Unifying Linearity and Dependency Analyses

Abstract. Linearity and dependency analyses are key to several applications in computer science. For example, restricting variable use in programs, ensuring secure information flow, etc. What connects these analyses is that both of them need to model two different worlds (linear and non-linear/low and high security) with constrained mutual interaction. However, for such modelling, type systems implementing these analyses use different methods. Linear type systems use the comonadic exponential modality from Girard's linear logic. Dependency type systems use the monadic modality from Moggi's computational metalanguage. Owing to this methodical difference, a unification of the two analyses, though theoretically and practically desirable, is not straightforward.
Fortunately, with recent advances in graded type systems, it has been realized that linearity and dependency analyses can be viewed through the same lens. However, the problem with existing graded type systems is that though their linearity analysis is general, their dependency analysis is limited, primarily because the graded modality employed is comonadic and not monadic. In this paper, we address this limitation by systematically extending existing graded type systems such that the graded modality is simultaneously comonadic and monadic. This extension enables us to unify linearity analysis with a general dependency analysis. We present a calculus, LDC, that analyses linearity and dependency using the same mechanism in an arbitrary Pure Type System. We show that LDC is a general linear and dependency calculus by subsuming in it the standard calculi for the individual analyses.

1 Introduction

Type systems formalize our intuition of correct programs: a correct program is one that is well-typed. Our intuition of correctness, however, varies depending upon application. For example: in a distributed system, correctness would entail absence of simultaneous write access to a file by multiple users; in a security system, correctness would entail absence of read access to secret files by public users. To formalize such notions of correctness, we employ linear type systems and dependency type systems. These type systems have a wide variety of applications.

Linear type systems are good at managing resource usage and sharing. In the form of session types, they provide useful guarantees like absence of deadlocks in distributed systems where resources like files, channels, etc. are shared among processes [14]. In functional programs, linear types can reason about state and enable compiler optimizations like in-place update of memory locations [40]. Owing to their utility, linear types have found their way into several programming languages like Haskell [10], Granule [33], Idris 2 [12], etc.

Dependency type systems are good at controlling information flow. In the form of security type systems [26,39], they guarantee that low-security outputs *do not depend upon* high-security inputs. In the form of binding-time type systems [25,18], they guarantee that early-bound expressions *do not depend upon* late-bound ones. Such type systems are commonly used in practice, for example, in metaprogramming languages like MetaOcaml [15], in static analysers like Jif extension of Java [32], etc.

Though linear type systems and dependency type systems serve different purposes, they essentially address the same abstract problem. The problem is to model two different worlds that interact following given constraints. Linear type systems need to model linear and non-linear worlds with the constraint that derivations in the non-linear world cannot depend upon assumptions from the linear world. Dependency type systems need to model low and high security worlds with the constraint that information from the high-security world cannot leak into the low-security world. This fundamental similarity suggests that linearity and dependency analyses could be unified.

There are several benefits to such a unification. First, from a theoretical perspective, it would unify the standard calculi for linearity analysis [8,5] with the standard calculi for dependency analysis [1,36]. Second, from a practical perspective, it would allow programmers to use the same type system for both linearity and dependency analyses. Presently, programmers use different type systems for this purpose. For example, Haskell programmers use Linear Haskell library [10] for linearity analysis and LIO library [37] for dependency analysis. Third, it would allow a combination of the two analyses. A combined analysis is more powerful than the individual analyses done separately because it allows arbitrary combination of usage and flow constraints. For example, in a combined analysis, a piece of data may be simultaneously linear and secure.

A unification of linear and dependency type systems, though desirable, is not straightforward. This is so because these type systems employ different methods to enforce their respective constraints. Linear type systems [3,9,5,38] employ the *comonadic* exponential modality, !, taken from linear logic [22], to manipulate non-linear resources in a base linear world. Dependency type systems [1,26,39] employ the *monadic* modality, T, taken from computational metalanguage [30], to manipulate high-security values in a base low-security world.

The modalities ! and T behave differently. For example: In a dependency calculus, there is no non-constant function of type $T\,\mathbf{Bool} \to \mathbf{Bool}$ because any such function would leak information. However, in a linear calculus [9], there exists a non-constant function, $\emptyset \vdash \lambda x.\mathbf{derelict}\ x : !\,\mathbf{Bool} \to \mathbf{Bool}$, since non-linear resources can be used linearly. Next, in a dependency calculus, $x : \mathbf{Bool} \vdash \eta\, x : T\,\mathbf{Bool}$ is a valid typing judgement because information can flow from low to high security world. However, in a linear calculus [3], $x : \mathbf{Bool} \vdash\!! x : !\,\mathbf{Bool}$ is not a valid typing judgement because non-linear resources cannot depend upon linear assumptions. Owing to these differences, for a long time, linearity and dependency analyses have been carried out independent of one another.

However, with recent advances in graded type systems [21,13,34,33,2], it has been realized that the two analyses can be viewed through the same lens. Graded type systems have their basis in bounded linear logic [23] that allows bounded use in lieu of just linear and non-linear use, as in linear logic. This makes graded type systems more general than linear type systems because by grading the !-modality, they can reason about linear use, unrestricted use, no use, bounded use, etc. For example, in a graded type system, unrestricted use and no use are expressed using $!_\omega$ and $!_0$ modalities respectively. Note that the standard ! and T modalities of linear and dependency calculi can be represented using $!_\omega$ and $!_0$ modalities respectively. This representation brings the two modalities under the same umbrella and gives us a perspective on the differences in their behaviour mentioned above.

Though existing graded type systems can analyse linearity well, they are limited in their dependency analysis. There are several aspects of dependency analysis that these systems cannot capture. We discuss them in detail in the next section. The main reason behind their shortcoming is that they have been designed for analysing coeffects, i.e. how programs depend upon their contexts. Coeffects include linearity (single use), irrelevance (no use), etc. Dependency, however, behaves more like an effect. To elaborate: low and high security computations may be seen as pure and effectful computations respectively. An effect like dependency is not well captured by existing graded type systems which are basically coeffect calculi.

