derivations $\Phi_u \triangleright \Gamma_u; y : \mathcal{M}'; x : \mathcal{M} \vdash^{(b_u,s_u)} u : \delta$ and $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,s_v)} v : \mathcal{M}$, such that $\Gamma_{u\{x\backslash v\}} = \Gamma_u + \Gamma_v$, $b = b_u + b_v$, and $s = s_u + s_v$. And we can build $\Phi_{\lambda y.u}$ as follows:
$$\dfrac{\Phi_u \triangleright \Gamma_u; y : \mathcal{M}'; x : \mathcal{M} \vdash^{(b_u,s_u)} u : \tau'}{\Gamma_u; x : \mathcal{M} \vdash^{(b_u,s_u)} \lambda y.u : \mathcal{M}' \Rightarrow \tau'}\ (\lambda)$$
So we can pick $\Phi_t = \Phi_{\lambda y.u}$, and conclude with $\Gamma_{t\{x\backslash v\}} = \Gamma_{u\{x\backslash v\}} = \Gamma_u + \Gamma_v$, $b = b_u + b_v$, and $s = s_u + s_v$.
  • Case $\Phi_{t\{x\backslash v\}}$ ends with rule $(\lambda_{\mathtt{p}})$, then is must be of the following form:
$$\dfrac{}{\vdash^{(0,0)} \lambda y.(u\{x\backslash v\}) : \mathtt{a}}\ (\lambda_{\mathtt{p}})$$
where $\tau = \mathtt{a}$, $\Gamma_{t\{x\backslash v\}} = \emptyset$, $b = 0$, and $s = 0$. Let $\Gamma_t = \emptyset$, $\mathcal{M} = [\,]$, $b_t = 0$, and $s_t = 0$. Then, we can build Φ_t as follows:
$$\dfrac{}{\vdash^{(0,0)} \lambda y.u : \mathtt{a}}\ (\lambda_{\mathtt{p}})$$
Let $\Gamma_v = \emptyset$, $b_v = 0$, and $s_v = 0$. Then Φ_v can be constructed by using rule $(\mathtt{m})$ with no premises. So we can conclude with $\Gamma_{t\{x\backslash v\}} = \emptyset = \Gamma_t + \Gamma_v$, and $b = 0 = b_t + b_v$, and $s = 0 = s_t + s_v$.

- Case $\Phi_{t\{x\backslash v\}}$ ends with rule ($\mathtt{m}$). Then $t\{x\backslash v\}$ and t are values, and $\Phi_{t\{x\backslash v\}}$ must be of the following form:

$$\frac{(\Phi_i \triangleright \Gamma_i \vdash^{(b_i,s_i)} t\{x\backslash v\} : \sigma_i)_{i\in I}}{+_{i\in I}\Gamma_i \vdash^{(+_{i\in I}b_i,+_{i\in I}s_i)} t\{x\backslash v\} : [\sigma_i]_{i\in I}}\ (\mathtt{m})$$

where $\tau = [\sigma_i]_{i\in I}$, $\Gamma_{t\{x\backslash v\}} = +_{i\in I}\Gamma_i$, $b = +_{i\in I}b_i$, and $s = +_{i\in I}s_i$. By the *i.h.* over each Φ_i, we have the following derivations $\Phi^i_t \triangleright \Gamma^i_t; x : \mathcal{M}_i \vdash^{(b^i_t,s^i_t)} t : \sigma_i$ and $\Phi^i_v \triangleright \Gamma^i_v \vdash^{(b^i_v,s^i_v)} v : \mathcal{M}_i$, such that $\Gamma_i = \Gamma^i_t + \Gamma^i_v$, $b_i = b^i_t + b^i_v$, and $s_i = s^i_t + s^i_v$,for each $i \in I$. So we can build Φ_t as follows:

$$\frac{(\Phi^i_t \triangleright \Gamma^i_t; x : \mathcal{M}_i \vdash^{(b^i_t,s^i_t)} t : \sigma_i)_{i\in I}}{+_{i\in I}\Gamma^i_t; x : \sqcup_{i\in I}\mathcal{M}_i \vdash^{(+_{i\in I}b^i_t,+_{i\in I}s^i_t)} t : [\sigma_i]_{i\in I}}\ (\mathtt{m})$$

such that $\Gamma_t = +_{i\in I}\Gamma^i_t$, $\mathcal{M} = \sqcup_{i\in I}\mathcal{M}_i$, $b_t = +_{i\in I}b^i_t$, and $s_t = +_{i\in I}s^i_t$. By Lemma 15, we can take the following derivation $\Phi_v \triangleright +_{i\in I}\Gamma^i_v \vdash^{(+_{i\in I}b^i_v,+_{i\in I}s^i_v)} v : \mathcal{M}$. And we can conclude with $\Gamma_{t\{x\backslash v\}} = +_{i\in I}\Gamma_i = +_{i\in I}(\Gamma^i_t + \Gamma^i_v) = +_{i\in I}\Gamma^i_t +_{i\in I} \Gamma^i_v = \Gamma_t + \Gamma_v$, $b = +_{i\in I}b_i = +_{i\in I}(b^i_t + b^i_v) = +_{i\in I}b^i_t +_{i\in I} b^i_v = b_t + b_v$, and $s = +_{i\in I}s_i = +_{i\in I}(s^i_t + s^i_v) = +_{i\in I}s^i_t +_{i\in I} s^i_v = s_t + s_v$.

– Case $t = up$. Then $t\{x\backslash v\} = (u\{x\backslash v\})(p\{x\backslash v\})$ and we have to consider three cases:
  - Case $\Phi_{t\{x\backslash v\}}$ ends with ($\mathtt{@}$), then it must be of the following form:

$$\frac{\Phi_{u\{x\backslash v\}} \triangleright \Gamma_{u\{x\backslash v\}} \vdash^{(b',s')} u\{x\backslash v\} : \mathcal{M}' \Rightarrow \tau \qquad \Phi_{p\{x\backslash v\}} \triangleright \Gamma_{p\{x\backslash v\}} \vdash^{(b'',s'')} p\{x\backslash v\} : \mathcal{M}'}{\Gamma_{u\{x\backslash v\}} + \Gamma_{p\{x\backslash v\}} \vdash^{(1+b'+b'',s'+s'')} (u\{x\backslash v\})(p\{x\backslash v\}) : \tau}\ (\mathtt{@})$$

where $\Gamma_{t\{x\backslash v\}} = \Gamma_{u\{x\backslash v\}} + \Gamma_{p\{x\backslash v\}}$, $b = 1 + b' + b''$, and $s = s' + s''$. By the *i.h.* over $\Phi_{u\{x\backslash v\}}$, we have the following derivations $\Phi_u \triangleright \Gamma_u; x : \mathcal{M}_1 \vdash^{(b_u,s_u)} u : \mathcal{M}' \Rightarrow \tau$ and $\Phi^1_v \triangleright \Gamma^1_v \vdash^{(b^1_v,s^1_v)} v : \mathcal{M}_1$, such that $\Gamma_{u\{x\backslash v\}} = \Gamma_u + \Gamma^1_v$, $b' = b_u + b^1_v$, and $s' = s_u + s^1_v$. And by the *i.h.* over $\Phi_{p\{x\backslash v\}}$, we have the following derivation $\Phi_p \triangleright \Gamma_p; x : \mathcal{M}_2 \vdash^{(b_p,s_p)} p : \mathcal{M}'$ and $\Phi^2_v \triangleright \Gamma^2_v \vdash^{(b^2_v,s^2_v)} v : \mathcal{M}_2$, such that $\Gamma_{p\{x\backslash v\}} = \Gamma_p + \Gamma^2_v$, $b'' = b_p + b^2_v$, and $s'' = s_p + s^2_v$. By Lemma 15, we can take the following derivation $\Phi_v \triangleright \Gamma^1_v + \Gamma^2_v \vdash^{(b^1_v+b^2_v,s^1_v+s^2_v)} v : \mathcal{M}_1 \sqcup \mathcal{M}_2$, such that $\Gamma_v = \Gamma^1_v + \Gamma^2_v$, $b_v = b^1_v + b^2_v$, and $s_v = s^1_v + s^2_v$. And we can build Φ_{up} as follows:

$$\frac{\Phi_u \triangleright \Gamma_u; x : [\sigma_i]_{i\in I_1} \vdash^{(b_u,s_u)} u : \mathcal{M}' \Rightarrow \tau \qquad \Phi_p \triangleright \Gamma_p; x : [\sigma_i]_{i\in I_2} \vdash^{(b_p,s_p)} p : \mathcal{M}'}{(\Gamma_u + \Gamma_p); x : [\sigma_i]_{i\in I} \vdash^{(1+b_u+b_p,s_u+s_p)} up : \tau}\ (\mathtt{@})$$

such that $\Gamma_t = \Gamma_u + \Gamma_p$, $b_t = 1 + b_u + b_p$, and $s_t = s_u + s_p$. So we can pick $\Phi_t = \Phi_{up}$, and conclude with $\Gamma_{t\{x\backslash v\}} = \Gamma_{u\{x\backslash v\}} + \Gamma_{p\{x\backslash v\}} = \Gamma_u + \Gamma^1_v + \Gamma_p + \Gamma^2_v = (\Gamma_u + \Gamma_p) + (\Gamma^1_v + \Gamma^2_v) = \Gamma_t + \Gamma_v$, $b = 1 + b' + b'' = 1 + b_u + b^1_v + b_p + b^2_v = 1 + (b_u + b_p) + (b^1_v + b^2_v) = b_t + b_v$, and $s = s_u + s^1_v + s_p + s^2_v = (s_u + s_p) + (s^1_v + s^2_v) = s_t + s_v$.

- Case $\Phi_{t\{x\backslash v\}}$ ends with $(@_{\mathtt{p1}})$ and $(@_{\mathtt{p2}})$. These cases are very similar to the case where $\Phi_{t\{x\backslash v\}}$ ends with rule $(@)$.

Lemma 17 (Split Exact Subject Reduction and Expansion).

1. *Let $\Phi_t \triangleright \Gamma \vdash^{(b,s)} t : \tau$ be tight. If $t \to t'$, then there exists $\Phi_{t'} \triangleright \Gamma \vdash^{(b-1,s)} t' : \tau$.*
2. *Let $\Phi_{t'} \triangleright \Gamma \vdash^{(b,s)} t' : \tau$ be tight. If $t \to t'$, then there exists $\Phi_t \triangleright \Gamma \vdash^{(b+1,s)} t : \tau$.*

Proof.

1. We will actually prove the following stronger version of the statement, which allows us to reason inductively:
   Let $\Phi_t \triangleright \Gamma \vdash^{(b,s)} t : \tau$, such that Γ is tight, and either τ is tight or $\neg\mathtt{val}(t)$. If $t \to t'$, then there exists $\Phi_{t'} \triangleright \Gamma \vdash^{(b-1,s)} t' : \tau$.
   The proof now follows by induction over $\to$:
   - Case $t = (\lambda x.u)v \to u\{x\backslash v\} = t'$.Assume that Φ_t ends with rule $(@_{\mathtt{p1}})$. Then $\lambda x.u$ must be assigned type $\bar{\mathtt{a}}$, which is not possible by Lemma 9. Now, assume that Φ_t ends with rule $(@_{\mathtt{p2}})$. Then v must be assigned typed $\mathtt{n}$, which is not possible by Lemma 8. Therefore, Φ_t must be of the following form:

$$\dfrac{\dfrac{\Phi_u \triangleright \Gamma_u; x : \mathcal{M} \vdash^{(b_u,s_u)} u : \tau}{\Gamma_u \vdash^{(b_u,s_u)} (\lambda x.u) : \mathcal{M} \Rightarrow \tau}\ (\lambda) \qquad \Phi_v \triangleright \Gamma_v \vdash^{(b_v,s_v)} v : \mathcal{M}}{\Gamma_u + \Gamma_v \vdash^{(1+b_u+b_v,s_u+s_v)} (\lambda x.u)v : \tau}\ (@)$$

     where $\tau \in \mathtt{tt}$, $\Gamma = \Gamma_u + \Gamma_v$ is tight, $b = 1 + b_u + b_v$, and $s = s_u + s_v$. By Lemma 16.1, we know there exists the following derivation $\Phi_{u\{x\backslash v\}} \triangleright \Gamma_u + \Gamma_v \vdash^{(b_u+b_v,s_u+s_v)} u\{x\backslash v\} : \tau$. So we can take $\Phi_{t'} = \Phi_{u\{x\backslash v\}}$ and conclude with $b - 1 = b_u + b_v$.
   - Case $t = up \to u'p = t'$, such that $u \to u'$. Then Φ_t must either end with $(@)$, $(@_{\mathtt{p1}})$, or $(@_{\mathtt{p2}})$:
     - Case Φ_t ends with rule $(@)$, then it must be of the following form:

$$\dfrac{\Phi_u \triangleright \Gamma_u \vdash^{(b_u,s_u)} u : \mathcal{M} \Rightarrow \tau \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \mathcal{M}}{\Gamma_u + \Gamma_p \vdash^{(1+b_u+b_p,s_u+s_p)} up : \tau}\ (@)$$

       where $\tau = \tau \in \mathtt{tt}$, $\Gamma = \Gamma_u + \Gamma_p$ is tight, $b = 1 + b_u + b_p$, and $s = s_u + s_p$. Since $u \to u'$, it is clear that $\neg\mathtt{val}(u)$ holds. Moreover, Γ_u is necessarily tight. Therefore, by the *i.h.*, there exists $\Phi_{u'} \triangleright \Gamma_u \vdash^{(b_u-1,s_u)} u' : \mathcal{M} \Rightarrow \tau$. Thus, we can build $\Phi_{t'}$ as follows:

$$\dfrac{\Phi_{u'} \triangleright \Gamma_u \vdash^{(b_u-1,s_u)} u' : \mathcal{M} \Rightarrow \tau \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,s_p)} p : \mathcal{M}}{\Gamma_u + \Gamma_p \vdash^{(b_u+b_p,s_u+s_p)} u'p : \tau}\ (@)$$

       And we can conclude with $b - 1 = b_u + b_p$.
     - Case Φ_t ends with rule $(@_{\mathtt{p1}})$ or $(@_{\mathtt{p2}})$, the proof are similar to the one where Φ_t ends with rule $(@)$.

- Case $t = up \to up' = t'$, such that $u \not\to$ and $p \to p'$. Then Φ_t must either end with $(\mathtt{@})$, $(\mathtt{@_{p1}})$, or $(\mathtt{@_{p2}})$:
  - Case Φ_t ends with rule $(\mathtt{@})$, then it must be of the following form:

$$\frac{\Phi_u \rhd \Gamma_u \vdash^{(b_u,s_u)} u : \mathcal{M} \Rightarrow \tau \qquad \Phi_p \rhd \Gamma_p \vdash^{(b_p,s_p)} p : \mathcal{M}}{\Gamma_u + \Gamma_p \vdash^{(1+b_u+b_p,s_u+s_p)} up : \tau} \; (\mathtt{@})$$

  where $\tau \in \mathtt{tt}$, $\Gamma = \Gamma_u + \Gamma_p$ is tight, $b = 1 + b_u + b_p$, and $s = s_u + s_p$. Since $p \to p'$, it is clear that $\neg\mathtt{val}(p)$. Moreover, Γ_p is necessarily tight. Therefore, by the *i.h.*, we know there exists the following derivation $\Phi_{p'} \rhd \Gamma_p \vdash^{(b_p-1,s_p)} p' : \mathcal{M}$. Thus, we can build $\Phi_{t'}$ as follows:

$$\frac{\Phi_u \rhd \Gamma_u \vdash^{(b_u,s_u)} u : \mathcal{M} \Rightarrow \tau \qquad \Phi_{p'} \rhd \Gamma_p \vdash^{(b_p-1,s_p)} p' : \mathcal{M}}{\Gamma_u + \Gamma_p \vdash^{(b_u+b_p,s_u+s_p)} up' : \tau} \; (\mathtt{@})$$

  And we can conclude with $b - 1 = b_u + b_p$.
  - Case Φ_t ends with rule $(\mathtt{@_{p1}})$ or $(\mathtt{@_{p2}})$, the proofs are similar to the ones where Φ_t ends with rule $(\mathtt{@})$.

2. Just like for Lemma 17.1, we will actually prove the following stronger version of the statement, which allows us to reason inductively:
Let $\Phi_{t'} \rhd \Gamma \vdash^{(b,s)} t' : \tau$, such that Γ is tight, and either ($\tau \in \mathtt{tt}$ or $\neg\mathtt{val}(t)$).
If $t \to t'$, then there exists $\Phi_t \rhd \Gamma \vdash^{(b+1,s)} t : \tau$.
The proof now follows by induction over $\to$:
   - Case $t = (\lambda x.u)v \to u\{x \backslash v\} = t'$. Then $\Phi_{t'} \rhd \Gamma \vdash^{(b,s)} u\{x \backslash v\} : \tau$ and, by Lemma 16.2, there exist the following derivations $\Phi_u \rhd \Gamma_u; x : \mathcal{M} \vdash^{(b_u,s_u)} u : \tau$ and $\Phi_v \rhd \Gamma_v \vdash^{(b_v,s_v)} v : \mathcal{M}$, such that $\tau \in \mathtt{tt}$, $\Gamma = \Gamma_u + \Gamma_v$ is tight, $b = b_u + b_v$, and $s = s_u + s_v$. So we can build Φ_t as follows:

$$\frac{\dfrac{\Phi_u \rhd \Gamma_u; x : \mathcal{M} \vdash^{(b_u,s_u)} u : \tau}{\Gamma_u \vdash^{(b_u,s_u)} \lambda x.u : \mathcal{M} \Rightarrow \tau} \; (\lambda) \qquad \Phi_v \rhd \Gamma_v \vdash^{(b_v,s_v)} v : \mathcal{M}}{\Gamma_u + \Gamma_v \vdash^{(1+b_u+b_v,s_u+s_v)} (\lambda x.u)v : \tau} \; (\mathtt{@})$$

     And we can conclude with $b + 1 = 1 + b_u + b_v$.
   - Case $t = up \to u'p = t'$, such that $u \to u'$. Then $\Phi_{t'}$ must either end with $(\mathtt{@})$, $(\mathtt{@_{p1}})$, or $(\mathtt{@_{p2}})$:
     - Case $\Phi_{t'}$ ends with rule $(\mathtt{@})$, then it must be of the following form:

$$\frac{\Phi_{u'} \rhd \Gamma_u \vdash^{(b_u,s_u)} u' : \mathcal{M}' \Rightarrow \tau \qquad \Phi_p \rhd \Gamma_p \vdash^{(b_p,s_p)} p : \mathcal{M}'}{\Gamma_u + \Gamma_p \vdash^{(1+b_u+b_p,s_u+s_p)} u'p : \tau} \; (\mathtt{@})$$

     where $\tau \in \mathtt{tt}$, $\Gamma = \Gamma_u + \Gamma_p$ it tight, $b = 1 + b_u + b_p$, and $s = s_u + s_p$. Since $u \to u'$, it is clear that $\neg\mathtt{val}(u)$. Moreover, Γ_p is tight. Therefore, by the *i.h.*, there exists the following derivation $\Phi_u \rhd \Gamma_u \vdash^{(b_u+1,s_u)} u : \mathcal{M}' \Rightarrow \tau$. Thus, we can build $\Phi_{t'}$ as follows:

$$\frac{\Phi_u \rhd \Gamma_u \vdash^{(b_u+1,s_u)} u : \mathcal{M}' \Rightarrow \tau \qquad \Phi_p \rhd \Gamma_p \vdash^{(b_p,s_p)} p : \mathcal{M}'}{\Gamma_u + \Gamma_p \vdash^{(1+b_u+1+b_p,s_u+s_p)} up : \tau} \; (\mathtt{@})$$

And we can conclude with $b+1 = (1+b_u+b_p)+1 = 1+b_u+1+b_p$.

- Case $\Phi_{t'}$ ends with rule ($\mathtt{@_{p1}}$) or ($\mathtt{@_{p2}}$), the proofs are similar to the one where $\Phi_{t'}$ ends with rule ($\mathtt{@}$).

– Case $t = up \to up' = t'$, such that $p \to p'$. Then $\Phi_{t'}$ must either ends with ($\mathtt{@}$), ($\mathtt{@_{p1}}$), or ($\mathtt{@_{p2}}$):

- Case $\Phi_{t'}$ ends with rule ($\mathtt{@}$), then it must be of the following form:

$$\frac{\Phi_u \triangleright \Gamma_u \vdash^{(b_u,s_u)} u : \mathcal{M}' \Rightarrow \tau \qquad \Phi_{p'} \triangleright \Gamma_p \vdash^{(b_p,s_p)} p' : \mathcal{M}'}{\Gamma_u + \Gamma_p \vdash^{(1+b_u+b_p,s_u+s_p)} up' : \tau} \ (\mathtt{@})$$

where $\tau \in \mathtt{tt}$, $\Gamma = \Gamma_u + \Gamma_{p'}$ is tight, $b = 1 + b_u + b_p$, $s_t = s_u + s_p$. Since $p \to p'$, it is clear that $\neg\mathtt{val}(p)$ holds. Moreover, Γ_p is tight. Therefore, by the *i.h.*, we have the following derivation $\Phi_p \triangleright \Gamma_p \vdash^{(b_p+1,s_p)} p : \mathcal{M}' \Rightarrow \tau$. Thus, we can build $\Phi_{t'}$ as follows:

$$\frac{\Phi_u \triangleright \Gamma \vdash^{(b_u,s_u)} u : \mathcal{M}' \Rightarrow \tau \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p+1,s_p)} p : \mathcal{M}'}{\Gamma_u + \Gamma_p \vdash^{(1+b_u+b_p+1,s_u+s_p)} up : \tau} \ (\mathtt{@})$$

And we can conclude with $b+1 = (1+b_u+b_p)+1 = 1+b_u+b_p+1$.

- Case $\Phi_{t'}$ ends with rule ($\mathtt{@_{p1}}$) or ($\mathtt{@_{p2}}$), the proofs are similar to the one where $\Phi_{t'}$ ends with rule ($\mathtt{@}$).

Theorem 2 (Quantitative Soundness and Completeness).
If $\Phi \triangleright \Gamma \vdash^{(b,s)} t : \tau$ is tight, then there exists $u \in \mathtt{no}$ such that $t \twoheadrightarrow^b u$ with $|u| = s$.
If $t \twoheadrightarrow^b u$ with $u \in \mathtt{no}$, then there exists a tight type derivation $\Phi_t \triangleright \Gamma \vdash^{(b,|u|)} t : \tau$.

Proof.

1. The proof follows by induction over b:
   – Case $b = 0$. Then $t \in \mathtt{no}$, by Lemma 10. And $d = |t|$, by Lemma 11. So we can conclude with $u = t$.
   – Case $b > 0$. Then $t \notin \mathtt{no}$, by Lemma 10. Therefore, there exists t' such that $t \to t'$, by Prop. 1. ByLemma 17.1, there exists $\Phi_{t'} \triangleright \Gamma \vdash^{(b-1,s)} t' : \tau$. By the *i.h.*, there exists $u \in \mathtt{no}$, such that $t' \twoheadrightarrow^{b-1} u$, such that $d = |u|$. So we can conclude with $t \to t' \twoheadrightarrow^{b-1} u$, which means that $t \twoheadrightarrow^b u$, as expected.
2. The proof follows by induction over b:
   – Case $b = 0$. Then $t = u$, which means that $t \in \mathtt{no}$. Therefore, we can conclude by Lemma 14.
   – Case $b > 0$. Then there exists t', such that $t \to t' \twoheadrightarrow^{b-1} u$. By the *i.h.*, there exists a tight derivation $\Phi_{t'} \triangleright \Gamma \vdash^{(b-1,|u|)} t' : \tau$. ByLemma 17.2, there exists a tight derivation $\Phi \triangleright \Gamma \vdash^{(b,|u|)} t : \tau$. So, we can conclude.

A.2 A λ-Calculus with Global State

General Lemmas

Proposition 2. *Let (t, s) be a configuration. Then (t, s) is final iff $(t, s) \not\to$.*

Proof. $\Rightarrow$) Let (t, s) be final. We consider two cases:

- Case (t, s) is blocked. We reason by induction on blocked configurations.
  - ∗ Case $(t, s) = (\mathtt{get}_l(\lambda x.u), s)$, such that $l \notin \mathtt{dom}(s)$. Then $(t, s) \not\to$ is straightforward.
  - ∗ Case $(t, s) = (vu, s)$ and (u, s) is blocked. Then by the *i.h.*, we have that $(u, s) \not\to$. Therefore, $(vu, s) \not\to$ holds.
- Case $t \in \mathtt{no}$. We reason by induction on $\mathtt{no}$.
  - ∗ Case $t = v \in \mathtt{Val}$. Then $(v, s) \not\to$ is straightforward.
  - ∗ Case $t \in \mathtt{ne}$. Then $t = vu$ and we have to consider two different cases:
    - · Case $v = x$ and $u \in \mathtt{no}$. Then by the *i.h.*, we have $(u, s) \not\to$. Therefore, $(vu, s) \not\to$ holds.
    - · Case $v = (\lambda x.p)$ and $u \in \mathtt{ne}$. Then $u \in \mathtt{no}$, and by the *i.h.*, we have that $(u, s) \not\to$. Therefore $(vu, s) \not\to$ holds.

$\Leftarrow$) Let $t \not\to$. We reason by induction on t:

- Case $t = v$. Then $t \in \mathtt{no}$. Therefore (t, s) is final.
- Case $t = vu$. Since $(vu, s) \not\to$, then $(u, s) \not\to$. By the *i.h.*, we have (u, s) final. Now, we reason by cases:
  - ∗ Case (u, s) is blocked. Then, (vu, s) is blocked by definition.
  - ∗ Case $u \in \mathtt{no}$. Then we have two cases:
    - · Case $u \in \mathtt{ne}$. Then $vu \in \mathtt{no}$. Therefore, (t, s) is final.
    - · Case $u \in \mathtt{Val}$ and $v = \lambda x.p$. Then $((\lambda x.p)u, s) \to (p\{x \backslash u\}, s)$, which yields a contradiction with the hypothesis $t = vu \not\to$. Thus, this case does not apply.
- Case $t = \mathtt{get}_l(\lambda x.u)$. Since $(\mathtt{get}_l(\lambda x.u), s) \not\to$, then $l \notin \mathtt{dom}(s)$. Therefore, $(\mathtt{get}_l(\lambda x.u), s)$ is blocked, which implies (t, s) is final.
- Case $t = \mathtt{set}_l(v, u)$. Then $(\mathtt{set}_l(v, u), s) \to (u, \mathtt{upd}_l(v, s))$, which yields to a contraction with the hypothesis $t \not\to$. Therefore, this case does not apply.

Proposition 3. *If $\Phi \triangleright \Gamma \vdash^{(b,m,d)} (t, s) : \kappa$, then (t, s) is unblocked.*

Proof. By induction on t:

– Case $t \in \mathtt{Val}$ or $t = \mathtt{set}_l(v, t)$. Then the conclusion trivially holds, since clearly (t, s) is not a blocked configuration.

– Case $t = \mathtt{get}_l(\lambda x.t)$. We have two cases:

- Case $l \in \mathtt{dom}(s)$. Then (t, s) is clearly unblocked.

- Case $l \notin \mathtt{dom}(s)$. Let $\mathcal{S}_0 = \{(l : \Gamma(x))\} \uplus \mathcal{S}$. Since $t = \mathtt{get}_l(\lambda x.u)$, then Φ must be of the following form:

$$\dfrac{\Phi \triangleright \Gamma_u \setminus\!\!\setminus x \vdash^{(b_u,m_u,d_u)} \mathtt{get}_l(\lambda x.u) : \mathcal{S}_0 \gg \kappa \qquad \Phi_s \triangleright \Delta \vdash^{(b_s,m_s,d_s)} s : \mathcal{S}_0}{(\Gamma_u \setminus\!\!\setminus x) + \Delta \vdash^{(b_u+b_s,1+m_u+m_s,d_u+d_s)} (\mathtt{get}_l(\lambda x.t), s) : \kappa}\ (\mathtt{conf})$$

where $\Gamma = \Gamma_u \setminus\!\!\setminus x$, $b = b_u + b_s$, $m = 1 + m_u + m_s$, and $d = d_u + d_s$. Thus, $l \in \mathtt{dom}(\{(l : \Gamma_u(x))\} \uplus \mathcal{S})$, and so by Lemma 20 we have $l \in \mathtt{dom}(s)$, which gives a contradiction with the hypothesis $l \notin \mathtt{dom}(s)$. Therefore, this case does not apply,

– Case $t = vu$. Assume $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa)$ and $\Phi_u \triangleright \Gamma_u \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}')$. Then Φ must be of the following form:

$$\dfrac{\dfrac{\Phi_v \qquad \Phi_u}{\Gamma_v + \Gamma_u \vdash^{(1+b_v+b_u,m_v+m_u,d_v+d_u)} vu : \mathcal{S} \gg \kappa}\ (@) \qquad \Phi_s \triangleright \Delta \vdash^{(b_s,m_s,d_s)} s : \mathcal{S}}{(\Gamma_v + \Gamma_u) + \Delta \vdash^{(1+b_v+b_u+b_s,m_v+m_u+m_s,d_v+d_u+d_s)} (vu, s) : \kappa}\ (\mathtt{conf})$$

where $\Gamma = (\Gamma_v + \Gamma_u) + \Delta$, $b = 1 + b_v + b_u + b_s$, $m = m_v + m_u + m_s$, and $d = d_v + d_u + d_s$. Thus, we can build the following derivation for (u, s):

$$\dfrac{\Phi_u \triangleright \Gamma_u \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}') \qquad \Phi_s \triangleright \Delta \vdash^{(b_s,m_s,d_s)} s : \mathcal{S}}{\Gamma + \Delta \vdash^{(b_u+b_s,m_u+m_s,d_u+d_s)} (u, s) : \mathcal{M} \times \mathcal{S}'}\ (\mathtt{conf})$$

By the *i.h.*, we have that (u, s) is unblocked. Therefore, (vu, s) also unblocked.

Lemma 18 (Relevance). *Let* $\Phi \triangleright \Gamma \vdash^{(b,m,d)} t : \mathcal{T}$ *(resp.* $\Phi' \triangleright \Gamma \vdash^{(b',m',d')} s : \mathcal{S}$*). Then* $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(t)$ *(resp.* $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(s)$*).*

Proof. The proof following by induction over Φ (resp. Φ'). Case Φ (resp. Φ') ends with rule $(\mathtt{ax})$, $(\mathtt{ax_p})$, or $(\lambda_\mathtt{p})$ (resp. rule $(\mathtt{emp})$), then Φ (resp. Φ') is clearly relevant. The other cases follow easily from the *i.h.*.

Soundness Lemmas (Auxiliary Lemmas)

Lemma 1.

1. *Let* $\Phi \triangleright \Gamma \vdash^{(0,0,d)} t : \delta$ *be tight. Then, (1)* $t \in \mathtt{no}$ *and (2)* $d = |t|$.
2. *Let* $\Phi \triangleright \Delta \vdash^{(0,0,d)} s : \mathcal{S}$ *be tight. Then* $d = 0$.

Proof.

1. We replace the statement by the following three ones.
   (1.1) If $\delta = \mathcal{S} \gg (\mathtt{n} \times \mathcal{S}')$, then $t \in \mathtt{ne}$.
   (1.2) If $\delta = \mathcal{S} \gg (\mathtt{tt} \times \mathcal{S}')$, then $t \in \mathtt{no}$.
   (2) $d = |t|$.
   We reason by simultaneous induction on tight derivations.

- Case Φ ends with $(\mathtt{ax})$. This case does not apply since the resulting type is not a monadic type.
- Case Φ ends with (λ). This case does not apply since the resulting type is not a monadic type.
- Case Φ ends with $(\mathtt{@})$. Then the first counter in the conclusion of the derivation is necessarily greater than 0 and thus this case does not apply.
- Case Φ ends with $(\mathtt{m})$. This case does not apply since the resulting type is not a monadic type.
- Case Φ ends with $(\uparrow)$. Then Φ cannot be tight.
- Case Φ ends with $(\mathtt{get})$. Then the second counter in the conclusion of the derivation is necessarily greater than 0 and thus this case does not apply.
- Case Φ ends with $(\mathtt{set})$. Then the second counter in the conclusion of the derivation is necessarily greater than 0 and thus this case does not apply.
- Case Φ ends with $(\mathtt{ax_p})$, so that $t = x$ and $s = 0$. The condition of case (1.1) is not possible by construction. In case (1.2) we can conclude $x \in \mathtt{Val} \subseteq \mathtt{no}$. The statement (2) $d = 0 = |x|$ is straightforward.
- Case Φ ends with $(\lambda_\mathtt{p})$ so that $t = \lambda x.u$ and $d = 0$. The condition of case (1.1) is not possible by construction. In case (1.2) we can conclude $\lambda x.u \in \mathtt{Val} \subseteq \mathtt{no}$. The statement (2) $d = 0 = |\lambda x.u|$ is straightforward.
- Case Φ ends with $(\mathtt{@_{p1}})$, so that $t = xu$ and $d = 1 + d'$. If the condition of case (1.1) holds for t, that means that the condition of case (1.2) holds for u. By the *i.h.* (1.2) $u \in \mathtt{no}$ so that $xu \in \mathtt{ne}$. The condition of case (1.2) holds for t only for $\mathtt{tt} = \mathtt{n}$, and then $xu \in \mathtt{ne}$ holds by case (1.1), which implies $xu \in \mathtt{no}$ by definition. To show statement (2), we apply the *i.h.* (2) to u and obtain $d' = |u|$, then $d = 1 + d' = 1 + |u| = |t|$.
- Case Φ ends with $(\mathtt{@_{p2}})$, so that $t = (\lambda x.p)u$ and $d = 1 + d'$. If the condition of case (1.1) holds for t, that means that the condition of case (1.1) holds for u. By the *i.h.* (1.1) $u \in \mathtt{ne}$ so that $t \in \mathtt{ne}$. The condition of case (1.2) holds for t only for $\mathtt{tt} = \mathtt{n}$, and then $t \in \mathtt{ne}$ holds by case (1.1), which implies $t \in \mathtt{no}$ by definition. To show statement (3), we apply the *i.h.* (3) to u and obtain $d' = |u|$, then $d = 1 + d' = 1 + |u| = |t|$.

2. By induction over Φ:
   - Case Φ ends with $(\mathtt{emp})$. Then it must be of the following form:

$$\frac{}{\vdash^{(0,0,0)} \epsilon : \emptyset}\ (\mathtt{emp})$$

   where $d = 0$. So we can conclude.
   - Case Φ ends with $(\mathtt{upd})$. Then it must be of the following form:

$$\frac{\Phi_v \triangleright \Gamma_v \vdash^{(0,0,d_v)} v : \mathcal{M} \qquad \Phi_q \triangleright \Delta_q \vdash^{(0,0,d_q)} q : \mathcal{S}_q}{\Gamma_v + \Delta_q \vdash^{(0,0,d_v+d_q)} \mathtt{upd}_l(v,q) : \{l : \mathcal{M}\}; \mathcal{S}_q}\ (\mathtt{upd})$$

   where $d = d_v + d_q$. Then Φ_v must be of the form:

$$\frac{(\Gamma_v^i \vdash^{(0,0,d_v^i)} v : \sigma_i)_{i\in I}}{+_{i\in I}\Gamma_v^i \vdash^{(0,0,+_{i\in I}d_v^i)} v : [\sigma_i]_{i\in I}}\ (\mathtt{m})$$

where $\Gamma_v = +_{i\in I}\Gamma_v^i$, $d_v = +_{i\in I}d_v^i$, and $\mathcal{M} = [\sigma_i]_{i\in I}$. Given that $\{l : \mathcal{M}\};\mathcal{S}_q$ is tight, then $\mathtt{tight}(\mathcal{M})$, and so σ_i is tight, for each $i \in I$. Then by point (1) of Lemma 1, $|v| = d_v^i$ for each $i \in I$. But since $v \in \mathtt{Val}$ then its size is 0, which means $d_v^i = 0$ for each $i \in I$, therefore $d_v = +_{i\in I}d_v^i = 0$. Furthermore, $d_q = 0$, by the *i.h.* Therefore we can conclude $d = d_v + d_q = +_{i\in I}d_v^i + 0 = 0 + 0 = 0$.