In this paper, we show that by systematically extending existing graded type systems, we can use them for a general linearity and dependency analysis. We design a calculus, LDC, that can simultaneously analyse, using the same mechanism, a coeffect like linearity and an effect like dependency. LDC is parametrized by an arbitrary Pure Type System and it subsumes standard calculi for linearity and dependency analyses. We show that linearity and dependency analyses in LDC are correct using a heap semantics.

In summary, we make the following contributions:

- We present a language, LDC, parametrized by an arbitrary pure type system, that analyses linearity and dependency using the same mechanism.
- We show that LDC subsumes the standard calculi for analysing linearity and dependency like Linear Nonlinear λ-calculus of Benton [8], DCC of Abadi et. al. [1], Sealing Calculus of Shikuma and Igarashi [36], etc.
- We show that correctness of both linearity and dependency analyses in LDC follow from the soundness theorem for the calculus.
- We show that LDC can carry out a combined linearity and dependency analysis.

2 Challenges and Resolution

2.1 Dependency Analysis: Salient Aspects

In the previous section, we discussed dependency analysis with respect to high and low security worlds. Such an analysis can be extended to an arbitrary (fi-

nite) number of worlds with dependency constraints among them. Denning [19] observed that dependency constraints upon worlds result in a lattice structure. For example, consider the following set of worlds: a low-security world $\mathbf{L}$, two medium-security worlds $\mathbf{M_1}$ and $\mathbf{M_2}$ that do not share information between each other, and a high-security world $\mathbf{H}$. These constraints are modelled by a diamond lattice $\mathcal{L}_\diamond$ where $\mathbf{L} \sqsubseteq \mathbf{M_1} \sqsubseteq \mathbf{H}$ and $\mathbf{L} \sqsubseteq \mathbf{M_2} \sqsubseteq \mathbf{H}$, but $\mathbf{M_1} <> \mathbf{M_2}$. Note that any information flow that goes against the lattice order is illegal.

Dependency type systems [1,36] are based on this lattice model of information flow. These type systems grade the monadic modality, T, of Moggi's computational metalanguage with labels drawn from a dependency lattice. For example, for a dependency lattice $\mathcal{L}$ and $\ell \in \mathcal{L}$, $T_\ell\ \mathbf{Bool}$ would be the type of ℓ-secure booleans. To enable dependency analysis, this modality needs to support functions like $T_\ell\, T_\ell\, A \to T_\ell\, A$ and $T_{\ell_1}\, T_{\ell_2}\, A \to T_{\ell_1 \sqcup \ell_2}\, A$, for an arbitrary type A and $\ell_1, \ell_2, \ell \in \mathcal{L}$ (Here, $\sqcup$ is the join operator of the lattice). The modality T, being monadic, supports these functions through the standard join operation. Now, if dependency labels are seen as effects, then the join operation is computing the union of these effects. Just as computing union is important in an effect calculus, having a join operation is important in a dependency calculus. Later in this section, we shall see that existing graded type systems cannot derive a join operation. This significantly limits dependency analysis in these systems.

Before moving further, we want to point out another important aspect of dependency analysis that is sometimes ignored. It pertains to the treatment of functions that are wrapped under T_ℓ's. Consider the type: $T_{\mathbf{H}}(T_{\mathbf{H}}\mathbf{Bool} \to \mathbf{Bool})$. What should the values of this type be? DCC [1] would answer: just $\eta_{\mathbf{H}}(\lambda x.\mathbf{true})$ and $\eta_{\mathbf{H}}(\lambda x.\mathbf{false})$. But Sealing Calculus [36] would answer: $[\lambda x.\mathbf{true}]_{\mathbf{H}}$, $[\lambda x.\mathbf{false}]_{\mathbf{H}}$, $[\lambda x.x^{\mathbf{H}}]_{\mathbf{H}}$ and $[\lambda x.\mathbf{not}\ (x^{\mathbf{H}})]_{\mathbf{H}}$. This difference stems from the fact that in Sealing Calculus, the function $T_{\mathbf{H}}\ \mathbf{Bool} \to \mathbf{Bool}$, if wrapped under $T_{\mathbf{H}}$, may return a high-security output, unlike DCC where it must always be constant. In this regard, Sealing Calculus is more general than DCC because it doesn't restrict any function from returning high-security values, if the function itself is wrapped under $T_{\mathbf{H}}$. Note here that over terminating computations, Sealing Calculus subsumes DCC and is, in fact, more general than DCC as we see above. So we use the Sealing Calculus as our model of dependency analysis. As an aside, we allow nonterminating computations in our language even though Sealing Calculus does not.

2.2 Graded Type Systems: Salient Aspects

Over the recent years, graded type systems [21,13,34,20,29,4,33,2,17,31] have been successfully employed for quantitative reasoning in programs. As we pointed out earlier, these systems can carry out fine-grained quantitative analysis. The power and flexibility of these systems stem from the fact that they are parametrized by an abstract preordered semiring ($\mathcal{Q}$) or a similar structure that represents an algebra of resources. By varying $\mathcal{Q}$, these systems can carry out a variety of analyses. For example: instantiating $\mathcal{Q}$ to the set of natural numbers with discrete order, they can carry out an exact usage analysis; instantiating $\mathcal{Q}$ to

the set of natural numbers with descending natural order, they can carry out a bounded usage analysis; etc.

Graded type systems use the elements of the parametrizing preordered semiring to grade the !-modality. For example: for a semiring $\mathcal{Q}$ and $q \in \mathcal{Q}$, $!_q$ **Bool** would be the type of q-use booleans. Now, quantitative analysis requires splitting of resources through functions like $!_{q_1 \cdot q_2}\ A \to !_{q_1}\ !_{q_2}\ A$, for an arbitrary type A and $q_1, q_2 \in \mathcal{Q}$ (Here, $\cdot$ is the multiplication operator of the semiring). The !-modality, being comonadic, supports this function through the standard fork operation. However, since quantitative analysis does not require the inverse function, these systems do not derive a monadic join. As a matter of fact, these systems cannot derive a monadic join in general. Proposition 1 in Appendix A gives a model-theoretic argument showing why.