Lemma 19. *Let $\Phi \triangleright \Gamma \vdash^{(0,0,d)} t : \delta$ be tight. If $t \in \mathtt{no}$, then $\delta = \mathcal{S} \to \mathtt{tt} \times \mathcal{S}'$ and $\mathcal{S} = \mathcal{S}'$.*

Proof. By induction on $t \in \mathtt{no}$. We consider two cases:

– Case $t \in \mathtt{Val}$. Then such a typing derivation can only end with rule $(\mathtt{ax_p})$ or $(\lambda_\mathtt{p})$, in which cases the statement is obvious.
– Case $t = vu \in \mathtt{ne}$. Since the first counter of the derivation is 0, Φ can only end with a persistent rule $(@_\mathtt{p1})$ or $(@_\mathtt{p2})$. In both cases, we can conclude by applying the *i.h.* to $u \in \mathtt{no}$ or $u \in \mathtt{ne}$ and their type derivations, which gives $\mathcal{S} = \mathcal{S}'$.

Lemma 2. *Let $\Phi \triangleright \Gamma \vdash^{(b,m,d)} t : \delta$ be tight. Then, $b = m = 0$ iff $t \in \mathtt{no}$.*

Proof.

$\Rightarrow$) By point (1) of Item 1.
$\Leftarrow$) By induction on t:
- Case $t \in \mathtt{Val}$. There are six cases to consider for Φ:
  - Φ ends with $(\mathtt{ax})$. This case does not apply since the resulting type is not a monadic type.
  - Φ ends with (λ). This case does not apply since the resulting type is not a monadic type.
  - Φ ends with $(\mathtt{m})$. This case does not apply since the resulting type is not a monadic type.
  - Φ ends with $(\uparrow)$. This case does not apply, since $\delta = \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}')$, but $\mathcal{M} \notin \mathtt{tt}$.
  - Φ ends with $(\mathtt{ax_p})$. Then $\Phi \triangleright x : [\bar{\mathtt{n}}] \vdash^{(0,0,0)} x : \mathcal{S} \gg (\bar{\mathtt{n}} \times \mathcal{S})$, with $\mathcal{S}$ tight, and the conclusion holds trivially.
  - Φ ends with $(\lambda_\mathtt{p})$. Then $\Phi \triangleright \vdash^{(0,0,0)} \lambda x.t : \mathcal{S} \gg (\mathtt{a} \times \mathcal{S})$, with $\mathcal{S}$ tight, and the conclusion holds trivially.
- Case $t = xu$. Then $u \in \mathtt{no}$, by definition and there are two cases to consider for Φ:
  - If Φ ends with $(@)$. Then $\Phi_u \triangleright \Gamma_u \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}')$, $\Phi_x \triangleright x : \mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa) \vdash^{(b_x,m_x,d_x)} x : \mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa)$, such that $\Gamma = (x : [\mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa)]) + \Gamma_u$ is tight. Absurd, since $\mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa)$ is not tight, therefore this case does not apply.
  - If Φ ends with $(@_\mathtt{p1})$. Then $\Phi_u \triangleright \Gamma_u \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg (\mathtt{tt} \times \mathcal{S}')$, such that $\Gamma = (x : [\mathtt{v}]) + \Gamma_u$ is tight, $b = b_u$, $m = m_u$, $d = d_u + 1$, and $\mathcal{S}'$ is tight. By the *i.h.* on u, we have $b_u = m_u = 0$, therefore $b = m = 0$.

- Case $t = (\lambda x.p)u$. Then $u \in \texttt{ne}$, by definition and there are two cases to consider for Φ:
  - ∗ If Φ ends with $(\texttt{@})$. Then $\Phi_u \triangleright \Gamma_u \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}')$, $\Phi_{\lambda x.p} \triangleright \Gamma_{\lambda x.p} \vdash^{(b_p,m_p,d_p)} \lambda x.p : \mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa)$, such that $\Gamma = \Gamma_u + \Gamma_{\lambda x.p}$ is tight, $b = 1 + b_l + b_u$, $m = m_l + m_u$, $d = d_l + d_m$. Since Γ_u is tight and $u \in \texttt{ne}$, by Lemma 3, $\mathcal{M} \in \texttt{tt}$, which is absurd. Therefore, this case does not apply.
  - ∗ If Φ ends with $(\texttt{@}_{\texttt{p2}})$. Then $\Phi_u \triangleright \Gamma_u \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg (\texttt{n} \times \mathcal{S}')$, such that $\Gamma = \Gamma_u$ is tight, $b = b_u$, $m = m_u$, $d = d_u + 1$ and $\mathcal{S}'$ is tight. By the *i.h.* on u, we have $b_u = m_u = 0$. Therefore $b = m = 0$.

Lemma 20. *Let $\Phi \triangleright \Delta \vdash^{(b,m,d)} s : \mathcal{S}$. If $l \in \texttt{dom}(\mathcal{S})$, then $l \in \texttt{dom}(s)$.*

Proof. We proceed by proving the following stronger version of the statement:

Let $\Phi_s \triangleright \Delta_s \vdash^{(b_s,m_s,d_s)} s : \mathcal{S}_s$. If $l \in \texttt{dom}(\mathcal{S}_s)$, then $s \equiv \texttt{upd}_l(v, q)$, for some value v and store q.

The proof follows by induction on Φ_s:

– Case Φ_s ends with $(\texttt{emp})$. Then the conclusion is vacuously true.
– Case Φ_s ends with $(\texttt{upd})$. Then Φ_s is of the following form:

$$\frac{\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M} \qquad \Phi_q \triangleright \Delta_q \vdash^{(b_q,m_q,d_q)} q : \mathcal{S}_q}{\Gamma_v + \Delta_q \vdash^{(b_v+b_q,m_v+m_q,d_v+v_q)} \texttt{upd}_{l'}(v, q) : \{l' : \mathcal{M}\}; \mathcal{S}_q} \ (\texttt{upd})$$

where $\Delta_s = \Gamma_v + \Delta_q$, $s = \texttt{upd}_{l'}(v, q)$, $\mathcal{S}_s = \{l' : \mathcal{M}\}; \mathcal{S}_q$, $b_s = b_v + b_q$, $m_s = m_v + m_q$, and $d_s = d_v + d_q$. Now we consider two cases:
- Case $l = l'$. Then we are done.
- Case $l \neq l'$. Since we are assuming that $l \in \texttt{dom}(\mathcal{S}_s)$, then it must be case that $l \in \texttt{dom}(\mathcal{S}_q)$. But, then by the *i.h.*, we have $q \equiv \texttt{upd}_l(w, q')$, for some value w and store q'. Therefore, $s \equiv \texttt{upd}_{l'}(v, \texttt{upd}_l(w, q')) \equiv \texttt{upd}_l(w, \texttt{upd}_{l'}(v, q'))$.

The correctness of the original statement now follows easily from the fact that, clearly, if $s \equiv \texttt{upd}_l(v, q)$, then $l \in \texttt{dom}(s)$, by Definition 3.1.

Lemma 21 (Split Lemma).

1. **(Values)** *Let $\Phi_v \triangleright \Gamma \vdash^{(b,m,d)} v : \mathcal{M}$, such that $\mathcal{M} = \sqcup_{i\in I}\mathcal{M}_i$. Then, there exist $(\Phi_v^i \triangleright \Gamma_i \vdash^{(b_i,m_i,d_i)} v : \mathcal{M}_i)_{i\in I}$, such that $\Gamma = +_{i\in I}\Gamma_i$, $b = +_{i\in I}b_i$, $m = +_{i\in I}m_i$, and $d = +_{i\in I}d_i$.*
2. **(States)** *Let $\Phi_s \triangleright \Gamma \vdash^{(b,m,d)} s : \mathcal{S}$, such that $l \in \texttt{dom}(\mathcal{S})$. Then, $s \equiv \texttt{upd}_l(v, q)$, $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{S}(l)$ and $\Phi_q \triangleright \Gamma_q \vdash^{(b_q,m_q,d_q)} q : \mathcal{S}'$, such that $\Gamma = \Gamma_v + \Gamma_q$, $\mathcal{S} = \{(l : \mathcal{S}(l))\}; \mathcal{S}'$, $b = b_v + b_q$, $m = m_v + m_q$, and $d = d_v + d_q$.*

Proof. The proof for values is very similar to the corresponding proof for λ_s, so we are only going to show the split lemma for states. The proof follows by induction on the structure of s:

– Case $s = \epsilon$. Then the statement is vacuously true.

– Case $s = \mathtt{upd}_{l'}(w, q')$. Then Φ_s is of the form:

$$\frac{\Phi_w \triangleright \Gamma_w \vdash^{(b_w,m_w,d_w)} w : \mathcal{M} \qquad \Phi_{q'} \triangleright \Gamma_{q'} \vdash^{(b_{q'},m_{q'},d_{q'})} q' : \mathcal{S}_{q'}}{\Gamma_w + \Gamma_{q'} \vdash^{(b_w+b_{q'},m_w+m_{q'},d_w+d_{q'})} \mathtt{upd}_{l'}(w, q') : \{(l' : \mathcal{M})\}; \mathcal{S}_{q'}} \ (\mathtt{upd})$$

where $\Gamma = \Gamma_w + \Gamma_{q'}$, $\mathcal{S} = \{(l' : \mathcal{M})\}; \mathcal{S}_{q'}$, $b = b_w + b_{q'}$, $m = m_w + m_{q'}$, and $d = d_w + d_{q'}$. We consider two cases:

- Case $l' = l$. Then we simply take $v = w$ and $q = q'$ and we are done.
- Case $l' \neq l$. Since $l \in \mathtt{dom}(\{(l' : \mathcal{M})\}; \mathcal{S}_{q'})$ and $l' \neq l$, then $l \in \mathtt{dom}(\mathcal{S}_{q'})$. By applying the *i.h.* to q', we have that $q' \equiv \mathtt{upd}_l(w', q'')$, $\Phi_{w'} \triangleright \Gamma_{w'} \vdash^{(b_{w'},m_{w'},d_{w'})} w' : \mathcal{S}_{q'}(l)$ and $\Phi_{q''} \triangleright \Gamma_{q''} \vdash^{(b_{q''},m_{q''},d_{q''})} q'' : \mathcal{S}_{q''}$, such that $\Gamma_{q'} = \Gamma_{w'} + \Gamma_{q''}$, $\mathcal{S}_{q'} = \{(l : \mathcal{S}_{q'}(l))\}; \mathcal{S}_{q''}$, $b_{q'} = b_{w'} + b_{q''}$, $m_{q'} = m_{w'} + m_{q''}$, and $d_{q'} = d_{w'} + d_{q''}$. But $s = \mathtt{upd}_{l'}(w, \mathtt{upd}_l(w', q'')) \equiv \mathtt{upd}_l(w', \mathtt{upd}_{l'}(w, q''))$, so we can take $v = w'$, $q = \mathtt{upd}_{l'}(w, q'')$, and consider Φ_q to be the following derivation:

$$\frac{\Phi_w \triangleright \Gamma_w \vdash^{(b_w,m_w,d_w)} w : \mathcal{M} \qquad \Phi_{q''} \triangleright \Gamma_{q''} \vdash^{(b_{q''},m_{q''},d_{q''})} q'' : \mathcal{S}_{q''}}{\Gamma_w + \Gamma_{q''} \vdash^{(b_w+b_{q''},m_w+m_{q''},d_w+d_{q''})} \mathtt{upd}_{l'}(w, q'') : \{(l' : \mathcal{M})\}; \mathcal{S}_{q''}} \ (\mathtt{upd})$$

  where $\Gamma_q = \Gamma_w + \Gamma_{q''}$ and $\mathcal{S}_q = \{(l' : \mathcal{M})\}; \mathcal{S}_{q''}$. We can then conclude with the following observations:
  - $\Gamma_v + \Gamma_q = \Gamma_{w'} + \Gamma_w + \Gamma_{q''} = \Gamma_w + \Gamma_{q'} = \Gamma$,
  - Since $\mathcal{S} = \{(l' : \mathcal{M})\}; \mathcal{S}_{q'}$ and $l' \neq l$, then $\mathcal{S}(l) = \mathcal{S}_{q'}(l)$ and

$$\begin{aligned} \mathcal{S} = \{(l' : \mathcal{M})\}; \mathcal{S}_{q'} &= \{(l' : \mathcal{M})\}; \{(l : \mathcal{S}_{q'}(l))\}; \mathcal{S}_{q''} \\ &= \{(l : \mathcal{S}_{q'}(l))\}; \mathcal{S}_q \\ &= \{(l : \mathcal{S}(l))\}; \mathcal{S}_q \end{aligned}$$

  - $b_v + b_q = b_{w'} + b_w + b_{q''} = b_w + b_{q'} = b$, $m_v + m_q = m_{w'} + m_w + m_{q''} = m_w + m_{q'} = b$ and $d_v + d_q = d_{w'} + d_w + d_{q''} = d_w + d_{q'} = d$.

Lemma 22. *Let $\Phi \triangleright \Gamma \vdash^{(b,m,d)} t : \mathcal{S} \gg (\tau \times \mathcal{S}')$. If $t \in \mathtt{Val}$, then $\tau \neq \mathtt{n}$.*

Proof. By case analysis on the form of $t \in \mathtt{Val}$:

– Case $t = x$. Then we have to consider three cases according to the last rule used in Φ:
  - Case Φ ends with rule $(\mathtt{ax})$, then t can only be assigned σ. Therefore, this case does not apply.
  - Case Φ ends with rule $(\mathtt{m})$, then $\tau = \mathcal{M} \neq \mathtt{n}$.
  - Case Φ ends with rule $(\mathtt{ax_p})$, then $\tau = \overline{\mathtt{n}} \neq \mathtt{n}$.

– Case $t = \lambda x.t$. Then we have to consider three cases according to the last rule used in Φ:
  - Case Φ ends with rule (λ), then t can only be assigned σ. Therefore, this case does not apply.
  - Case Φ ends with rule $(\mathtt{m})$, then $\tau = \mathcal{M} \neq \mathtt{n}$.
  - Case Φ ends with rule $(\lambda_\mathtt{p})$, then $\tau = \mathtt{a} \neq \mathtt{n}$.

Lemma 23. *Let $\Phi \triangleright \Gamma \vdash^{(b,m,d)} t : \mathcal{S} \gg (\tau \times \mathcal{S}')$, such that Γ is tight. If $\tau \in \bar{\mathtt{a}}$, then $\neg\mathtt{abs}(t)$.*

Proof. By induction over Φ:

– Case Φ ends with rule $(\mathtt{ax})$, $(\mathtt{@})$, $(\mathtt{get})$, $(\mathtt{set})$, $(\mathtt{ax_p})$ $(\mathtt{@_{p1}})$, or $(\mathtt{@_{p2}})$, then $\neg\mathtt{abs}(t)$ holds by definition.
– Case Φ ends with rule (λ), $(\mathtt{m})$, or $(\lambda_\mathtt{p})$, then $\tau \in \bar{\mathtt{a}}$ does not hold. Therefore, these cases do not apply.

Completeness (Auxiliary Lemmas)

Lemma 24 (Merge for Values). *Let $(\Phi_v^i \triangleright \Gamma_i \vdash^{(b_i,m_i,d_i)} v : \mathcal{M}_i)_{i\in I}$. Then, there exists $\Phi_v \triangleright \Gamma \vdash^{(b,m,d)} v : \mathcal{M}$, such that $\Gamma = +_{i\in I}\Gamma_i$, $\mathcal{M} = +_{i\in I}\mathcal{M}_i$, $b = +_{i\in I}b_i$, $m = +_{i\in I}m_i$, and $d = +_{i\in I}$.*

We omit this proof given its similarity with the proof for system $\mathcal{O}$.

Lemma 3 (Tight Spreading). *Let $\Phi \triangleright \Gamma \vdash^{(b,m,d)} t : \mathcal{S} \gg (\tau \times \mathcal{S}')$, such that Γ is tight. If $t \in \mathtt{ne}$, then $\tau \in \mathtt{tt}$.*

Proof. We want to show that, if $t \in \mathtt{ne}$, then $\tau \in \mathtt{tt}$, for some $\mathcal{S}'$. We proceed by induction on the predicate $t \in \mathtt{ne}$:

– Case $t = xu$, such that $u \in \mathtt{no}$. Then we have to consider the following two cases depending on the last rule in Φ:
  • Case Φ ends with rule $(\mathtt{@})$, then it must be of the following form:

$$\dfrac{\dfrac{}{x : [\mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa)] \vdash^{(0,0,0)} x : \mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa)}\,(\mathtt{ax}) \qquad \Phi_u \triangleright \Gamma_u \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}')}{(x : [\mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa)]) + \Gamma_u \vdash^{(1+b_u,m_u,d_u)} xu : \mathcal{S} \gg \kappa}\,(\mathtt{@})$$

  where $\Gamma = (x : [\mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa)]) + \Gamma_p$ is tight, $b = 1 + b_u$, $m = m_u$, and $d = d_u$. But $\mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa) \notin \mathtt{tt}$, therefore Γ is not tight and we have a contraction. Thus, this case does not apply.
  • Case Φ ends with rule $(\mathtt{@_{p1}})$, then $\tau = \mathtt{n} \in \mathtt{tt}$, so we can conclude immediately.
– Case $t = (\lambda x.p)u$, such that $u \in \mathtt{ne}$. Then we have to consider the following two cases depending on the last rule in Φ:
  • Case Φ ends with rule $(\mathtt{@})$, then it must be of the following form:

$$\dfrac{\Gamma_p \vdash^{(b_p,m_p,d_p)} \lambda x.p : \mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa) \qquad \Phi_u \triangleright \Gamma_u \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}')}{\Gamma_p + \Gamma_u \vdash^{(1+b_p+b_u,m_p+m_u,d_p+d_u)} (\lambda x.p)u : \mathcal{S} \gg \kappa}\,(\mathtt{@})$$

  where $\Gamma = \Gamma_u + \Gamma_p$ is tight, $b = 1 + b_p + b_u$, $m = m_p + m_u$, and $d = d_p + d_u$. By the *i.h.* on u, we have that $\mathcal{M} \in \mathtt{tt}$, which is a contradiction. Therefore, this case does not apply.
  • Case Φ ends with rule $(\mathtt{@_{p2}})$. Then $\tau = \mathtt{n} \in \mathtt{tt}$, so we can conclude immediately.