2.3 Limitations of Dependency Analysis in Graded Type Systems

Next, we point out the limitations of dependency analysis in existing graded type systems. First, as discussed above, these type systems are parametrized by preordered semirings. Dependency analysis, however, is parametrized by lattices. Both preordered semirings and lattices are algebraic structures with two binary operators and a binary order relation. However, a crucial distinction between these structures is that while semirings need to ensure that one operator (multiplication) distributes over the other (addition), lattices do not need to ensure any such property.

In fact, there are lattices where either operator does not distribute over the other. The simplest of such lattices is M_3 [11] that has 5 elements: $\bot, \ell_1, \ell_2, \ell_3, \top$, ordered as: $\bot \sqsubseteq l_i \sqsubseteq \top$, where $i = 1, 2, 3$. In M_3, join and meet do not distribute over one another. Thus, an arbitrary lattice can not be viewed as a preordered semiring. Distributive lattices, i.e. lattices where join and meet distribute over one another, however, may be viewed as preordered semirings where multiplication, addition and order are given by join, meet and the lattice order respectively. So, existing graded type systems could potentially carry out dependency analysis over distributive lattices. However, we view this as a limitation since we see no principled justification in restricting dependency analysis to distributive lattices only.

Second, as discussed above, dependency analysis needs a join operator. However, existing graded type systems cannot derive a join operator. An ad hoc solution to this problem might be to add an explicit join operator to the type system, for example, an operator **join** of type $!_{q_1}\ !_{q_2}\ A \to !_{q_1 \cdot q_2}\ A$ for any A and $q_1, q_2 \in \mathcal{Q}$. However, such a solution creates several problems. First, what should the semantics of this operator be? If v is a value, should **join** v be also a value? Then, we would have more values like **join** $!_{q_1}\ !_{q_2}$ **true**. Alternatively, if **join** v is not a value, how should it step? One might answer **join** $!_{q_1}\ !_{q_2}\ v$ should step to $!_{q_1 \cdot q_2}\ v$. While such a stepping rule would make sense in a call-by-value calculus, it would not in a call-by-name calculus because there $!_q\ a$ is a value, irrespective of whether a itself is a value or not. Second, one would expect **join** to be the inverse of fork that is already derivable in these systems. However, since **join** is

added as an ad hoc construct, there is no principled way to ensure this property. Third, the addition of this construct breaks the symmetry of the type system since the corresponding typing rule is neither an introduction rule nor an elimination rule for !. So we avoid this solution towards enabling dependency analysis in graded type systems.

Third, as discussed above, we want a dependency calculus where any function can return high-security values, whenever the function itself is wrapped under a high-security label. Existing graded type systems do not allow this. For a lattice, $\mathbf{L} \sqsubseteq \mathbf{H}$, the type $!_{\mathbf{H}}\,(!_{\mathbf{H}}\,\mathbf{Bool} \to \mathbf{Bool})$ contains only two distinct terms in these systems: $!_{\mathbf{H}}\,(\lambda x.\mathbf{true})$ and $!_{\mathbf{H}}\,(\lambda x.\mathbf{false})$.

Hence, we see that there are several limitations of dependency analysis in existing graded type systems. This motivates us to look for other solutions for unifying linearity and dependency analyses.

2.4 Towards Resolution

Recent work by Choudhury et. al. points towards a possible way of unifying the two analyses. Choudhury et. al. [16] present $\mathrm{DDC}^{\top}$, a type system for general dependency analysis in Pure Type Systems. $\mathrm{DDC}^{\top}$, though similar to existing graded type systems, avoids all their limitations listed above. It subsumes the Sealing Calculus and can analyse dependencies in a dependent setting, for example, runtime irrelevance in dependent programs. The key difference between $\mathrm{DDC}^{\top}$ and the graded type systems discussed above is in the form of their typing judgement. Graded type systems use a typing judgement of the form:

$$x_1 :^{q_1} A_1, x_2 :^{q_2} A_2, \ldots, x_n :^{q_n} A_n \vdash b : B,$$

where $q_i \in \mathcal{Q}$. On the other hand, $\mathrm{DDC}^{\top}$ uses a typing judgement of the form:

$$x_1 :^{\ell_1} A_1, x_2 :^{\ell_2} A_2, \ldots, x_n :^{\ell_n} A_n \vdash b :^{\ell} B,$$

where $\ell_i, \ell \in \mathcal{L}$. Note the extra label to the right of the turnstile! The label to the right symbolizes the observer's world in $\mathrm{DDC}^{\top}$. In graded type systems, the observer's world is fixed at 1. This added flexibility enables $\mathrm{DDC}^{\top}$ carry out a general dependency analysis.

However, $\mathrm{DDC}^{\top}$ cannot carry out linearity or for that matter, any quantitative analysis. So the problem of unifying linearity and dependency analyses still remains open. One possible solution might be to allow graded type systems to vary the observer's world using typing judgements of the form:

$$x_1 :^{q_1} A_1, x_2 :^{q_2} A_2, \ldots, x_n :^{q_n} A_n \vdash b :^{q} B \tag{1}$$

where $q_i, q \in \mathcal{Q}$. There is, however, a known roadblock that awaits us along this direction. Atkey [4] showed that a type system that uses the above judgement form and is parametrized by an arbitrary semiring does not admit substitution. But all is not lost! We find that though substitution is inadmissible over some semirings, it is in fact admissible over several preordered semirings that interest us. In particular, substitution is admissible over the standard

preordered semirings used for tracking linear and affine use. Both these semirings, denoted $\mathcal{Q}_{\text{Lin}}$ and $\mathcal{Q}_{\text{Aff}}$ respectively, have 3 elements: $0, 1$ and ω, with $1 + 1 = \omega + 1 = 1 + \omega = \omega + \omega = \omega$ and $\omega \cdot \omega = \omega$. However, $\mathcal{Q}_{\text{Lin}}$ is ordered as: $\omega <: 0$ and $\omega <: 1$ while $\mathcal{Q}_{\text{Aff}}$ is ordered as: $\omega <: 1 <: 0$.