Lemma 4 (Typability of States and Normal Forms).

1. *Let s be a state. Then, there exists $\Phi \triangleright \vdash^{(0,0,0)} s : \mathcal{S}$ tight.*
2. *Let $t \in \texttt{no}$. Then for any tight $\mathcal{S}$ there exists $\Phi \triangleright \Gamma \vdash^{(0,0,d)} t : \mathcal{S} \gg (\texttt{tt} \times \mathcal{S})$ tight s.t. $d = |t|$.*

Proof.

1. By induction on s:
   - Case $s = \epsilon$. Then we can build Φ_s as follows:

$$\frac{}{\vdash^{(0,0,0)} \epsilon : \emptyset}\ (\texttt{ax})$$

   And we can conclude with $\mathcal{S} = \emptyset$ tight.
   - Case $s = \texttt{upd}_l(v, q)$. By the *i.h.*, there exists $\Phi_q \triangleright \Delta_q \vdash^{(0,0,0)} q : \mathcal{S}_q$ tight. Therefore, we can build Φ_s as follows:

$$\frac{\dfrac{}{\vdash^{(0,0,0)} v : [\,]}\ (\texttt{m}) \qquad \Phi_q \triangleright \vdash^{(0,0,0)} q : \mathcal{S}_q}{\vdash^{(0,0,0)} \texttt{upd}_l(v, q) : \{(l : [\,])\}; \mathcal{S}_q}\ (\texttt{upd})$$

   And we can conclude with $\mathcal{S} = \{(l : [\,])\}; \mathcal{S}_q$ tight.
2. By simultaneous induction on the following claims:
   (a) If $t \in \texttt{ne}$, then $\Phi \triangleright \Gamma \vdash^{(0,0,d)} t : \mathcal{S} \to \texttt{n} \times \mathcal{S}$ tight, such that $d = |t|$, for any tight $\mathcal{S}$.
   (b) If $t \in \texttt{no}$, then $\Phi \triangleright \Gamma \vdash^{(0,0,d)} t : \mathcal{S} \to \texttt{tt} \times \mathcal{S}$ tight, such that $d = |t|$, for any tight $\mathcal{S}$.
   (a) By induction on $t \in \texttt{ne}$:
   - Case $t = xu$, such that $u \in \texttt{no}$. By the *i.h.* (Lemma 4.2b), we have $\Phi_u \triangleright \Gamma_u \vdash^{(0,0,d_u)} u : \mathcal{S} \gg (\texttt{tt} \times \mathcal{S})$ tight, such that $d_u = |u|$, for any tight $\mathcal{S}$. Therefore, we can build Φ is as follows:

$$\frac{\Gamma_u \vdash^{(0,0,d_u)} u : \mathcal{S} \gg (\texttt{tt} \times \mathcal{S})}{(x : [\texttt{v}]) + \Gamma_u \vdash^{(0,0,1+d_u)} xu : \mathcal{S} \gg (\texttt{n} \times \mathcal{S})}\ (@_{\texttt{p1}})$$

   And we can conclude with $\Gamma = (x : [\texttt{v}]) + \Gamma_u$ tight and $d = 1 + d_u = 1 + |u| = 1 + |x| + |u| = |xu|$.
   - Case $t = (\lambda x.p)u$, such that $u \in \texttt{ne}$. By the *i.h.* (Lemma 4.2a), we have $\Phi_u \triangleright \Gamma \vdash^{(0,0,d_u)} u : \mathcal{S} \gg (\texttt{n} \times \mathcal{S})$ tight, such that $d_u = |u|$, for any tight $\mathcal{S}$. Therefore, we can build Φ is as follows:

$$\frac{\Phi_u \triangleright \Gamma \vdash^{(0,0,d_u)} u : \mathcal{S} \gg (\texttt{n} \times \mathcal{S})}{\Gamma_u \vdash^{(0,0,1+d_u)} (\lambda x.p)u : \mathcal{S} \gg (\texttt{n} \times \mathcal{S})}\ (@_{\texttt{p2}})$$

   And we can conclude with $d = 1 + d_u = 1 + |u| = 1 + |\lambda x.p| + |u| = 1 + |(\lambda x.p)u|$.
   (b) By induction on $t \in \texttt{no}$:
   - Case $t \in \texttt{Val}$:

- Assume $t = x$. Then we can build Φ_t as follows:

$$\frac{}{x : [\overline{\mathtt{n}}] \vdash^{(0,0,0)} x : \mathcal{S} \gg (\overline{\mathtt{n}} \times \mathcal{S})} \; (\mathtt{ax_p})$$

for any $\mathcal{S}$ tight. And we can conclude with $\Gamma = (x : [\overline{\mathtt{n}}])$ tight, and $b = m = d = 0 = |x|$.

- Assume $t = \lambda x.u$. Then we can build Φ_t as follows:

$$\frac{}{\vdash^{(0,0,0)} \lambda x.u : \mathcal{S} \gg (\mathtt{a} \times \mathcal{S})} \; (\lambda_{\mathtt{p}})$$

for any $\mathcal{S}$ tight. And we can conclude with $\Gamma = \emptyset$ tight, $b = m = d = 0 = |\lambda x.u|$.

– Case $t \notin \mathtt{Val}$. Then $t \in \mathtt{ne}$, and this case is subsumed by the previous cases.

Soundness and Completeness (Main Lemmas)

Lemma 5 (Substitution and Anti-Substitution).

1. **(Substitution)** *If $\Phi_t \triangleright \Gamma_t; x : \mathcal{M} \vdash^{(b_t,m_t,d_t)} t : \delta$ and $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M}$, then $\Phi_{t\{x\backslash v\}} \triangleright \Gamma_t + \Gamma_v \vdash^{(b_t+b_v,m_t+m_v,d_t+d_v)} t\{x\backslash v\} : \delta$.*
2. **(Anti-Substitution)** *If $\Phi_{t\{x\backslash v\}} \triangleright \Gamma_{t\{x\backslash v\}} \vdash^{(b,m,d)} t\{x\backslash v\} : \delta$, then $\Phi_t \triangleright \Gamma_t; x : \mathcal{M} \vdash^{(b_t,m_t,d_t)} t : \delta$ and $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M}$, such that $\Gamma_{t\{x\backslash v\}} = \Gamma_t + \Gamma_v$, $b = b_t + b_v$, $m = m_t + m_v$, and $d = d_t + d_v$.*

Proof.

1. We are going to generalize the original statement by replacing δ with $\mathcal{T}$.

   The proof now follows by induction over the structure of Φ_t:

   – Case Φ_t ends with rule $(\mathtt{ax})$. Then t must be a variable and we must consider two cases:
     - Assume $t = y = x$. Then $\Gamma_t = \emptyset$, $\mathcal{T} = \mathcal{M}$, $t\{x\backslash v\} = v$, $b_t = m_t = d_t = 0$. So we can take $\Phi_{t\{x\backslash v\}} = \Phi_v$ and conclude with $\Gamma_t + \Gamma_v = \Gamma_v$, $b_t + b_v = b_v$, $m_t + m_v = m_v$, and $d_t + d_v = d_v$.
     - Assume $t = y \neq x$. Then $\mathcal{M} = [\,]$, $\Gamma_v = \emptyset$, $t\{x\backslash v\} = t$, $b_v = 0$, $m_v = 0$, and $d_v = 0$. So we can take $\Phi_{t\{x\backslash v\}} = \Phi_t$ and conclude with $\Gamma_t + \Gamma_v = \Gamma_t$, $b_t + b_v = b_t$, $m_t + m_v = m_t$, and $d_t + d_v = d_t$.

   – Case Φ_t ends with (λ). Then t must be of the form $\lambda y.u$ and Φ_t must be of the following form (by α-conversion):

$$\frac{\Phi_u \triangleright \Gamma; x : \mathcal{M} \vdash^{(b_t,m_t,d_t)} u : \mathcal{S} \gg \kappa}{(\Gamma \setminus\!\!\setminus y); x : \mathcal{M} \vdash^{(b_t,m_t,d_t)} \lambda y.u : \Gamma(y) \Rightarrow (\mathcal{S} \gg \kappa)} \; (\lambda)$$

   where $\Gamma_t = (\Gamma \setminus\!\!\setminus y)$, and $\mathcal{T} = \Gamma(y) \Rightarrow (\mathcal{S} \gg \kappa)$. By the *i.h.*, we have the following derivation $\Phi_{u\{x\backslash v\}} \triangleright \Gamma + \Gamma_v \vdash^{(b_t+b_v,m_t+m_v,d_t+d_v)} u\{x\backslash v\} : \mathcal{S} \gg \kappa$. Therefore, we can build $\Phi_{t\{x\backslash v\}}$ as follows:

$$\frac{\Phi_{u\{x\backslash v\}} \triangleright \Gamma + \Gamma_v \vdash^{(b_t+b_v,m_t+m_v,d_t+d_v)} u\{x\backslash v\} : \mathcal{S} \gg \kappa}{(\Gamma + \Gamma_v) \setminus\!\!\setminus y \vdash^{(b_t+b_v,m_t+m_v,d_t+d_v)} \lambda y.u\{x\backslash v\} : \Gamma(y) \Rightarrow (\mathcal{S} \gg \kappa)} \; (\lambda)$$

And we conclude with $(\Gamma + \Gamma_v) \setminus\!\!\setminus y = (\Gamma \setminus\!\!\setminus y) + \Gamma_v = \Gamma_t + \Gamma_v$, by α-conversion.

- Case Φ_t ends with ($\mathtt{@}$). Then t must be of the form wu and Φ_t must be of following form:

$$\frac{\Phi_w \triangleright \Gamma; x : \mathcal{M}_1 \vdash^{(b_w,m_w,d_w)} w : \mathcal{M}' \Rightarrow (\mathcal{S}' \gg \kappa) \qquad \Phi_u \triangleright \Delta; x : \mathcal{M}_2 \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg (\mathcal{M}' \times \mathcal{S}')}{\Gamma + \Delta; x : \mathcal{M}_1 \sqcup \mathcal{M}_2 \vdash^{(1+b_w+b_u,m_w+m_u,d_w+d_u)} wu : \mathcal{S} \gg \kappa} (\mathtt{@})$$

such that $\Gamma_t = \Gamma + \Delta$, $\mathcal{M} = \mathcal{M}_1 \sqcup \mathcal{M}_2$, $\mathcal{T} = \mathcal{S} \gg \kappa$, $b_t = 1 + b_w + b_u$, $m_t = m_w + m_u$, and $d_t = d_w + d_u$. By Lemma 21.1, we know there exist the following derivations $(\Phi^i_v \triangleright \Gamma^i_v \vdash^{(b_i,m_i,d_i)} v : \mathcal{M}_i)_{i\in\{1,2\}}$, such that $\Gamma_v = \Gamma^1_v + \Gamma^2_v$, $b_v = b_1 + b_2$, $m_v = m_1 + m_2$, and $d_v = d_1 + d_2$. By the *i.h.*, we know there exist $\Phi_{w\{x\backslash v\}} \triangleright \Gamma + \Gamma^1_v \vdash^{(b_w+b_1,m_w+m_1,d_w+d_1)} w\{x\backslash v\} : \mathcal{M}' \Rightarrow (\mathcal{S}' \gg \kappa)$ and $\Phi_{u\{x\backslash v\}} \triangleright \Delta + \Gamma^2_v \vdash^{(b_u+b_2,m_u+m_2,d_u+d_2)} u\{x\backslash v\} : \mathcal{S} \gg (\mathcal{M}' \times \mathcal{S}')$. We can build $\Phi_{t\{x\backslash v\}}$ as follows:

$$\frac{\Phi_{w\{x\backslash v\}} \qquad \Phi_{u\{x\backslash v\}}}{(\Gamma + \Delta) + (\Gamma^1_v + \Gamma^2_v) \vdash^{(1+b_w+b_u+b_1+b_2,m_w+m_u+m_1+m_2,d_w+d_u+d_1+d_2)} (wu)\{x\backslash v\} : \mathcal{S} \gg \kappa} (\mathtt{@})$$

And we can conclude with $\Gamma_t+\Gamma_v = (\Gamma+\Delta)+(\Gamma^1_v+\Gamma^2_v)$, $b_t+b_v = 1+b_w+b_u+b_1+b_2$, $m_t+m_v = m_w+m_u+m_1+m_2$, and $d_t+d_v = d_w+d_u+d_1+d_2$.

- Case Φ_w ends with ($\mathtt{m}$). Then t must be of the form w and Φ_t must be of the following form:

$$\frac{(\Phi^i_w \triangleright \Gamma_i; x : \mathcal{M}_i \vdash^{(b_i,m_i,d_i)} w : \sigma_i)_{i\in I}}{+_{i\in I}\Gamma_i; x : \sqcup_{i\in I}\mathcal{M}_i \vdash^{(+_{i\in I}b_i,+_{i\in I}m_i,+_{i\in I}d_i)} w : [\sigma_i]_{i\in I}} (\mathtt{m})$$

such that $\Gamma_t = +_{i\in I}\Gamma_i$, $\mathcal{T} = [\sigma_i]_{i\in I}$, $b_t = +_{i\in I}b_i$, $m_t = +_{i\in I}m_i$, and $d_t = +_{i\in I}d_i$. By Lemma 21.1, $(\Phi^i_v \triangleright \Gamma^i_v \vdash^{(b^i_v,m^i_v,d^i_v)} v : \mathcal{M}_i)_{i\in I}$, such that $\Gamma_v = +_{i\in I}\Gamma^i_v$, $b_v = +_{i\in I}b^i_v$, $m_v = +_{i\in I}m^i_v$, and $d_v = +_{i\in I}d^i_v$. By the *i.h.* over each Φ^i_v, we have $(\Phi^i_{w\{x\backslash v\}} \triangleright \Gamma_i + \Gamma^i_v \vdash^{(b_i+b^i_v,m_i+m^i_v,d_i+d^i_v)} w\{x\backslash v\} : \sigma_i)_{i\in I}$. Therefore, we can build $\Phi_{t\{x\backslash v\}}$ as follows:

$$\frac{(\Phi^i_{w\{x\backslash v\}} \triangleright \Gamma_i + \Gamma^i_v \vdash^{(b_i+b^i_v,m_i+m^i_v,d_i+d^i_v)} w\{x\backslash v\} : \sigma_i)_{i\in I}}{+_{i\in I}(\Gamma^i_v + \Gamma^i_w) \vdash^{(+_{i\in I}(b_i+b^i_v),+_{i\in I}(m_i+m^i_v),+_{i\in I}(d_i+d^i_v))} w\{x\backslash v\} : [\tau_i]_{i\in I}} (\mathtt{m})$$

And we can conclude with $\Gamma_t + \Gamma_v = +_{i\in I}\Gamma_i +_{i\in I} \Gamma^i_v = +_{i\in I}(\Gamma_i + \Gamma^i_v)$, $b_t + b_v = +_{i\in I}b_i +_{i\in I} b^i_v = +_{i\in I}(b_i + b^i_v)$, $m_t + m_v = +_{i\in I}m_i +_{i\in I} m^i_v = +_{i\in I}(m_i + m^i_v)$, and $d_t + d_v = +_{i\in I}d_i +_{i\in I} d^i_v = +_{i\in I}(d_i + d^i_v)$.

- Case Φ_t ends with ($\uparrow$). Then t is a variable and Φ_t must be of the following form:

$$\frac{\Phi_w \triangleright \Gamma; x : \mathcal{M} \vdash^{(b_t,m_t,d_t)} w : \mathcal{M}'}{\Gamma; x : \mathcal{M} \vdash^{(b_t,m_t,d_t)} w : \mathcal{S} \gg (\mathcal{M}' \times \mathcal{S})} (\uparrow)$$

where $\mathcal{T} = \mathcal{S} \gg (\mathcal{M}' \times \mathcal{S})$. By the *i.h.*, we have $\Phi_{w\{x\backslash v\}} \triangleright \Gamma + \Gamma_v \vdash^{(b_t+b_v,m_t+m_v,d_t+d_v)} w\{x\backslash v\} : \mathcal{M}'$. Therefore, we can build $\Phi_{t\{x\backslash v\}}$ as follows:

$$\frac{\Phi_{w\{x\backslash v\}} \triangleright \Gamma + \Gamma_v \vdash^{(b_t+b_v, m_t+m_v, d_t+d_v)} w\{x\backslash v\} : \mathcal{M}'}{\Gamma + \Gamma_v \vdash^{(b_t+b_v, m_t+m_v, d_t+d_v)} w\{x\backslash v\} : \mathcal{S} \gg (\mathcal{M}' \times \mathcal{S})} \ (\uparrow)$$

And we can conclude.