Thus, we finally have a way to unify linearity and dependency analyses. In our Linearity Dependency Calculus, LDC, we use typing judgements of the form shown in (1). For linearity and other quantitative analyses, we parametrize LDC over certain preordered semirings that we describe as we go along. For dependency analysis, we parametrize LDC over an *arbitrary* lattice.

Now, LDC can analyse linearity and dependency in an arbitrary pure type system. However, some of the key ideas of the calculus are best explained in a simply-typed setting. So, we first present the simply-typed version of LDC and thereafter generalize it to its pure type system version.

3 Linearity and Dependency Analyses over Simple Types

3.1 Type System for Linearity Analysis

First, we analyse exact use and bounded use. We shall add unrestricted use, referred to by ω, to our calculus in Section 6. Exact use can be analysed by $\mathbb{N}_{=}$, the semiring of natural numbers with discrete order. Bounded use can be analysed by $\mathbb{N}_{\geq}$, the semiring of natural numbers with descending natural order. The ordering in $\mathbb{N}_{\geq}$ looks like: $\ldots <: 4 <: 3 <: 2 <: 1 <: 0$. The reason behind the difference in ordering is that in bounded use analysis, resources may be discarded. But note that resources are never copied in either of these analyses.

Next, we present LDC parametrized over these semirings. Whenever we need precision, we refer to LDC parametrized over an algebraic structure, $\mathcal{AS}$, as LDC($\mathcal{AS}$). The algebraic structure, $\mathcal{AS}$, may be either a preordered semiring or a lattice or in the case of combined analysis, their cartesian product.

Now, let $\mathcal{Q}_{\mathbb{N}}$ vary over $\{\mathbb{N}_{=}, \mathbb{N}_{\geq}\}$. The grammar of LDC($\mathcal{Q}_{\mathbb{N}}$) appears in Figure 1. The calculus has function types, weak and strong product types, sum types and a **Unit** type. Weak and strong product types correspond to multiplicative conjunction and additive conjunction of linear logic respectively. Sum type corresponds to additive disjunction. Observe that the function type and the weak product type are annotated with grades. Types ${}^{r}A \to B$ and ${}^{r}A_1 \times A_2$ are essentially $(!_r\, A) \to B$ and $(!_r\, A_1) \times A_2$ respectively. In lieu of ${}^{r}A \to B$ and ${}^{r}A_1 \times A_2$, we could have used unannotated function and weak product types along with a graded exponential modal type. However, we chose to present this way because it generalizes easily to the Pure Type System setting. For terms, we have the introduction and the elimination forms corresponding to these types. The terms are also annotated with grades that track resources used by them. Assumptions in the context also appear along with their allowed usages.

Next, we look at the type system. The type system appears in Figure 2. There are a few things to note:

$$
\begin{array}{ll}
\text{types}, A, B, C & ::= \mathbf{Unit} \mid {}^{r}A \to B \mid {}^{r}A_1 \times A_2 \mid A_1 \,\&\, A_2 \mid A_1 + A_2 \\
\text{terms}, a, b, c & ::= x \mid \lambda^{r} x : A.b \mid b\ a^{r} \\
& \mid \mathbf{unit} \mid \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b \mid (a_1{}^{r}, a_2) \mid \mathbf{let}_{q_0}\ (x^{r}, y) = a\ \mathbf{in}\ b \\
& \mid (a_1; a_2) \mid \mathbf{proj}_1\ a \mid \mathbf{proj}_2\ a \mid \mathbf{inj}_1\ a_1 \mid \mathbf{inj}_2\ a_2 \mid \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \\
\text{contexts}, \Gamma & ::= \emptyset \mid \Gamma, x :^{q} A
\end{array}
$$

Fig. 1. Grammar of LDC($\mathcal{Q}_{\mathbb{N}}$) (Simple Version)

- For a context Γ, we denote the underlying context without the grades by $\Delta := \lfloor \Gamma \rfloor$. To denote the vector of grades associated with the assumptions in Γ, we use $\overline{\Gamma}$.
- For contexts Γ_1 and Γ_2 such that $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor = \Delta$, we define $\Gamma := \Gamma_1 + \Gamma_2$ as the context with $\lfloor \Gamma \rfloor = \Delta$ and $\overline{\Gamma} = \overline{\Gamma_1} + \overline{\Gamma_2}$ (pointwise vector addition).
- For context Γ_0 and grade q, we define $\Gamma := q \cdot \Gamma_0$ as the context with $\lfloor \Gamma \rfloor = \lfloor \Gamma_0 \rfloor$ and $\overline{\Gamma} = q \cdot \overline{\Gamma_0}$ (scalar multiplication).
- For contexts Γ and Γ' such that $\lfloor \Gamma \rfloor = \lfloor \Gamma' \rfloor$, we say $\Gamma <: \Gamma'$ if and only if $\overline{\Gamma} <: \overline{\Gamma'}$ (pointwise order).
- For any context Γ, we implicitly assume that no two assumptions in Γ use the same variable.

The typing judgement $\Gamma \vdash a :^{q} A$ may be intuitively understood as: the resources in Γ can produce q copies of A. With this understanding, let us look at the typing rules.

Most of the rules are as expected. A point to note is that the elimination rules ST-LetPair, ST-LetUnit, and ST-Case have a side condition $q_0 <: 1$. The reason behind this condition is that for reducing any of these elimination forms, we first need to reduce the term a, implying that in any case, we need the resources for reducing at least one copy of a. Further, the grade q_0 makes these rules more flexible. For example, owing to this flexibility, in rule ST-LetPair, b may use $q \cdot q_0$ copies of y in lieu of just q copies of y. Another point to note is how the rules ST-SubL and ST-SubR help discard resources in LDC($\mathbb{N}_{\geq}$).

Now, let us look at a few examples of derivable and non-derivable terms in LDC($\mathcal{Q}_{\mathbb{N}}$). (To reduce complexity, we omit the domain types in the λs.)

$$
\begin{array}{c}
\emptyset \vdash \lambda^{1} x.x :^{1} {}^{1}A \to A \text{ but } \emptyset \nvdash \lambda^{0} x.x :^{1} {}^{0}A \to A \\
\emptyset \vdash \lambda^{1} x.x :^{2} {}^{1}A \to A \text{ but } \emptyset \nvdash \lambda^{1} x.(x^{2}, \mathbf{unit}) :^{1} {}^{1}A \to {}^{2}A \times \mathbf{Unit} \\
\emptyset \vdash \lambda^{1} x.(x; x) :^{1} {}^{1}A \to A \,\&\, A \text{ but } \emptyset \nvdash \lambda^{1} x.(x^{1}, x) :^{1} {}^{1}A \to {}^{1}A \times A
\end{array}
$$

On a closer look, we find that the non-derivable terms above can not use resources fairly. Consider the types of these terms: ${}^{0}A \to A$, ${}^{1}A \to {}^{2}A \times \mathbf{Unit}$ and ${}^{1}A \to {}^{1}A \times A$. Such types may be inhabited only if resources can be copied. Since we disallow copying, they are essentially uninhabited.

Note that in order to produce 0 copy of any term, we do not need any resource. So in the 0 world, any annotated type is inhabited, provided its unannotated

$\boxed{\Gamma \vdash a :^q A}$ *(Simple Version)*

$$\text{ST-Var}\ \frac{}{0 \cdot \Gamma_1, x :^q A, 0 \cdot \Gamma_2 \vdash x :^q A} \qquad \text{ST-Lam}\ \frac{\Gamma, x :^{q \cdot r} A \vdash b :^q B}{\Gamma \vdash \lambda^r x : A.b :^q {}^r A \to B} \qquad \text{ST-App}\ \frac{\Gamma_1 \vdash b :^q {}^r A \to B \quad \Gamma_2 \vdash a :^{q \cdot r} A \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 + \Gamma_2 \vdash b\ a^r :^q B}$$

$$\text{ST-WPair}\ \frac{\Gamma_1 \vdash a_1 :^{q \cdot r} A_1 \quad \Gamma_2 \vdash a_2 :^q A_2 \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 + \Gamma_2 \vdash (a_1{}^r, a_2) :^q {}^r A_1 \times A_2} \qquad \text{ST-LetPair}\ \frac{\Gamma_1 \vdash a :^{q \cdot q_0} {}^r A_1 \times A_2 \quad \Gamma_2, x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2 \vdash b :^q B \quad q_0 <: 1 \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^q B}$$

$$\text{ST-Unit}\ \frac{}{0 \cdot \Gamma \vdash \mathbf{unit} :^q \mathbf{Unit}} \qquad \text{ST-LetUnit}\ \frac{\Gamma_1 \vdash a :^{q \cdot q_0} \mathbf{Unit} \quad \Gamma_2 \vdash b :^q B \quad q_0 <: 1 \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b :^q B}$$

$$\text{ST-SPair}\ \frac{\Gamma \vdash a_1 :^q A_1 \quad \Gamma \vdash a_2 :^q A_2}{\Gamma \vdash (a_1; a_2) :^q A_1\ \&\ A_2} \qquad \text{ST-Proj1}\ \frac{\Gamma \vdash a :^q A_1\ \&\ A_2}{\Gamma \vdash \mathbf{proj}_1\ a :^q A_1} \qquad \text{ST-Proj2}\ \frac{\Gamma \vdash a :^q A_1\ \&\ A_2}{\Gamma \vdash \mathbf{proj}_2\ a :^q A_2}$$

$$\text{ST-Inj1}\ \frac{\Gamma \vdash a_1 :^q A_1}{\Gamma \vdash \mathbf{inj}_1\ a_1 :^q A_1 + A_2} \qquad \text{ST-Inj2}\ \frac{\Gamma \vdash a_2 :^q A_2}{\Gamma \vdash \mathbf{inj}_2\ a_2 :^q A_1 + A_2}$$

$$\text{ST-Case}\ \frac{\Gamma_1 \vdash a :^{q \cdot q_0} A_1 + A_2 \quad \Gamma_2, x_1 :^{q \cdot q_0} A_1 \vdash b_1 :^q B \quad \Gamma_2, x_2 :^{q \cdot q_0} A_2 \vdash b_2 :^q B \quad q_0 <: 1 \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 + \Gamma_2 \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B} \qquad \text{ST-SubL}\ \frac{\Gamma' \vdash a :^q A \quad \Gamma <: \Gamma'}{\Gamma \vdash a :^q A}$$

$$\text{ST-SubR}\ \frac{\Gamma \vdash a :^q A \quad q <: q'}{\Gamma \vdash a :^{q'} A}$$

Fig. 2. Typing rules for LDC($\mathcal{Q}_{\mathbb{N}}$)

counterpart is inhabited. However, resources do not have any meaning in the 0 world. In other words, the judgement $\Gamma \vdash a :^0 A$ conveys no more information than its corresponding standard λ-calculus counterpart.

Next, we look at the operational semantics and metatheory of LDC($\mathcal{Q}_{\mathbb{N}}$).

3.2 Metatheory of Linearity Analysis

First, we consider some syntactic properties. The calculus satisfies the multiplication lemma stated below. This lemma says that if we need Γ resources to produce q copies of a, then we would need $r_0 \cdot \Gamma$ resources to produce $r_0 \cdot q$ copies of a.

Lemma 1 (Multiplication). *If $\Gamma \vdash a :^q A$, then $r_0 \cdot \Gamma \vdash a :^{r_0 \cdot q} A$.*

Note that the proofs of all the lemmas and the theorems stated in the paper appear in the appendices.

The calculus also satisfies a factorization lemma, stated below. This lemma says that if a context Γ produces q copies of a, then Γ can be split into q parts whereby each part produces 1 copy of a. We need the precondition $q \neq 0$ since resources don't have any meaning in the 0 world but they are meaningful in all other worlds, in particular, the 1 world.