- Case Φ_t ends with $(\mathtt{get})$. Then t must be of the form $\mathtt{get}_l(\lambda y.u)$ and Φ_t must be of the following form:

$$\frac{\Phi_u \triangleright \Gamma_u; x : \mathcal{M} \vdash^{(b_u, m_u, d_u)} u : \mathcal{S} \gg \kappa}{(\Gamma_u \setminus\!\!\setminus y); x : \mathcal{M} \vdash^{(b_u, 1+m_u, d_u)} \mathtt{get}_l(\lambda y.u) : \{(l : \Gamma_u(y))\} \uplus \mathcal{S} \gg \kappa} \ (\mathtt{get})$$

where $\mathcal{T} = \{(l : \Gamma_u(y))\} \uplus \mathcal{S} \gg \kappa$, $\Gamma_t = \Gamma_u \setminus\!\!\setminus y$, $b_t = b_u$, $m_t = 1 + m_u$, and $d_t = d_u$. By the *i.h.*, we have $\Phi_{u\{x\backslash v\}} \triangleright \Gamma_u + \Gamma_v \vdash^{(b_u+b_v, m_u+m_v, d_u+d_v)} u\{x\backslash v\} : \mathcal{S} \to \kappa$. Therefore, we can build $\Phi_{t\{x\backslash v\}}$ as follows:

$$\frac{\Phi_{u\{x\backslash v\}} \triangleright \Gamma_u + \Gamma_v \vdash^{(b_u+b_v, m_u+m_v, d_u+d_v)} u\{x\backslash v\} : \mathcal{S} \gg \kappa}{(\Gamma_u + \Gamma_v) \setminus\!\!\setminus y \vdash^{(b_u+b_v, 1+m_u+m_v, d_u+d_v)} \mathtt{get}_l(\lambda y.u)\{x\backslash v\} : \{(l : \Gamma_u(y))\} \uplus \mathcal{S} \gg \kappa} \ (\mathtt{get})$$

And we can conclude with $(\Gamma_u + \Gamma_v) \setminus\!\!\setminus y = (\Gamma \setminus\!\!\setminus y) + \Gamma_v = \Gamma_t + \Gamma_v$ by α-conversion, $b_t+b_v = b_u+b_v$, $m_t+m_v = 1+m_u+m_v$, and $d_t+d_v = d_u+d_v$.

- Case Φ_t ends with $(\mathtt{set})$. Then t must be of the form $\mathtt{set}_l(w, u)$ and Φ_t must be of the following form:

$$\frac{\Phi_w \triangleright \Gamma_w; x : \mathcal{M}_1 \vdash^{(b_w, m_w, d_w)} w : \mathcal{M}' \qquad \Phi_u \triangleright \Gamma_u; x : \mathcal{M}_2 \vdash^{(b_u, m_u, d_u)} u : \{(l : \mathcal{M}')\}; \mathcal{S} \gg \kappa}{\Gamma_w + \Gamma_u; x : \mathcal{M}_1 \sqcup \mathcal{M}_2 \vdash^{(b_w+b_u, 1+m_w+m_u, d_w+d_u)} \mathtt{set}_l(w, u) : \mathcal{S} \gg \kappa} \ (\mathtt{set})$$

where $\mathcal{T} = \mathcal{S} \gg \kappa$, $\Gamma_t = \Gamma_w + \Gamma_u$, $\delta = \mathcal{S} \gg \kappa$, $b_t = b_w + b_u$, $m_t = 1 + m_w + m_u$, and $d_t = d_w + d_u$. By Lemma 21.1, we have $\Phi_v^1 \triangleright \Gamma_v^1 \vdash^{(b_v^1, m_v^1, d_v^1)} v : \mathcal{M}_1$ and $\Phi_v^2 \triangleright \Gamma_v^2 \vdash^{(b_v^2, m_v^2, d_v^2)} v : \mathcal{M}_2$, such that $\Gamma_v = \Gamma_v^1 + \Gamma_v^2$, $b_v = b_v^1 + b_v^2$, $m_v = m_v^1 + m_v^2$, and $d_v = d_v^1 + d_v^2$. By the *i.h.*, we have $\Phi_{w\{x\backslash v\}} \triangleright \Gamma_w + \Gamma_v^1 \vdash^{(b_w+b_v^1, m_w+m_v^1, d_w+d_v^1)} w\{x\backslash v\} : \mathcal{M}'$ and $\Phi_{u\{x\backslash v\}} \triangleright \Gamma_u + \Gamma_v^2 \vdash^{(b_u+b_v^2, m_u+m_u^2, d_u+d_v^2)} u\{x\backslash v\} : \{(l : \mathcal{M}')\}; \mathcal{S} \gg \kappa$. Assume $\Phi_{w\{x\backslash v\}} \triangleright \Gamma_w + \Gamma_v^1 \vdash^{(b_w+b_v^1, m_w+m_v^1, d_w+d_v^1)} w\{x\backslash v\} : \mathcal{M}'$ and $\Phi_{u\{x\backslash v\}} \triangleright \Gamma_u + \Gamma_v^2 \vdash^{(b_u+b_v^2, m_u+m_u^2, d_u+d_v^2)} u\{x\backslash v\} : \{(l : \mathcal{M}')\}; \mathcal{S} \gg \kappa$. We can build $\Phi_{t\{x\backslash v\}}$ as follows:

$$\frac{\Phi_{w\{x\backslash v\}} \qquad \Phi_{u\{x\backslash v\}}}{(\Gamma_w + \Gamma_u) + (\Gamma_v^1 + \Gamma_v^2) \vdash^{(b_w+b_u+b_v^1+b_v^2, 1+m_w+m_u+m_v^1+m_v^2, d_w+d_u+d_v^1+d_v^2)} (wu)\{x\backslash v\} : \mathcal{S} \gg \kappa} \ (\mathtt{set})$$

And we can conclude with $\Gamma_t+\Gamma_v = (\Gamma_w+\Gamma_u)+(\Gamma_v^1+\Gamma_v^2)$, $b_t+b_v = b_w+b_u+b_v^1+b_v^2$, $m_t+m_v = 1+m_w+m_u+m_v^1+m_v^2$, $d_t+d_v = d_w+d_u+d_v^1+d_v^2$.

- Case Φ_t ends with $(\mathtt{ax_p})$. Then t must be a variable and we must consider two cases:

- Assume $t = y = x$. Then $\Gamma_t = \emptyset$, $\mathcal{T} = \mathcal{S} \Rightarrow (\bar{\mathtt{n}} \gg \mathcal{S})$, $t\{x \backslash v\} = v$, $b_t = m_t = d_t = 0$. Moreover, $\mathcal{M} = [\bar{\mathtt{n}}]$. We have to consider two cases:
  * Case $v = z$. Then $\Phi_v \triangleright z : [\bar{\mathtt{n}}] \vdash^{(0,0,0)} z : [\bar{\mathtt{n}}]$. So we can take $\Phi_{t\{x\backslash v\}}$ as the following derivation:

$$\frac{}{z : [\bar{\mathtt{n}}] \vdash^{(0,0)} z : \mathcal{S} \gg (\bar{\mathtt{n}} \times \mathcal{S})}\ (\mathtt{ax_p})$$

    and conclude with $\Gamma_t + \Gamma_v = \Gamma_v = (z : [\bar{\mathtt{n}}])$, $b_t + b_v = b_v = 0$, $m_t + m_v = m_v = 0$, and $d_t + d_v = d_v$.
  * Case $v = \lambda z.p$. This case does not apply, by Lemma 23.
- Assume $t = y \neq x$. Then $\mathcal{M} = [\,]$, $\Gamma_v = \emptyset$, $t\{x \backslash v\} = t$, $b_v = 0$, $m_v = 0$, and $d_v = 0$. So we can take $\Phi_{t\{x\backslash v\}} = \Phi_t$ and conclude with $\Gamma_t + \Gamma_v = \Gamma_t$, $b_t + b_v = b_t$, $m_t + m_v = m_t$, and $d_t + d_v = d_t$.

– Case Φ_t ends with $(\lambda_{\mathtt{p}})$. Then t is of the form $\lambda y.u$, $\Gamma_t = \emptyset$, $\mathcal{T} = \mathcal{S} \gg (\mathtt{a} \times \mathcal{S})$, $\mathcal{M} = [\,]$, $\Gamma_v = \emptyset$, $t\{x \backslash v\} = \lambda y.(u\{x \backslash v\}) = (\lambda y.u)\{x \backslash v\}$, $b_t = b_v = 0$, $m_t = m_v = 0$, and $d_t = d_v = 0$. So we can build $\Phi_{t\{x\backslash v\}}$ as follows:

$$\frac{}{\vdash^{(0,0,0)} (\lambda y.u)\{x \backslash v\} : \mathcal{S} \gg (\mathtt{a} \times \mathcal{S})}\ (\lambda_{\mathtt{p}})$$

And conclude with $\Gamma_t + \Gamma_v = \emptyset$, $b_t = b_v = 0$, $m_t = m_v = 0$, and $d_t = d_v = 0$.

– Case Φ_t ends with $(@_{\mathtt{p1}})$. Then t is of the form yu and we have to consider two cases:
- Case $y = x$. Then Φ_t must be of the following form:

$$\frac{\Gamma_u \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg (\mathtt{tt} \times \mathcal{S}')}{(x : [\mathtt{v}] \sqcup \Gamma_u(x)); (\Gamma_u \setminus\!\!\setminus x) \vdash^{(b_u,m_u,1+d_u)} xu : \mathcal{S} \gg (\mathtt{n} \times \mathcal{S}')}\ (@_{\mathtt{p1}})$$

  such that $\Gamma_t = (\Gamma_u \setminus\!\!\setminus x)$, $b = b_u$, $m = m_u$, and $d = 1 + d_u$. Then $\mathcal{M} = [\mathtt{v}] \sqcup \Gamma_u(x)$ and, by Lemma 21.1, we have $\Phi^1_v \triangleright \Gamma^1_v \vdash^{(b^1_v,m^1_v,d^1_v)} v : [\mathtt{v}]$ and $\Phi^2_v \triangleright \Gamma^2_v \vdash^{(b^2_v,m^2_v,d^2_v)} v : \Gamma_u(x)$, such that $\Gamma_v = \Gamma^1_v + \Gamma^2_v$, $b_v = b^1_v + b^2_v$, $m_v = m^1_v + m^2_v$, and $d_v = d^1_v + d^2_v$. By the *i.h.*, we know there exists $\Phi_{u\{x\backslash v\}} \triangleright (\Gamma_u \setminus\!\!\setminus x) + \Gamma^2_u \vdash^{(b_u+b^2_v,m_u+m^2_v,d_u+d^2_v)} u\{x \backslash v\} : \mathcal{S} \gg (\mathtt{tt} \times \mathcal{S}')$. Now, we need to consider two cases:
  * Case $v = z$. Then $\Phi^1_v \triangleright z : [\mathtt{v}] \vdash^{(0,0,0)} z : [\mathtt{v}]$ and $\Phi^2_v \triangleright z : \Gamma_u(x) \vdash^{(0,0)} z : \Gamma_u(x)$. Therefore, we can build $\Phi_{t\{x\backslash v\}} = \Phi_{v\{x\backslash v\}}$ as follows:

$$\frac{\Phi_{u\{x\backslash v\}} \triangleright (\Gamma_u \setminus\!\!\setminus x) + \Gamma_u(x) \vdash^{(b_u+b^2_v,m_u+m^2_v,d_u+d^2_v)} u\{x \backslash v\} : \mathcal{S} \gg (\mathtt{tt} \times \mathcal{S}')}{(z : [\mathtt{v}]) + (\Gamma_u \setminus\!\!\setminus x + (z : \Gamma_u(x))) \vdash^{(b_u+b^2_v,m_u+m^2_v,1+d_u+d^2_v)} z(u\{x \backslash v\}) : \mathcal{S} \gg (\mathtt{n} \times \mathcal{S}')}\ (@_{\mathtt{p1}})$$

    where $(z : [\mathtt{v}]) + (\Gamma_u \backslash x + (z : \Gamma_u(x))) = (\Gamma_u \backslash x) + (z : [\mathtt{v}] \cup \Gamma_u(x)) = \Gamma_u + \Gamma_v$, $b_u + b^2_v = b + b^1_v + b^2_v = b + b_v$, $m_u + m^2_v = m + m^1_v + m^2_v = m + m_v$, and $d_u + d^2_v = d + d^1_v + d^2_v = d + d_v$.
  * Case $v = \lambda z.p$. This case does not apply, since it is not possible to assign $\mathtt{v}$ to $\lambda z.p$, by Lemma 23.

    - Case $y \neq x$. Then, the proof is very similar to when Φ_t ends with rule ($\mathtt{@}$).
  - Case Φ_t ends with ($\mathtt{@_{p2}}$), the proof is very similar to when Φ_t ends with rule ($\mathtt{@_{p1}}$).
2. We are going to generalize the original statement by replacing δ with $\mathcal{T}$.

   The proof follows by induction over t:
  - Case $t = y$. Then we have to consider two cases:
    - Let $t = y \neq x$. Then $t\{x\backslash v\} = y$. Let $\Gamma_v = \emptyset$, $\mathcal{M} = [\,]$, $b_v = m_v = d_v = 0$. Then, Φ_v is derivable using rule ($\mathtt{m}$) with no premise. We also take $\Phi_t = \Phi_{t\{x\backslash v\}}$, so that, in particular $\Gamma_t = \Gamma_{t\{x\backslash v\}}$. Then, we can conclude with $\Gamma_{t\{x\backslash v\}} = \Gamma_t + \Gamma_v = \Gamma_t$, $b = b_t + b_v = b_t$, $m = m_t + m_v = m_t$, and $d = d_t + d_v = d_t$.
    - Let $t = y = x$. Then $t\{x\backslash v\} = v$. Let $\Gamma_t = \emptyset$, and $b_t = m_t = s_t = 0$. Now we will consider two cases depending on the form of v:
      * Case $v = z$. Then $t\{x\backslash v\} = z$ and we can proceed by case analysis of the last rule in $\Phi_{t\{x\backslash v\}}$. In all of them, we can build Φ_t from $\Phi_{t\{x\backslash v\}}$, by simply replacing x with z, and Φ_v as follows:

$$\dfrac{\dfrac{}{z : [\sigma] \vdash^{(0,0,0)} z : \sigma}\ (\mathtt{ax})}{z : [\sigma] \vdash^{(0,0,0)} z : [\sigma]}\ (\mathtt{m})$$

        And we can conclude since all the counters are zero.
      * Case $v = \lambda z.p$. Then $t\{x\backslash v\} = \lambda z.p$ and we can proceed by case analysis of the last rule in $\Phi_{t\{x\backslash v\}}$. In all of them, we can always build Φ_t using either ($\mathtt{ax}$) (case ($\mathtt{@}$)), ($\mathtt{ax_p}$) (case ($\lambda_\mathtt{p}$)), ($\mathtt{ax}$) plus ($\mathtt{m}$) (case ($\mathtt{m}$)), or ($\mathtt{ax}$) plus ($\mathtt{m}$) plus ($\uparrow$) (case ($\uparrow$)). Φ_v is either $\Phi_{t\{x\backslash v\}}$ (case ($\mathtt{m}$)), or it can be built from $\Phi_{t\{x\backslash v\}}$ plus rule ($\mathtt{m}$) (all other cases).
  - Case $t = \lambda y.u$. Then $t\{x\backslash v\} = (\lambda y.u)\{x\backslash v\} = \lambda y.(u\{x\backslash v\})$ and we must consider three cases:
    - Case $\Phi_{t\{x\backslash v\}}$ ends with rule (λ), then it must be of the following form:

$$\dfrac{\Phi_{u\{x\backslash v\}} \triangleright \Gamma_{u\{x\backslash v\}}; y : \mathcal{M}' \vdash^{(b,m,d)} u\{x\backslash v\} : \mathcal{S} \gg \kappa}{\Gamma_{u\{x\backslash v\}} \vdash^{(b,m,d)} \lambda y.(u\{x\backslash v\}) : \mathcal{M}' \Rightarrow (\mathcal{S} \gg \kappa)}\ (\lambda)$$

      where $\mathcal{T} = \mathcal{M}' \Rightarrow (\mathcal{S} \gg \kappa)$ and $\Gamma_{t\{x\backslash v\}} = \Gamma_{u\{x\backslash v\}}$. By the *i.h.*, we have $\Phi_u \triangleright \Gamma_u; y : \mathcal{M}'; x : \mathcal{M} \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg \kappa$ and $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M}$, such that $\Gamma_{u\{x\backslash v\}} = \Gamma_u + \Gamma_v$, $b = b_u + b_v$, $m = m_u + m_v$, and $d = d_u + d_v$. So we can build $\Phi_{\lambda y.u}$ as follows:

$$\dfrac{\Phi_u \triangleright \Gamma_u; y : \mathcal{M}'; x : \mathcal{M} \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg \kappa}{\Gamma_u; x : \mathcal{M} \vdash^{(b_u,m_u,d_u)} \lambda y.u : \mathcal{M}' \Rightarrow (\mathcal{S} \gg \kappa)}\ (\lambda)$$

      And we can pick $\Phi_t = \Phi_{\lambda y.u}$, and conclude with $\Gamma_{t\{x\backslash v\}} = \Gamma_{u\{x\backslash v\}} = \Gamma_u + \Gamma_v$, $b = b_u + b_v$, $m = m_u + m_v$, and $d = d_u + d_v$.

- Case $\Phi_{t\{x\backslash v\}}$ ends with rule $(\lambda_{\mathtt{p}})$. Then it must be of the following form:

$$\frac{}{\vdash^{(0,0,0)} \lambda y.(u\{x\backslash y\}) : \mathcal{S} \gg (\mathtt{a} \times \mathcal{S})}\ (\lambda_{\mathtt{p}})$$

where $\Gamma_{t\{x\backslash v\}} = \emptyset$, $\mathcal{T} = \mathcal{S} \gg (\mathtt{a} \times \mathcal{S})$, and $b = m = d = 0$. Let $\Gamma_t = \emptyset$, $\mathcal{M} = [\,]$, and $b_t = m_t = d_t = 0$. Then, we can construct Φ_t as follows:

$$\frac{}{\vdash^{(0,0,0)} \lambda y.u : \mathcal{S} \gg (\mathtt{a} \times \mathcal{S})}\ (\lambda_{\mathtt{p}})$$

Let $\Gamma_v = \emptyset$, and $b_v = m_v = d_v = 0$. Then Φ_v can be constructed by using rule $(\mathtt{m})$ with no premises. So we can conclude with $\Gamma_{t\{x\backslash v\}} = \emptyset = \Gamma_t + \Gamma_v$, and $b = 0 = b_t + b_v$, $m = 0 = m_t + m_v$, and $d = 0 = d_t + d_v$.