Lemma 2 (Factorization). *If $\Gamma \vdash a :^q A$ and $q \neq 0$, then there exists Γ' such that $\Gamma' \vdash a :^1 A$ and $\Gamma <: q \cdot \Gamma'$.*

Using the above two lemmas, we can prove a splitting lemma stated below. This lemma says that if we have the resources, Γ, to produce $q_1 + q_2$ copies of a, then we can split Γ into two parts, Γ_1 and Γ_2, such that Γ_1 and Γ_2 can produce q_1 and q_2 copies of a respectively. Atkey [4] showed that the splitting lemma does not hold for some semirings. However, it holds for the preordered semirings we use for parametrizing LDC, for example, any $\mathcal{Q}_{\mathbb{N}}$.

Lemma 3 (Splitting). *If $\Gamma \vdash a :^{q_1+q_2} A$, then there exists Γ_1 and Γ_2 such that $\Gamma_1 \vdash a :^{q_1} A$ and $\Gamma_2 \vdash a :^{q_2} A$ and $\Gamma = \Gamma_1 + \Gamma_2$.*

Next, we look at weakening and substitution. For substitution, the allowed usage of the variable should match the number of available copies of the substitute. Further, the term, after substitution, would need the combined resources.

Lemma 4 (Weakening). *If $\Gamma_1, \Gamma_2 \vdash a :^q A$, then $\Gamma_1, z :^0 C, \Gamma_2 \vdash a :^q A$.*

Lemma 5 (Substitution). *If $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$ and $\Gamma \vdash c :^{r_0} C$ and $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$, then $\Gamma_1 + \Gamma, \Gamma_2 \vdash a\{c/z\} :^q A$.*

Now, we consider the operational semantics of the language. Our operational semantics is a call-by-name small-step semantics. All the step rules are standard other than the two rules that appear in Figure 3. With this operational semantics, our language enjoys the standard type soundness property.

$\boxed{\vdash a \rightsquigarrow a'}$ *(Excerpt)*

$$\textsc{Step-AppBeta}\quad \frac{}{\vdash (\lambda^r x : A.b)\ a^r \rightsquigarrow b\{a/x\}} \qquad \textsc{Step-LetPairBeta}\quad \frac{}{\vdash \mathbf{let}_{q_0}\ (x^r, y) = ({a_1}^r, a_2)\ \mathbf{in}\ b \rightsquigarrow b\{a_1/x\}\{a_2/y\}}$$

Fig. 3. Small-step reduction for LDC($\mathcal{Q}_{\mathbb{N}}$)

Theorem 1 (Preservation). *If $\Gamma \vdash a :^q A$ and $\vdash a \rightsquigarrow a'$, then $\Gamma \vdash a' :^q A$.*

Theorem 2 (Progress). *If $\emptyset \vdash a :^q A$, then either a is a value or there exists a' such that $\vdash a \rightsquigarrow a'$.*

In standard small-step semantics, we don't substitute the free variables of a term. So such a semantics cannot really model resource usage of programs. Environment-based semantics, where free variables of terms get substituted with values from the environment, are more amenable to modelling resource usage by programs. In Section 4, we present an environment-based semantics for this calculus and show that the type system accounts usage correctly. But for now, we move on to dependency analysis.

3.3 Type System for Dependency Analysis

Let $\mathcal{L} = (L, \sqcap, \sqcup, \top, \bot, \sqsubseteq)$ be an arbitrary lattice. We use ℓ, m to denote elements of L. Now, interpreting $+, \cdot, 0, 1$ and $<:$ as $\sqcap, \sqcup, \top, \bot$ and $\sqsubseteq$ respectively, and using q and r for the elements of $\mathcal{L}$, we have a dependency calculus in the type system presented in Figure 2. In Figure 4, we present a few selected rules from this type system with just changed notation.

The type system is now parametrized by $\mathcal{L}$ in lieu of $\mathcal{Q}_{\mathbb{N}}$. We define $\ell \sqcup \Gamma$, $\Gamma_1 \sqcap \Gamma_2$ and $\Gamma \sqsubseteq \Gamma'$ in the same way as their counterparts $q \cdot \Gamma$, $\Gamma_1 + \Gamma_2$ and $\Gamma <: \Gamma'$ respectively. The typing judgement $x_1 :^{\ell_1} A_1, x_2 :^{\ell_2} A_2, \ldots, x_n :^{\ell_n} A_n \vdash b :^{\ell} B$ may be read as: b is visible at ℓ, assuming x_i is visible at ℓ_i for $i = 1, 2, \ldots, n$. With this reading, let us consider some of the typing rules.

The rule ST-VarD is as expected. The rule ST-LamD is interesting. The type ${}^m A \to B$ contains functions that take arguments which are m-secure, i.e. visible only at m and higher levels. The body of such a function, therefore, needs to be checked setting the visibility level of the argument at least as high as m.

Conversely in rule ST-AppD, the argument a needs to be visible at m or higher levels only. Observe that the context in the conclusion judgement is formed by taking pointwise meet of the contexts checking the function and the argument. This ensures that no subterm that is visible in the premises becomes invisible in the conclusion.

The rule ST-WPairD shows how to embox a secure term in a potentially insecure world. If a_1 is m-secure, then we can release it at level ℓ, but only after putting it in a box that may be opened at m and higher levels. This is similar to

$\boxed{\Gamma \vdash a :^{\ell} A}$ *(Selected rules)*

$$\text{ST-VarD}\ \frac{}{\top \sqcup \Gamma_1, x :^{\ell} A, \top \sqcup \Gamma_2 \vdash x :^{\ell} A} \qquad \text{ST-LamD}\ \frac{\Gamma, x :^{\ell \sqcup m} A \vdash b :^{\ell} B}{\Gamma \vdash \lambda^m x : A.b :^{\ell}\ ^m A \to B}$$

$$\text{ST-AppD}\ \frac{\Gamma_1 \vdash b :^{\ell}\ ^m A \to B \quad \Gamma_2 \vdash a :^{\ell \sqcup m} A \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 \sqcap \Gamma_2 \vdash b\ a^m :^{\ell} B} \qquad \text{ST-WPairD}\ \frac{\Gamma_1 \vdash a_1 :^{\ell \sqcup m} A_1 \quad \Gamma_2 \vdash a_2 :^{\ell} A_2 \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 \sqcap \Gamma_2 \vdash ({a_1}^m, a_2) :^{\ell}\ ^m A_1 \times A_2}$$

$$\text{ST-LetPairD}\ \frac{\Gamma_1 \vdash a :^{\ell \sqcup \ell_0}\ ^m A_1 \times A_2 \quad \Gamma_2, x :^{\ell \sqcup \ell_0 \sqcup m} A_1, y :^{\ell \sqcup \ell_0} A_2 \vdash b :^{\ell} B \quad \ell_0 \sqsubseteq \bot \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{let}_{\ell_0}\ (x^m, y) = a\ \mathbf{in}\ b :^{\ell} B} \qquad \text{ST-SubLD}\ \frac{\Gamma' \vdash a :^{\ell} A \quad \Gamma \sqsubseteq \Gamma'}{\Gamma \vdash a :^{\ell} A}$$

$$\text{ST-SubRD}\ \frac{\Gamma \vdash a :^{\ell} A \quad \ell \sqsubseteq \ell'}{\Gamma \vdash a :^{\ell'} A}$$

Fig. 4. Typing rules of LDC($\mathcal{L}$)

how the modal type $T_m\ A$ of DCC and Sealing Calculus, essentially ${}^m A \times \mathbf{Unit}$, protects information.

Conversely, the rule ST-LetPairD ensures that an m-secure box may be opened only at m and higher levels. Also, since $\bot$ is the bottom element, the side-condition $\ell_0 \sqsubseteq \bot$ forces ℓ_0 to be $\bot$. So rule ST-LetPairD may be simplified with ℓ_0 set to $\bot$. However, we present it in this way to emphasize the similarity between LDC($\mathcal{Q}_{\mathbb{N}}$) and LDC($\mathcal{L}$).

Next, we consider a few examples of derivable and non-derivable terms in LDC($\mathcal{L}_\diamond$), where $\mathcal{L}_\diamond$ is the diamond lattice (see Section 1).

Let $\mathbf{Bool} := \mathbf{Unit} + \mathbf{Unit}$ and let $\mathbf{true} := \mathbf{inj}_1\ \mathbf{unit}$ and $\mathbf{false} := \mathbf{inj}_2\ \mathbf{unit}$. Then,

$$\emptyset \vdash \lambda^{\mathbf{L}} x.x :^{\mathbf{H}\ \mathbf{L}} \mathbf{Bool} \to \mathbf{Bool} \text{ but } \emptyset \nvdash \lambda^{\mathbf{H}} x.x :^{\mathbf{L}\ \mathbf{H}} \mathbf{Bool} \to \mathbf{Bool}$$

$$\emptyset \vdash \lambda^{\mathbf{H}} x.\eta_{\mathbf{M_1}}\ \eta_{\mathbf{M_2}}\ x :^{\mathbf{L}\ \mathbf{H}} \mathbf{Bool} \to T_{\mathbf{M_1}}\ T_{\mathbf{M_2}}\ \mathbf{Bool} \text{ but } x :^{\mathbf{H}}\ \mathbf{Bool} \nvdash x :^{\mathbf{M_1}}\ \mathbf{Bool}$$

Here, $T_m\ A \triangleq {}^m A \times \mathbf{Unit}$ and $\eta_\ell\ a \triangleq (a^\ell, \mathbf{unit})$. On a closer look, we find that the non-derivable terms violate the dependence structure of $\mathcal{L}_\diamond$. The first term transfers information from **H** to **L** while the second one does so from **H** to $\mathbf{M_1}$. The derivable terms, on the other hand, respect the dependence structure of $\mathcal{L}_\diamond$. The first term transfers information from **L** to **H** while the second one emboxes **H** information in an $\mathbf{M_2}$ box nested within an $\mathbf{M_1}$ box.

Now, we present the terms that witness the standard join and fork operations in LDC($\mathcal{L}$). (Note that any erased annotation is assumed to be $\bot$.)

$\emptyset \vdash c_1 : T_{\ell_1}\ T_{\ell_2}\ A \to T_{\ell_1 \sqcup \ell_2}\ A$ and $\emptyset \vdash c_2 : T_{\ell_1 \sqcup \ell_2}\ A \to T_{\ell_1}\ T_{\ell_2}\ A$ where,

$c_1 := \lambda x.\eta_{\ell_1 \sqcup \ell_2}\ (\mathbf{let}\ (y^{\ell_2}, \_) = (\mathbf{let}\ (z^{\ell_1}, \_) = x\ \mathbf{in}\ z)\ \mathbf{in}\ y)$

$c_2 := \lambda x.\eta_{\ell_1}\ \eta_{\ell_2}\ (\mathbf{let}\ (y^{\ell_1 \sqcup \ell_2}, \_) = x\ \mathbf{in}\ y)$

The derivations of these terms appear in Appendix J.

Next, we look at the metatheory of LDC($\mathcal{L}$).

3.4 Metatheory of Dependency Analysis

We consider the dependency counterparts of the properties of LDC($\mathcal{Q}_{\mathbb{N}}$), presented in Section 3.2. Most of these properties are true of LDC($\mathcal{L}$). LDC($\mathcal{L}$) satisfies the multiplication lemma. The lemma says that we can always simultaneously upgrade the context and the level at which the derived term is viewed.

Lemma 6 (Multiplication). *If $\Gamma \vdash a :^{\ell} A$, then $m_0 \sqcup \Gamma \vdash a :^{m_0 \sqcup \ell} A$.*

The splitting lemma is also true. Since $\sqcap$ is idempotent, it follows directly from rule ST-SubRD.