- Case $\Phi_{t\{x\backslash v\}}$ ends with rule $(\mathtt{m})$. Then $t\{x\backslash v\}$ is a value, and $\Phi_{t\{x\backslash v\}}$ must be of the following form:

$$\frac{(\Phi_i \triangleright \Gamma_i \vdash^{(b_i,m_i,d_i)} t\{x\backslash v\} : \sigma_i)_{i\in I}}{+_{i\in I}\Gamma_i \vdash^{(+_{i\in I}b_i,+_{i\in I}m_i,+_{i\in I}d_i)} t\{x\backslash v\} : [\sigma_i]_{i\in I}}\ (\mathtt{m})$$

where $\mathcal{T} = [\sigma_i]_{i\in I}$, $\Gamma_{t\{x\backslash v\}} = +_{i\in I}\Gamma_i$, $b = +_{i\in I}b_i$, $m = +_{i\in I}m_i$, and $d = +_{i\in I}d_i$. By the *i.h.* over each Φ_i, we have the following derivations $\Phi_t^i \triangleright \Gamma_t^i; x : \mathcal{M}_i \vdash^{(b_t^i,m_t^i,d_t^i)} t : \sigma_i$ and $\Phi_v^i \triangleright \Gamma_v^i \vdash^{(b_v^i,m_v^i,d_v^i)} v : \mathcal{M}_i$, such that $\Gamma_i = \Gamma_t^i + \Gamma_v^i$, $b = b_t^i + b_v^i$, $m = m_t^i + m_v^i$, and $d = d_t^i + d_v^i$, for each $i \in I$. So we can construct Φ_t as follows:

$$\frac{(\Phi_t^i \triangleright \Gamma_t^i; x : \mathcal{M}_i \vdash^{(b_t^i,m_t^i,d_t^i)} t : \sigma_i)_{i\in I}}{+_{i\in I}\Gamma_t^i; x : \sqcup_{i\in I}\mathcal{M}_i \vdash^{(+_{i\in I}b_t^i,+_{i\in I}m_t^i,+_{i\in I}d_t^i)} t : [\sigma_i]_{i\in I}}\ (\mathtt{m})$$

such that $\Gamma_t = +_{i\in I}\Gamma_t^i$, $\mathcal{M} = \sqcup_{i\in I}\mathcal{M}_i$, $b_t = +_{i\in I}b_t^i$, $m_t = +_{i\in I}m_t^i$, and $d_t = +_{i\in I}d_t^i$. By Lemma 24, we can take the following derivation $\Phi_v \triangleright +_{i\in I}\Gamma_v^i \vdash^{(+_{i\in I}b_v^i,+_{i\in I}m_v^i,+_{i\in I}d_v^i)} v : \mathcal{M}$. And we can conclude with $\Gamma_{t\{x\backslash v\}} = +_{i\in I}\Gamma_i = +_{i\in I}(\Gamma_t^i + \Gamma_v^i) = +_{i\in I}\Gamma_t^i +_{i\in I} \Gamma_v^i = \Gamma_t + \Gamma_v$, $b = +_{i\in I}b_i = +_{i\in I}(b_t^i + b_v^i) = +_{i\in I}b_t^i +_{i\in I} b_v^i = b_t + b_v$, $m = +_{i\in I}m_i = +_{i\in I}(m_t^i + m_v^i) = +_{i\in I}m_t^i +_{i\in I} m_v^i = m_t + m_v$, and $d = +_{i\in I}d_i = +_{i\in I}(d_t^i + d_v^i) = +_{i\in I}d_t^i +_{i\in I} d_v^i = d_t + d_v$.

– Let $t = wu$. Then $t\{x\backslash v\} = (wu)\{x\backslash v\} = (w\{x\backslash v\})(u\{x\backslash v\})$, and we have to consider three cases:

- Case $\Phi_{t\{x\backslash v\}}$ ends with $(@)$. Assume $\Phi_{w\{x\backslash v\}} \triangleright \Gamma_{w\{x\backslash v\}} \vdash^{(b',m',d')} w\{x\backslash v\} : \mathcal{M}' \Rightarrow (\mathcal{S}' \gg \kappa)$ and $\Phi_{u\{x\backslash v\}} \triangleright \Gamma_{u\{x\backslash v\}} \vdash^{(b'',m'',d'')} u\{x\backslash v\} : \mathcal{S} \gg (\mathcal{M}' \times \mathcal{S}')$. $\Phi_{t\{x\backslash v\}}$ must be of the following form:

$$\frac{\Phi_{w\{x\backslash v\}} \qquad \Phi_{u\{x\backslash v\}}}{\Gamma_{w\{x\backslash v\}} + \Gamma_{u\{x\backslash v\}} \vdash^{(1+b'+b'',m'+m'',d'+d'')} (w\{x\backslash v\})(u\{x\backslash v\}) : \mathcal{S} \gg \kappa}\ (@)$$

where $\mathcal{T} = \mathcal{S} \gg \kappa$, $\Gamma_{t\{x\backslash v\}} = \Gamma_{w\{x\backslash v\}} + \Gamma_{u\{x\backslash v\}}$, $b = 1 + b' + b''$, $m = m' + m''$, and $d = d' + d''$. By the *i.h.* over $\Phi_{w\{x\backslash v\}}$, we have $\Phi_w \triangleright \Gamma_w; x :$

$\mathcal{M}_1 \vdash^{(b_w,m_w,d_w)} w : \mathcal{M}' \Rightarrow (\mathcal{S}' \gg \kappa)$ and $\Phi^1_v \triangleright \Gamma^1_v \vdash^{(b^1_v,m^1_v,d^1_v)} v : \mathcal{M}_1$, such that $\Gamma_{w\{x\backslash v\}} = \Gamma_w + \Gamma^1_v$, $b' = b_w + b^1_v$, $m' = m_w + m^1_v$, and $d' = d_w + d^1_v$. And by the *i.h.* over $\Phi_{u\{x\backslash v\}}$, we have $\Phi_u \triangleright \Gamma_u; x : \mathcal{M}_2 \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg (\mathcal{M}' \times \mathcal{S}')$ and $\Phi^2_v \triangleright \Gamma^2_v \vdash^{(b^2_v,m^2_v,d^2_v)} v : \mathcal{M}_2$, such that $\Gamma_{u\{x\backslash v\}} = \Gamma_u + \Gamma^2_v$, $b'' = b_u + b^2_v$, $m'' = m_u + m^2_v$, and $d'' = d_u + d^2_v$. By Lemma 24, we can take $\Phi_v \triangleright \Gamma^1_v + \Gamma^2_v \vdash^{(b^1_v+b^2_v,m^1_v+m^2_v,d^1_v+d^2_v)} v : \mathcal{M}_1 \sqcup \mathcal{M}_2$, such that $\Gamma_v = \Gamma^1_v + \Gamma^2_v$, $b_v = b^1_v + b^2_v$, $m_v = m^1_v + m^2_v$, and $d_v = d^1_v + d^2_v$. And we can build Φ_{wu} as follows:

$$\frac{\Phi_w \qquad \Phi_u}{(\Gamma_w + \Gamma_u); x : \mathcal{M}_1 \sqcup \mathcal{M}_2 \vdash^{(1+b_w+b_u,m_w+m_u,d_w+d_u)} wu : \mathcal{S} \gg \kappa}\ (\mathtt{@})$$

such that $\Gamma_t = \Gamma_w + \Gamma_u$, $b_t = 1 + b_w + b_u$, $m_t = b_w + b_u$, and $d_t = d_w + d_u$. So we can pick $\Phi_t = \Phi_{wu}$, and conclude with $\Gamma_{t\{x\backslash v\}} = \Gamma_{w\{x\backslash v\}} + \Gamma_{u\{x\backslash v\}} = (\Gamma_w + \Gamma^1_v) + (\Gamma_u + \Gamma^2_v) = (\Gamma_w + \Gamma_u) + (\Gamma^1_v + \Gamma^2_v) = \Gamma_t + \Gamma_v$, $b = 1+b'+b'' = 1+b_w+b^1_v+b_u+b^2_v = (1+b_w+b_u)+(b^1_v+b^2_v) = b_t+b_v$, $m = m' + m'' = m_w + m^1_v + m_u + m^2_v = (m_w + m_u) + (m^1_v + m^2_v) = m_t + m_v$, and $d = d' + d'' = d_w + d^1_v + d_u + d^2_v = (d_w + d_u) + (d^1_v + d^2_v) = d_t + d_v$.

- Case $\Phi_{t\{x\backslash v\}}$ ends with $(\mathtt{@_{p1}})$ or $(\mathtt{@_{p2}})$. These cases are very similar to the case where $\Phi_{t\{x\backslash v\}}$ ends with $(\mathtt{@})$.

– Let $t = \mathtt{get}_l(\lambda y.u)$. Then $t\{x\backslash v\} = \mathtt{get}_l(\lambda y.u\{x\backslash v\})$ and $\Phi_{t\{x\backslash v\}}$ must be of the following form:

$$\frac{\Phi_{u\{x\backslash v\}} \triangleright \Gamma_{u\{x\backslash v\}}; y : \mathcal{M}' \vdash^{(b,m',d)} u\{x\backslash v\} : \mathcal{S} \gg \kappa}{\Gamma_{u\{x\backslash v\}} \vdash^{(b,1+m',d)} \mathtt{get}_l(\lambda y.u\{x\backslash v\}) : \{(l : \mathcal{M}')\} \uplus \mathcal{S} \gg \kappa}\ (\mathtt{get})$$

where $\Gamma_{t\{x\backslash v\}} = \Gamma_{u\{x\backslash v\}}$ and $m = 1 + m'$. By the *i.h.*, we have $\Phi_u \triangleright \Gamma_u; y : \mathcal{M}'; x : \mathcal{M} \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg \kappa$ and $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M}$, such that $\Gamma_{u\{x\backslash v\}} = \Gamma_u + \Gamma_v$, $b = b_u + b_v$, $m' = m_u + m_v$, and $d = d_u + d_v$. So we can build $\Phi_{\mathtt{get}_l(\lambda y.u)}$ as follows:

$$\frac{\Phi_u \triangleright \Gamma_u; y : \mathcal{M}'; x : \mathcal{M} \vdash^{(b_u,m_u,d_u)} u : \mathcal{S} \gg \kappa}{\Gamma_u; x : \mathcal{M} \vdash^{(b_u,1+m_u,d_u)} \mathtt{get}_l(\lambda y.u) : \{(l : \mathcal{M}')\} \uplus \mathcal{S} \gg \kappa}\ (\mathtt{get})$$

And we can pick $\Phi_t = \Phi_{\mathtt{get}_l(\lambda y.u)}$, and conclude with $\Gamma_{t\{x\backslash v\}} = \Gamma_{u\{x\backslash v\}} = \Gamma_u + \Gamma_v$, $b = b_u + b_v$, $m = 1 + m' = 1 + m_u + m_v = (1 + m_u) + m_v$, and $d = d_u + d_v$.

– Let $t = \mathtt{set}_l(w, u)$. Then $t\{x\backslash v\} = (\mathtt{set}_l(w, u))\{x\backslash v\} = \mathtt{set}_l(w\{x\backslash v\}, u\{x\backslash v\})$. Assume $\Phi_{w\{x\backslash v\}} \triangleright \Gamma_{w\{x\backslash v\}} \vdash^{(b',m',d')} w\{x\backslash v\} : \mathcal{M}$ and $\Phi_{u\{x\backslash v\}} \triangleright \Gamma_{u\{x\backslash v\}} \vdash^{(b'',m'',d'')} u\{x\backslash v\} : \{(l : \mathcal{M})\}; \mathcal{S} \gg \kappa$. $\Phi_{t\{x\backslash v\}}$ must be of the following form:

$$\frac{\Phi_{w\{x\backslash v\}} \qquad \Phi_{t\{x\backslash v\}}}{\Gamma_{w\{x\backslash v\}} + \Gamma_{u\{x\backslash v\}} \vdash^{(b'+b'',1+m'+m'',d'+d'')} \mathtt{set}_l(w\{x\backslash v\}, u\{x\backslash v\}) : \mathcal{S} \gg \kappa}\ (\mathtt{set})$$

where $\Gamma_{t\{x\backslash v\}} = \Gamma_{w\{x\backslash v\}} + \Gamma_{u\{x\backslash v\}}$, $b = b' + b''$, $m = 1 + m' + m''$, and $d = d' + d''$. By the *i.h.* over $\Phi_{w\{x\backslash v\}}$, we have $\Phi_w \triangleright \Gamma_w; x : \mathcal{M}_1 \vdash^{(b_w, m_w, d_w)} w : \mathcal{M}$ and $\Phi_v^1 \triangleright \Gamma_v^1 \vdash^{(b_v^1, m_v^1, d_v^1)} v : \mathcal{M}_1$, such that $\Gamma_{w\{x\backslash v\}} = \Gamma_w + \Gamma_v^1$, $b' = b_w + b_v^1$, $m' = m_w + m_v^1$, and $d' = d_w + d_v^1$. And by the *i.h.* over $\Phi_{u\{x\backslash v\}}$, we have $\Phi_u \triangleright \Gamma_u; x : \mathcal{M}_2 \vdash^{(b_u, m_u, d_u)} u : \{(l : \mathcal{M})\}; \mathcal{S} \gg \kappa$ and $\Phi_v^2 \triangleright \Gamma_v^2 \vdash^{(b_v^2, m_v^2, d_v^2)} v : \mathcal{M}_2$, such that $\Gamma_{u\{x\backslash v\}} = \Gamma_u + \Gamma_v^2$, $b'' = b_u + b_v^2$, $m'' = m_u + m_v^2$, and $d'' = d_u + d_v^2$. By Lemma 24, we can take $\Phi_v \triangleright \Gamma_v^1 + \Gamma_v^2 \vdash^{(b_v^1 + b_v^2, m_v^1 + m_v^2, d_v^1 + d_v^2)} v : \mathcal{M}_1 \sqcup \mathcal{M}_2$, such that $\Gamma_v = \Gamma_v^1 + \Gamma_v^2$, $b_v = b_v^1 + b_v^2$, $m_v = m_v^1 + m_v^2$, and $d_v = d_v^1 + d_v^2$. And we can build $\Phi_{\mathtt{set}_l(w,u)}$ as follows:

$$\dfrac{\Phi_w \triangleright \Gamma_w; x : \mathcal{M}_1 \vdash^{(b_w, m_w, d_w)} w : \mathcal{M} \qquad \Phi_u \triangleright \Gamma_u; x : \mathcal{M}_2 \vdash^{(b_u, m_u, d_u)} u : \{(l : \mathcal{M})\}; \mathcal{S} \gg \kappa}{(\Gamma_w + \Gamma_u); x : \mathcal{M}_1 \sqcup \mathcal{M}_2 \vdash^{(b_w + b_u, 1 + m_w + m_u, d_w + d_u)} \mathtt{set}_l(w, u) : \mathcal{S} \gg \kappa}\ (\mathtt{set})$$

such that $\Gamma_t = \Gamma_w + \Gamma_u$, $b_t = b_w + b_u$, $m_t = 1 + m_w + m_u$, and $d_t = d_w + d_u$. So we can pick $\Phi_t = \Gamma_{\mathtt{set}_l(w,u)}$, and conclude with $\Gamma_{t\{x\backslash v\}} = \Gamma_{w\{x\backslash v\}} + \Gamma_{u\{x\backslash v\}} = (\Gamma_w + \Gamma_u^1) + (\Gamma_u + \Gamma_v^2) = (\Gamma_w + \Gamma_u) + (\Gamma_v^1 + \Gamma_v^2) = \Gamma_t + \Gamma_v$, $b = b' + b'' = (b_w + b_v^1) + (b_u + b_v^2) = (b_w + b_u) + (b_v^1 + b_v^2) = b_t + b_v$, $m = 1 + m' + m'' = 1 + (m_w + m_v^1) + (m_u + m_v^2) = (1 + m_w + m_u) + (m_v^1 + m_v^2) = m_t + m_v$, and $d = d' + d'' = (d_w + d_v^1) + (d_u + d_v^2) = (d_w + d_u) + (d_v^1 + d_v^2) = d_t + d_v$.

Lemma 6 (Split Exact Subject Reduction and Expansion).

1. **(Subject Reduction)** *Let* $(t, s) \to_{\mathtt{r}} (u, q)$. *If* $\Phi \triangleright \Gamma \vdash^{(b,m,d)} (t, s) : \kappa$ *is tight, then* $\Phi' \triangleright \Gamma \vdash^{(b',m',d)} (u, q) : \kappa$, *where* $\mathtt{r} = \beta$ *implies* $b' = b - 1$ *and* $m' = m$, *while* $\mathtt{r} \in \{\mathtt{g}, \mathtt{s}\}$ *implies* $b' = b$ *and* $m' = m - 1$.
2. **(Subject Expansion)** *Let* $(t, s) \to_{\mathtt{r}} (u, q)$. *If* $\Phi' \triangleright \Gamma \vdash^{(b',m',d)} (u, q) : \kappa$ *is tight, then* $\Phi \triangleright \Gamma \vdash^{(b,m,d)} (t, s) : \kappa$, *where* $\mathtt{r} = \beta$ *implies* $b' = b - 1$ *and* $m' = m$, *while* $\mathtt{r} \in \{\mathtt{g}, \mathtt{s}\}$ *implies* $b' = b$ *and* $m' = m - 1$.

Proof.

1. We show a stronger statement of the form:
   Let $(t, s) \to_{\mathtt{r}} (u, q)$. If $\Phi \triangleright \Gamma \vdash^{(b,m,d)} (t, s) : \kappa$, Γ is tight, and (κ is tight or $\neg\mathtt{val}(t)$), then $\Phi' \triangleright \Gamma \vdash^{(b',m',d)} (u, q) : \kappa$, where $\mathtt{r} = \beta$ implies $b' = b - 1$ and $m' = m$, while $\mathtt{r} \in \{\mathtt{g}, \mathtt{s}\}$ implies $b' = b$ and $m' = m - 1$.
   We proceed by induction on $(t, s) \to (u, q)$:
   - Case $(t, s) = ((\lambda x.p)v, s) \to_\beta (p\{x \backslash v\}, s) = (u, q)$. Let $\Phi_{(\lambda x.p)v}$ be the sub-derivation for $(\lambda x.p)v$ in Φ. Assume that $\Phi_{(\lambda x.p)v}$ ends with rule $(@_{\mathtt{p2}})$. Then v must be assigned type $\mathcal{S} \gg \mathtt{n} \times \mathcal{S}$, which is not possible by Lemma 22. Let Φ_0 be the following derivation:

$$\dfrac{\dfrac{\Phi_p \triangleright \Gamma_{\lambda x.p}; x : \mathcal{M} \vdash^{(b_p, m_p, d_p)} p : \mathcal{S} \gg \kappa}{\Gamma_{\lambda x.p} \vdash^{(b_p, m_p, d_p)} \lambda x.p : \mathcal{M} \Rightarrow (\mathcal{S} \gg \kappa)}\ (\lambda) \qquad \dfrac{\Phi_v \triangleright \Gamma_v \vdash^{(b_v, m_v, d_v)} v : \mathcal{M}}{\Gamma_v \vdash^{(b_v, m_v, d_v)} v : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S})}\ (\uparrow)}{\Gamma_{\lambda x.p} + \Gamma_v \vdash^{(1 + b_v + b_p, m_v + m_p, d_v + d_p)} (\lambda x.p)v : \mathcal{S} \gg \kappa}\ (@)$$

$\Phi_{(\lambda x.p)v}$ must end with rule (@) and Φ must be of the following form:

$$\frac{\Phi_0 \qquad \Phi_s \triangleright \Delta \vdash^{(b_s,m_s,d_s)} s : \mathcal{S}}{\Gamma_{\lambda x.p} + \Gamma_v + \Delta \vdash^{(1+b_v+b_p+b_s,m_v+m_p+m_s,d_v+d_p+d_s)} ((\lambda x.p)v, s) : \kappa} \ (\texttt{conf})$$

where $\Gamma = \Gamma_{\lambda x.p} + \Gamma_v + \Delta$, $b = 1 + b_v + b_p + b_s$, $m = m_v + m_p + m_s$, and $d = d_v + d_p + d_s$. By Lemma 5.1, there exists $\Phi_{p\{x\backslash v\}} \triangleright \Gamma_{\lambda x.p} + \Gamma_v \vdash^{(b_v+b_p,m_v+m_p,d_v+d_p)} p\{x \setminus v\} : \mathcal{S} \gg \kappa$, therefore we can build $\Phi_{(p\{x\backslash v\},s)}$ as follows:

$$\frac{\Phi_{p\{x\backslash v\}} \triangleright \Gamma_{\lambda x.p} + \Gamma_v \vdash^{(b_v+b_p,m_v+m_p,d_v+d_p)} p\{x \setminus v\} : \mathcal{S} \gg \kappa \qquad \Phi_s \triangleright \Delta \vdash^{(b_s,m_s,d_s)} s : \mathcal{S}}{\Gamma_{\lambda x.p} + \Gamma_v + \Delta \vdash^{(b_v+b_p+b_s,m_v+m_p+m_s,d_v+d_p+d_s)} (p\{x \setminus v\}, s) : \kappa} \ (\texttt{conf})$$

We can finally conclude since the first counter is equal to $b - 1$, while the second and third remain the same.

– Case $(t, s) = (vp, s) \to (vp', q) = (u, q)$, such that $(p, s) \to (p', q)$. Then we have three cases for the type derivation Φ_p of p inside Φ:
  - Case Φ_p ends with (@). Let Φ_0 be the following derivation:

$$\frac{\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa) \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,m_p,d_p)} p : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}')}{\Gamma_v + \Gamma_p \vdash^{(1+b_v+b_p,m_v+m_p,d_v+d_p)} vp : \mathcal{S} \gg \kappa} \ (@)$$

    Φ must be of the following form:

$$\frac{\Phi_0 \qquad \Phi_s \triangleright \Delta \vdash^{(b_s,m_s,d_s)} s : \mathcal{S}}{\Gamma_v + \Gamma_p + \Delta \vdash^{(1+b_v+b_p+b_s,m_v+m_p+m_s,d_v+d_p+d_s)} (vp, s) : \kappa} \ (\texttt{conf})$$

    where $\Gamma = \Gamma_v + \Gamma_p + \Delta$ is tight, $b = 1 + b_v + b_p + b_s$, $m = m_v + m_p + m_s$, and $s = d_v + d_p + d_s$. Therefore, we can build the following derivation for (p, s):

$$\frac{\Phi_p \triangleright \Gamma_p \vdash^{(b_p,m_p,d_p)} p : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}') \qquad \Phi_s \triangleright \Delta \vdash^{(b_s,m_s,d_s)} s : \mathcal{S}}{\Gamma_p + \Delta \vdash^{(b_p+b_s,m_p+m_s,d_p+d_s)} (p, s) : \mathcal{M} \times \mathcal{S}'} \ (\texttt{conf})$$

    Since Γ is tight, then $\Gamma_p + \Delta$ is tight. Moreover, $(p, s) \to (p', q)$ implies that $\neg\texttt{val}(p)$. Then we can apply the *i.h.*, and thus there exists a derivation for (p', q) that must be of the following form:

$$\frac{\Phi_{p'} \triangleright \Gamma_{p'} \vdash^{(b_{p'},m_{p'},d_{p'})} p' : \mathcal{S}'' \gg (\mathcal{M} \times \mathcal{S}') \qquad \Phi_q \triangleright \Delta_q \vdash^{(b_q,m_q,d_q)} q : \mathcal{S}''}{\Gamma_{p'} + \Delta_q \vdash^{(b_{p'}+b_q,m_{p'}+m_q,d_{p'}+d_q)} (p', q) : \mathcal{M} \times \mathcal{S}'} \ (\texttt{conf})$$

    where $\Gamma_{p'} + \Delta_q = \Gamma_p + \Delta$ is tight, and the counters are related properly. Let Φ_0 be the following derivation:

$$\frac{\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M} \Rightarrow \mathcal{S}' \gg \kappa' \qquad \Phi_{p'} \triangleright \Gamma_{p'} \vdash^{(b_{p'},m_{p'},d_{p'})} p' : \mathcal{S}'' \gg (\mathcal{M} \times \mathcal{S}')}{\Gamma_v + \Gamma_{p'} \vdash^{(1+b_v+b_{p'},m_v+m_{p'},d_v+d_{p'})} vp' : \mathcal{S}'' \gg \kappa'} \ (@)$$

We can build $\Phi_{(u,q)}$ as follows:

$$\frac{\Phi_0 \qquad \Phi_q \triangleright \Delta_q \vdash^{(b_q,m_q,d_q)} q : \mathcal{S}''}{\Gamma_v + \Gamma_{p'} + \Delta_q \vdash^{(1+b_v+b_{p'}+b_q, m_v+m_{p'}+m_q, d_v+d_{p'}+d_q)} (vp', q) : \kappa'} \; (\texttt{conf})$$

where $\Gamma_{p'} + \Gamma_v + \Delta_q = \Gamma_v + \Gamma_p + \Delta = \Gamma$, $b' = 1 + b_v + b_{p'} + b_q$, $m' = m_v + m_{p'} + m_q$, and $d' = d_v + d_{p'} + d_q$. We can conclude since the counters are related properly according to the *i.h.*.

- Case Φ_p ends with ($\texttt{@}_{\texttt{p1}}$) or ($\texttt{@}_{\texttt{p2}}$). These two cases are very similar to the previous case.

– Case $(t, s) = (\texttt{get}_l(\lambda x.p), s) \to (p\{x\backslash v\}, s) = (u, q)$, where $s \equiv \texttt{upd}_l(v, s')$. Let Φ_0 be the following derivation:

$$\frac{\Phi_p \triangleright \Gamma_p \vdash^{(b_p,m_p,d_p)} p : \mathcal{S} \gg \kappa}{\Gamma_p \setminus\!\!\setminus x \vdash^{(b_p,1+m_p,d_p)} \texttt{get}_l(\lambda x.p) : \{(l : \Gamma_p(x))\} \Cup \mathcal{S} \gg \kappa} \; (\texttt{get})$$

Φ must be of the following form:

$$\frac{\Phi_0 \qquad \Phi_s \triangleright \Delta \vdash^{(b_s,m_s,d_s)} s : \{(l : \Gamma_p(x))\} \Cup \mathcal{S}}{(\Gamma_p \setminus\!\!\setminus x) + \Delta \vdash^{(b_p+b_s, 1+m_p+m_s, d_p+d_s)} (\texttt{get}_l(\lambda x.p), s) : \kappa} \; (\texttt{conf})$$

where $\Gamma = (\Gamma_p \setminus\!\!\setminus x) + \Delta$ is tight, $b = b_p + b_s$, $m = 1 + m_p + m_s$, and $d = d_p + d_s$. Since $\Phi_s \triangleright \Delta \vdash^{(b_s,m_s,d_s)} s : \{(l : \Gamma_p(x))\} \Cup \mathcal{S}$, then Lemma 21.2 gives $s \equiv \texttt{upd}_l(v_0, s'_0)$, but we necessarily have $v_0 = v$ and $s'_0 = s'$. Moreover, the lemma also gives $\Phi_v \triangleright \Delta_v \vdash^{(b_v,m_v,d_v)} v : \Gamma_p(x) \sqcup \mathcal{S}(l)$ and $\Phi_{s'} \triangleright \Delta_{s'} \vdash^{(b_{s'},m_{s'},d_{s'})} s' : \mathcal{S}'$, where $\{(l : \Gamma_p(x))\} \Cup \mathcal{S} = \{(l : \Gamma_p(x) \sqcup \mathcal{S}(l))\}; \mathcal{S}'$, $\Delta = \Delta_v + \Delta_{s'}$, $b_s = b_v + b_{s'}$, $m_s = m_v + m_{s'}$, and $d_s = d_v + d_{s'}$. Thus, by Lemma 21.1 there exist $\Phi^1_v \triangleright \Delta^1_v \vdash^{(b^1_v,m^1_v,d^1_v)} v : \Gamma_p(x)$ and $\Phi^2_v \triangleright \Delta^2_v \vdash^{(b^2_v,m^2_v,d^2_v)} v : \mathcal{S}(l)$, such that $\Delta_v = \Delta^1_v + \Delta^2_v$, $b_v = b^1_v + b^2_v$, $m_v = m^1_v + m^2_v$, and $d_v = d^1_v + d^2_v$. From $\Phi_p \triangleright \Gamma_p \vdash^{(b_p,m_p,d_p)} p : \mathcal{S} \gg \kappa$ and $\Phi^1_v \triangleright \Delta^1_v \vdash^{(b^1_v,m^1_v,d^1_v)} v : \Gamma_p(x)$, we obtain $\Phi_{p\{x\backslash v\}} \triangleright (\Gamma_p \setminus\!\!\setminus x) + \Delta^1_v \vdash^{(b_p+b^1_v, m_p+m^1_v, d_p+d^1_v)} p\{x \setminus v\} : \mathcal{S} \gg \kappa$, by Lemma 5.1. We now construct an alternative type derivation for s of the form:

$$\frac{\Phi^2_v \triangleright \Delta^2_v \vdash^{(b^2_v,m^2_v,d^2_v)} v : \mathcal{S}(l) \qquad \Phi_{s'} \triangleright \Delta_{s'} \vdash^{(b_{s'},m_{s'},d_{s'})} s' : \mathcal{S}'}{\Delta^2_v + \Delta_{s'} \vdash^{(b^2_v+b_{s'}, m^2_v+m_{s'}, d^2_v+d_{s'})} \texttt{upd}_l(v, s') : \{(l : \mathcal{S}(l))\}; \mathcal{S}'} \; (\texttt{upd})$$

Let $q = s = \texttt{upd}_l(v, s')$ and let Φ_q be this new derivation above. Notice also that $\mathcal{S} = \{(l : \mathcal{S}(l))\}; \mathcal{S}'$. Then we can construct Φ' as follows:

$$\frac{\Phi_{p\{x\backslash v\}} \qquad \Phi_q}{(\Gamma_p \setminus\!\!\setminus x) + \Delta^1_v + \Delta^2_v + \Delta_{s'} \vdash^{(b,m,d)} (p\{x \setminus v\}, s) : \kappa} \; (\texttt{conf})$$

Notice that the type environment of the conclusion is $(\Gamma_p \setminus\!\!\setminus x) + \Delta^1_v + \Delta^2_v + \Delta_{s'} = (\Gamma_p \setminus\!\!\setminus x) + \Delta_v + \Delta_{s'} = (\Gamma_p \setminus\!\!\setminus x) + \Delta = \Gamma$, and the counters are as expected.

– Case $(t,s) = (\mathtt{set}_l(v,p),s) \to (p, \mathtt{upd}_l(v,s)) = (u,q)$. Let Φ_0 be the following derivation:

$$\dfrac{\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M} \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,m_p,d_p)} p : \{(l:\mathcal{M})\};\mathcal{S} \gg \kappa}{\Gamma_v + \Gamma_p \vdash^{(b_v+b_p,1+m_v+m_p,s_v+s_p)} \mathtt{set}_l(v,p) : \mathcal{S} \gg \kappa}\ (\mathtt{set})$$

Φ must be of the following form:

$$\dfrac{\Phi_0 \qquad \Phi_s \triangleright \Gamma_s \vdash^{(b_s,m_s,d_s)} s : \mathcal{S}}{(\Gamma_v + \Gamma_p) + \Gamma_s \vdash^{(b_v+b_p+b_s,1+m_v+m_p+m_s,d_v+d_p+d_s)} (\mathtt{set}_l(v,p),s) : \kappa}\ (\mathtt{conf})$$

where $\Gamma = (\Gamma_v+\Gamma_p)+\Gamma_s$ is tight, $b = b_v+b_p+b_s$, $m = 1+m_v+m_p+m_s$ and $d = d_v + d_p + d_s$. Therefore, we can build $\Phi_{\mathtt{upd}_l(v,s)}$ as follows:

$$\dfrac{\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M} \qquad \Phi_s \triangleright \Gamma_s \vdash^{(b_s,m_s,d_s)} s : \mathcal{S}}{\Gamma_v + \Gamma_s \vdash^{(b_v+b_s,m_v+m_s,d_v+d_s)} \mathtt{upd}_l(v,s) : \{(l:\mathcal{M})\};\mathcal{S}}\ (\mathtt{upd})$$

Assume And we can build Φ' as follows:

$$\dfrac{\Phi_p \triangleright \Gamma_p \vdash^{(b_p,m_p,d_p)} p : \{(l:\mathcal{M})\};\mathcal{S} \gg \kappa \qquad \Phi_{\mathtt{upd}_l(v,s)}}{\Gamma_p + (\Gamma_v + \Gamma_s) \vdash^{(b_v+b_v+b_s,m_v+m_v+m_s,d_v+d_v+d_s)} (p, \mathtt{upd}_l(v,s)) : \kappa}\ (\mathtt{conf})$$

Notice that the type environment of the conclusion is $\Gamma_p + (\Gamma_v + \Gamma_s) = \Gamma$, and the counters are as expected.

2. We show a stronger statement of the form:
Let $(t,s) \to_{\mathtt{r}} (u,q)$. If $\Phi' \triangleright \Gamma \vdash^{(b',m',d)} (u,q) : \kappa$, Γ is tight, and (κ is tight or $\neg\mathtt{val}(t)$), then $\Phi \triangleright \Gamma \vdash^{(b,m,d)} (t,s) : \kappa$, where $\mathtt{r} = \beta$ implies $b' = b-1$ and $m' = m$, while $\mathtt{r} \in \{\mathtt{g},\mathtt{s}\}$ implies $b' = b$ and $m' = m-1$.
We proceed by induction on $(t,s) \to (u,q)$:
– Case $(t,s) = ((\lambda x.p)v,s) \to_\beta (p\{x\backslash v\},s) = (u,q)$. Then Φ' must be of the following form:

$$\dfrac{\Phi_{p\{x\backslash v\}} \triangleright \Gamma_{p\{x\backslash v\}} \vdash^{(b'',m'',d'')} p\{x\backslash v\} : \mathcal{S} \gg \kappa \qquad \Phi_s \triangleright \Gamma_s \vdash^{(b_s,m_s,d_s)} s : \mathcal{S}}{\Gamma_{p\{x\backslash v\}} + \Gamma_s \vdash^{(b''+b_s,m''+m_s,d''+d_s)} (p\{x\backslash v\},s) : \kappa}\ (\mathtt{conf})$$

such that $\Gamma = \Gamma_{p\{x\backslash v\}} + \Gamma_s$, $b' = b''+b_s$, $m' = m''+m_s$, and $d' = d''+d_s$. By Lemma 5.2, there exist $\Phi_p \triangleright \Gamma_p; x : \mathcal{M} \vdash^{(b_p,m_p,d_p)} p : \mathcal{S} \gg \kappa$ and $\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M}$, such that $\Gamma_{p\{x\backslash v\}} = \Gamma_p + \Gamma_v$, $b'' = b_p + b_v$, $m'' = m_p + m_v$, and $d'' = d_p + d_v$. We can build Φ as follows:

$$\dfrac{\dfrac{\dfrac{\Phi_p \triangleright \Gamma_p; x:\mathcal{M} \vdash^{(b_p,m_p,d_p)} p : \mathcal{S} \gg \kappa}{\Gamma_p \vdash^{(b_p,m_p,d_p)} \lambda x.p : \mathcal{M} \Rightarrow (\mathcal{S} \gg \kappa)}\ (\lambda) \qquad \dfrac{\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{M}}{\Gamma_v \vdash^{(b_v,m_v,d_v)} v : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S})}\ (\uparrow)}{\Gamma_p + \Gamma_v \vdash^{(1+b_p+b_v,m_p+m_v,d_p+d_v)} (\lambda x.p)v : \mathcal{S} \gg \kappa}\ (\mathtt{@}) \qquad \Phi_s}{(\Gamma_p + \Gamma_v) + \Gamma_s \vdash^{(1+b_p+b_v+b_s,m_p+m_v+m_s,d_p+d_v+d_s)} ((\lambda x.t')v,s) : \kappa}\ (\mathtt{conf})$$

such that $b = 1 + b_p + b_v + b_s$, $m = m_p + m_v + m_s$, and $d = d_p + d_v + d_s$. And we can conclude with $\Gamma = \Gamma_{p\{x\backslash v\}} + \Gamma_s = (\Gamma_p + \Gamma_v) + \Gamma_s$, $b' = b'' + b_s = b_p + b_v + b_s = (1 + b_p + b_v + b_s) - 1 = b - 1$, $m' = m'' + m_s = (m_p + m_v) + m_s = m$, and $d' = d'' + d_s = (d_p + d_v) + d_s = d$.

– Case $(t, s) = (vp, s) \to (vp', q) = (u, q)$, such that $(p, s) \to (p', q)$. Then we have three cases for the type derivation $\Phi_{p'}$ of p' inside Φ':
  - Case $\Phi_{vp'}$ ends with $(\texttt{@})$. Let Φ_0 be the following derivation:

$$\dfrac{\Phi_v \rhd \Gamma_v \vdash^{(b_v, m_v, d_v)} v : \mathcal{M} \Rightarrow \mathcal{S}' \gg \kappa \qquad \Phi_{p'} \rhd \Gamma_{p'} \vdash^{(b'', m'', d'')} p' : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}')}{\Gamma_v + \Gamma_{p'} \vdash^{(1 + b_v + b'', m_v + m'', d_v + d'')} vp' : \mathcal{S} \gg \kappa} \ (\texttt{@})$$

Φ' must be of the following form:

$$\dfrac{\Phi_0 \qquad \Phi_q \rhd \Gamma_q \vdash^{(b_q, m_q, d_q)} q : \mathcal{S}}{(\Gamma_v + \Gamma_{p'}) + \Gamma_q \vdash^{(1 + b_v + b'' + b_q, m_v + m'' + m_q, d_v + d'' + d_q)} (vp', q) : \kappa} \ (\texttt{conf})$$

such that $\Gamma = (\Gamma_v + \Gamma_{p'}) + \Gamma_q$ tight, $b' = 1 + b_v + b'' + b_q$, $m' = m_v + m'' + m_q$, and $d' = d_v + d'' + d_q$. So we can build $\Phi_{(p',q)}$ as follows:

$$\dfrac{\Phi_{p'} \rhd \Gamma_{p'} \vdash^{(b'', m'', d'')} p' : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S}') \qquad \Phi_q \rhd \Gamma_q \vdash^{(b_q, m_q, d_q)} q : \mathcal{S}}{\Gamma_{p'} + \Gamma_q \vdash^{(b'' + b_q, m'' + m_q, d'' + d_q)} (p', q) : \mathcal{M} \times \mathcal{S}'} \ (\texttt{conf})$$

Since Γ is tight, then $\Gamma_{p'} + \Gamma_q$ is tight. Moreover, $(p, s) \to (p', q)$ implies $\neg\texttt{val}(p)$. Then we can apply the *i.h.*, and thus there exists a derivation for (p, s) that must be of the following form:

$$\dfrac{\Phi_p \rhd \Gamma_p \vdash^{(b_p, m_p, d_p)} p : \mathcal{S}'' \gg (\mathcal{M} \times \mathcal{S}') \qquad \Phi_s \rhd \Gamma_s \vdash^{(b_s, m_s, d_s)} s : \mathcal{S}''}{\Gamma_p + \Gamma_s \vdash^{(b_p + b_s, m_p + m_s, d_p + d_s)} (p, s) : \mathcal{M} \times \mathcal{S}'} \ (\texttt{conf})$$

where $\Gamma_p + \Gamma_s = \Gamma_{p'} + \Gamma_q$ is tight, and either (1) $b'' + b_q = b_p + b_s - 1$, $m'' + m_q = m_p + m_s$, and $d'' + d_q = d_p + d_s$, or (2) $b'' + b_q = b_p + b_s$, $m'' + m_q = m_p + m_s - 1$, and $d'' + d_q = d_p + d_s$. So, we can build Φ as follows:

$$\dfrac{\dfrac{\Phi_v \rhd \Gamma_v \vdash^{(b_v, m_v, d_v)} v : \mathcal{M} \Rightarrow (\mathcal{S}' \gg \kappa) \qquad \Phi_p \rhd \Gamma_p \vdash^{(b_p, m_p, d_p)} p : \mathcal{S}'' \gg (\mathcal{M} \times \mathcal{S}')}{\Gamma_v + \Gamma_p \vdash^{(1 + b_v + b_p, m_v + m_p, d_v + d_p)} vp : \mathcal{S}'' \gg \kappa} \ (\texttt{@}) \qquad \Phi_s}{(\Gamma_v + \Gamma_p) + \Gamma_s \vdash^{(1 + b_v + b_p + b_s, m_v + m_p + m_s, d_v + d_p + d_s)} (vp, s) : \kappa} \ (\texttt{conf})$$

where $\Gamma_v + \Gamma_p + \Gamma_s = \Gamma_v + \Gamma_{p'} + \Gamma_q = \Gamma$, $b = 1 + b_v + b_p + b_s$, $m = m_v + m_p + m_s$, and $d = d_v + d_p + d_s$. We can conclude since:
    - Case (1): $b' = 1 + b_v + b'' + b_q = 1 + b_v + b_p + b_s - 1 = b - 1$, and the other counters are easy to check;
    - Case (2): $m' = m_v + m'' + m_q = m_v + m_p + m_s - 1 = m - 1$, and the other counters are easy to check.

- Case $\Phi_{vp'}$ ends with ($\texttt{@}_{\texttt{p1}}$) or ($\texttt{@}_{\texttt{p2}}$). These two cases are very similar to the previous case.

– Case $(t,s) = (\texttt{get}_l(\lambda x.p), s) \to (p\{x \backslash v\}, s) = (u,q)$, such that $s \equiv \texttt{upd}_l(v,s')$. Let Φ_0 be the following derivation:

$$\dfrac{\Phi_v^2 \rhd \Gamma_v^2 \vdash^{(b_v^2, m_v^2, d_v^2)} v : \mathcal{M}_2 \qquad \Phi_{s'} \rhd \Gamma_{s'} \vdash^{(b_{s'}, m_{s'}, d_{s'})} s' : \mathcal{S}}{\Gamma_v^2 + \Gamma_{s'} \vdash^{(b_v^2 + b_{s'}, m_v^2 + m_{s'}, d_v^2 + d_{s'})} \texttt{upd}_l(v,s') : \{(l : \mathcal{M}_2)\}; \mathcal{S}}\ (\texttt{upd})$$

Then Φ' must be of the following form:

$$\dfrac{\Phi_{p\{x\backslash v\}} \rhd \Gamma_{p\{x\backslash v\}} \vdash^{(b'', m'', d'')} p\{x\backslash v\} : \{(l : \mathcal{M})\}; \mathcal{S} \gg \kappa \qquad \Phi_0}{\Gamma_{p\{x\backslash v\}} + (\Gamma_v^2 + \Gamma_{s'}) \vdash^{(b'' + b_v^2 + b_{s'}, m'' + m_v^2 + m_{s'}, d'' + d_v^2 + d_{s'})} (p\{x\backslash v\}, \texttt{upd}_l(v,s')) : \kappa}\ (\texttt{conf})$$

such that $\Gamma = \Gamma_{p\{x\backslash v\}} + (\Gamma_v^2 + \Gamma_{s'})$, $b' = b'' + b_v^2 + b_{s'}$, $m' = m'' + b_v^2 + b_{s'}$, and $d' = d'' + d_v^2 + d_{s'}$. By Lemma 5.2, there exist $\Phi_p \rhd \Gamma_p; x : \mathcal{M}_1 \vdash^{(b_p, m_p, d_p)} p : \{(l : \mathcal{M}_2)\}; \mathcal{S} \gg \kappa$ and $\Phi_v^1 \rhd \Gamma_v^1 \vdash^{(b_v^1, m_v^1, d_v^1)} v : \mathcal{M}_1$, such that $\Gamma_{p\{x\backslash v\}} = \Gamma_p + \Gamma_v^1$, $b'' = b_p + b_v^1$, $m'' = m_p + m_v^1$, and $d'' = d_p + d_v^1$. Therefore, we can build $\Phi_{\texttt{get}_l(\lambda x.p)}$ as follows:

$$\dfrac{\Phi_p \rhd \Gamma_p; x : \mathcal{M}_1 \vdash^{(b_p, m_p, d_p)} p : \{(l : \mathcal{M}_2)\}; \mathcal{S} \gg \kappa}{\Gamma_p \vdash^{(b_p, 1 + m_p, d_p)} \texttt{get}_l(\lambda x.p) : \{(l : \mathcal{M}_1 \sqcup \mathcal{M}_2)\}; \mathcal{S} \gg \kappa}\ (\texttt{get})$$

By Lemma 24, we have $\Phi_v \rhd \Gamma_v^1 + \Gamma_v^2 \vdash^{(b_v^1 + b_v^2, m_v^1 + m_v^2, d_v^1 + d_v^2)} v : \mathcal{M}_1 \sqcup \mathcal{M}_2$. Thus, we can build $\Phi_{\texttt{upd}_l(v,s')}$ as follows:

$$\dfrac{\Phi_v \rhd \Gamma_v^1 + \Gamma_v^2 \vdash^{(b_v^1 + b_v^2, m_v^1 + m_v^2, d_v^1 + d_v^2)} v : \mathcal{M}_1 \sqcup \mathcal{M}_2 \qquad \Phi_{s'} \rhd \Gamma_{s'} \vdash^{(b_{s'}, m_{s'}, d_{s'})} s' : \mathcal{S}}{(\Gamma_v^1 + \Gamma_v^2) + \Gamma_{s'} \vdash^{(b_v^1 + b_v^2 + b_{s'}, m_v^1 + m_v^2 + m_{s'}, d_v^1 + d_v^2 + d_{s'})} \texttt{upd}_l(v,s') : \{(l : \mathcal{M}_1 \sqcup \mathcal{M}_2)\}; \mathcal{S}}\ (\texttt{upd})$$

Finally, we can build Φ as follows:

$$\dfrac{\Phi_{\texttt{get}_l(\lambda x.p)} \qquad \Phi_{\texttt{upd}_l(v,s')}}{\Gamma_p + (\Gamma_v^1 + \Gamma_v^2) + \Gamma_{s'} \vdash^{(b_p + b_v^1 + b_v^2 + b_{s'}, 1 + m_p + m_v^1 + m_v^2 + m_{s'}, d_p + d_v^1 + d_v^2 + d_{s'})} (\texttt{get}_l(\lambda x.p), \texttt{upd}_l(v,s')) : \kappa}\ (\texttt{conf})$$

such that $b = b_p + b_v^1 + b_v^2 + b_{s'}$, $m = 1 + m_p + m_v^1 + m_v^2 + m_{s'}$, and $d = d_p + d_v^1 + d_v^2 + d_{s'}$. And we can conclude with $\Gamma = \Gamma_{p\{x\backslash v\}} + (\Gamma_v^2 + \Gamma_{s'}) = \Gamma_p + \Gamma_v^1 + \Gamma_v^2 + \Gamma_{s'}$, $b' = b'' + b_v^2 + b_{s'} = b_p + b_v^1 + b_v^2 + b_{s'} = b$, and $m' = m'' + m_v^2 + m_{s'} = m_p + m_v^1 + m_v^2 + m_{s'} = (1 + m_p + m_v^1 + m_v^2 + m_{s'}) - 1 = m - 1$, $d' = d'' + d_v^2 + d_{s'} = d_p + d_v^1 + d_v^2 + d_{s'} = d$.

– Case $(t,s) = (\texttt{set}_l(v,p), s) \to (p, \texttt{upd}_l(v,s)) = (u,q)$. Let Φ_0 be the following derivation:

$$\dfrac{\Phi_v \rhd \Gamma_v \vdash^{(b_v, m_v, d_v)} v : \mathcal{M} \qquad \Phi_s \rhd \Gamma_s \vdash^{(b_s, m_s, d_s)} s : \mathcal{S}}{\Gamma_v + \Gamma_s \vdash^{(b_v + b_s, m_v + m_s, d_v + d_s)} \texttt{upd}_l(v,s) : \{(l : \mathcal{M})\}; \mathcal{S}}\ (\texttt{upd})$$

Φ' must be of the following form:

$$\dfrac{\Phi_p \triangleright \Gamma_p \vdash^{(b_p,m_p,d_p)} p : \{(l:\mathcal{M})\};\mathcal{S} \gg \kappa \qquad \Phi_0}{\Gamma_p + (\Gamma_v + \Gamma_s) \vdash^{(b_p+b_v+b_s,m_p+m_v+m_s,d_p+d_v+d_s)} (p, \mathtt{upd}_l(v,s)) : \kappa}\ (\mathtt{conf})$$

such that $\Gamma = \Gamma_p + (\Gamma_v + \Gamma_s)$, $b' = b_p + b_v + b_s$, $m' = m_p + m_v + m_s$, and $d' = d_p + d_v + d_s$. Therefore, we can build Φ as follows:

$$\dfrac{\dfrac{\Phi_v \triangleright \Gamma_v \vdash^{(b_v,m_v,d_v,)} v : \mathcal{M} \qquad \Phi_p \triangleright \Gamma_p \vdash^{(b_p,m_p,d_p)} p : \{(l:\mathcal{M})\};\mathcal{S} \gg \kappa}{\Gamma_v + \Gamma_p \vdash^{(b_v+b_p,1+m_v+m_p,d_v+d_p)} \mathtt{set}_l(v,p) : \mathcal{S} \gg \kappa}\ (\mathtt{set}) \qquad \Phi_s}{(\Gamma_v + \Gamma_p) + \Gamma_s \vdash^{(b_v+b_p+b_s,1+m_v+m_p+m_s,d_v+d_p+d_s)} (\mathtt{set}_l(v,p), s) : \kappa}\ (\mathtt{conf})$$

Notice that the type environment of the conclusion is $(\Gamma_v + \Gamma_p) + \Gamma_s = \Gamma$, and the counters are as expected.

Theorem 1 (Quantitative Soundness and Completeness).

1. **(Soundness)** *If* $\Phi \triangleright \Gamma \vdash^{(b,m,d)} (t,s) : \kappa$ *tight, then there exists* (u,q) *such that* $u \in \mathtt{no}$ *and* $(t,s) \twoheadrightarrow^{(b,m)} (u,q)$ *with* b *β-steps,* m $\mathtt{g}/\mathtt{s}$*-steps, and* $|(u,q)| = d$.
2. **(Completeness)** *If* $(t,s) \twoheadrightarrow^{(b,m,d)} (u,q)$ *and* $u \in \mathtt{no}$*, then there exists* $\Phi \triangleright \Gamma \vdash^{(b,m,|(u,q)|)} (t,s) : \kappa$ *tight.*

Proof.

1. The proof follows by induction over $b + m$:
   - Case $b + m = 0$. Then $b = m = 0$, therefore $t \in \mathtt{no}$, by Lemma 1.1, and $d = |t|$, by Lemma 1.2. Let $u = t$ and $q = s$, then we can conclude since $|(u,q)| = |u| = |t| = d$.
   - Case $b + m > 0$. Then $b > 0$ or $m > 0$, and in either case $t \notin \mathtt{no}$, by Lemma 2. Note that (t,s) is not final because t is unblocked by Prop. 3. Therefore, by Prop. 2 there exists (t',s') such that $(t,s) \to (t',s')$. By Lemma 6.1, there exists $\Phi' \triangleright \Gamma \vdash^{(b',m',d)} (t',s') : \kappa$, such that $b' + m' = b + m - 1$. By the *i.h.*, there exists (u,q), such that $u \in \mathtt{no}$, $(t',s') \twoheadrightarrow^{(b',m')} (u,q)$ and $d = |(u,q)|$. So we can conclude with $(t,s) \to (t',s') \twoheadrightarrow^{(b',m')} (u,q)$, which means that $(t,s) \twoheadrightarrow^{(b,m)} (u,q)$, as expected.
2. By induction over $b + m$:
   - Case $b + m = 0$. Then $b = m = 0$ and $(t,s) = (u,q)$. We can conclude by Lemma 4.1 and Lemma 4.2.
   - Case $b + m > 0$. Then there exists (t',s'), such that $(t,s) \to^{(1,0)} (t',s') \twoheadrightarrow^{(b-1,m)} (u,q)$ or $(t,s) \to^{(0,1)} (t',s') \twoheadrightarrow^{(b,m-1)} (u,q)$. By the *i.h.*, there exists $\Phi' \triangleright \Gamma \vdash^{(b',m',|(u,q)|)} (t',s') : \kappa$ tight, such that $b' + m' = b + m - 1$. By Lemma 6.2, we have $\Phi \triangleright \Gamma \vdash^{(b'',m'',|(u,q)|)} (t,s) : \kappa$ tight, such that $b'' + m'' = 1 + b' + m'$. Therefore, $b'' + m'' = b + m$, since the fact that $b'' = b$, and $m'' = m$ can be easily checked by a simple case analysis.