Lemma 7 (Splitting). *If $\Gamma \vdash a :^{q_1 \sqcap q_2} A$, then there exists Γ_1 and Γ_2 such that $\Gamma_1 \vdash a :^{q_1} A$ and $\Gamma_2 \vdash a :^{q_2} A$ and $\Gamma = \Gamma_1 \sqcap \Gamma_2$.*

The factorization lemma, however, is not true here because if true, it would allow information to leak. To see how, consider the following LDC($\mathcal{L}_{\diamond}$) judgement: $\emptyset \vdash \lambda x.\mathbf{let}\ (y^{\mathbf{M_1}}, \_) = x\ \mathbf{in}\ y :^{\mathbf{M_1}} T_{\mathbf{M_1}}\ \mathbf{Bool} \to \mathbf{Bool}$. If the factorization lemma were true, this judgement would give us: $\emptyset \vdash \lambda x.\mathbf{let}\ (y^{\mathbf{M_1}}, \_) = x\ \mathbf{in}\ y :^{\mathbf{L}} T_{\mathbf{M_1}}\ \mathbf{Bool} \to \mathbf{Bool}$, a non-constant function from $T_{\mathbf{M_1}}\ \mathbf{Bool}$ to $\mathbf{Bool}$ in an $\mathbf{L}$-secure world, representing a leak of information. Note that in LDC($\mathcal{Q}_{\mathbb{N}}$), the factorization lemma is required for proving the splitting lemma which, in turn, is necessary to show substitution. In LDC($\mathcal{L}$), the splitting lemma holds trivially and does not need any factorization lemma.

Next, we consider weakening and substitution. The substitution lemma substitutes an assumption held at m_0 with a term derived at m_0.

Lemma 8 (Weakening). *If $\Gamma_1, \Gamma_2 \vdash a :^{\ell} A$, then $\Gamma_1, z :^{\top} C, \Gamma_2 \vdash a :^{\ell} A$.*

Lemma 9 (Substitution). *If $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a :^{\ell} A$ and $\Gamma \vdash c :^{m_0} C$ and $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$, then $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash a\{c/z\} :^{\ell} A$.*

Now, LDC($\mathcal{L}$) can be given the exact operational semantics as LDC($\mathcal{Q}_{\mathbb{N}}$). With respect to this operational semantics, LDC($\mathcal{L}$) enjoys the standard type soundness property.

Theorem 3 (Preservation). *If $\Gamma \vdash a :^{\ell} A$ and $\vdash a \rightsquigarrow a'$, then $\Gamma \vdash a' :^{\ell} A$.*

Theorem 4 (Progress). *If $\emptyset \vdash a :^{\ell} A$, then either a is a value or there exists a' such that $\vdash a \rightsquigarrow a'$.*

These theorems are not strong enough to show that LDC($\mathcal{L}$) analyses dependencies correctly. For that, we need to show the calculus passes the noninterference test. The noninterference test [24] ensures that if $\neg(\ell_1 \sqsubseteq \ell_2)$, then variations in ℓ_1-inputs do not affect ℓ_2-outputs. In Section 4, we shall prove that LDC($\mathcal{L}$) passes the noninterference test. But before that, we discuss the relation between Sealing Calculus and LDC($\mathcal{L}$).

3.5 Sealing Calculus and LDC

The Sealing Calculus [36] embeds into LDC($\mathcal{L}$). We don't provide details of the embedding here because LDC($\mathcal{L}$) is the same as SDC of Choudhury et. al. [16]. Choudhury et. al. [16] show that the Sealing Calculus embeds into SDC. Note here that SDC can only be parametrized over lattices and not over semirings that help track linearity.

A technical difference between LDC($\mathcal{L}$) and SDC is that SDC does not have security annotations on function and product types. However, SDC has modal types, $T_\ell\ A$. Annotated function and product types of LDC($\mathcal{L}$), ${}^mA \to B$ and ${}^mA \times B$, correspond to types $T_m\ A \to B$ and $T_m\ A \times B$ respectively in SDC. Conversely, the modal type of SDC, $T_m\, A$, corresponds to ${}^mA \times \mathbf{Unit}$ in LDC($\mathcal{L}$).

Given that Sealing Calculus is a general dependency calculus that embeds into LDC($\mathcal{L}$), we conclude that LDC($\mathcal{L}$) is also a general dependency calculus. Thus, LDC can be used for both usage and dependency analyses by parametrizing the calculus over appropriate structures. In the next section, we prove correctness of these analyses in LDC through a heap semantics for the calculus.

4 Heap Semantics for LDC

LDC models usage of resources and flow of information. Standard operational semantics cannot enforce constraints on usage and flow. However, an environment-based semantics, for example a heap semantics, can do that. In this section, we present a weighted-heap-based semantics for LDC. We use the same heap semantics for analysing both resource usage and information flow in LDC, just as we used the same standard small-step semantics for both the analyses.

Heap-based semantics have been used for analysing resource usage in literature [38,17,28]. Our heap semantics follows Choudhury et. al. [17]. The difference between our heap semantics and that of Choudhury et. al. [17] is that we use it for both usage and flow analyses while they use it for usage analysis only. Other than this difference, both the semantics are essentially the same.

Heap semantics shows how a term reduces in a heap that assigns values to the free variables of the term. Heaps are ordered lists of variable-term pairs where the terms may be seen as the definitions of the corresponding variables. To every variable-term pair in a heap, we assign a weight, which may be either a $q \in \mathcal{Q}_{\mathbb{N}}$

or an $\ell \in \mathcal{L}$. A heap where every variable-term pair has a weight associated with it is referred to as a weighted heap. We assume our weighted heaps to be unique, i.e. a variable is not defined twice and acyclic, i.e. definition of a variable does not refer to itself or to other variables appearing subsequently in the heap. We model reduction as an interaction between a term and a weighted heap that defines the free variables of the term.

4.1 Reduction Relation

The heap-based reduction rules appear in Figure 5. There are a few things to note with regard to this reduction relation $[H]a \Longrightarrow_S^q [H']a'$:

- S here denotes a support set [35] of variables that must be avoided while choosing fresh names.
- From a resource usage perspective, the judgement above may be read as: q copies of a use resources from heap H to produce q copies of a' with H' being the left-over resources. Regarding a heap as a memory, usage of resources correspond to the number of memory look-ups during reduction.
- From an information flow perspective, the judgement may be read as: under the security constraints of H, the term a steps to a' at security level q with H' being the updated security constraints. Regarding a heap as a memory, security labels on assignments correspond to access permissions on data while the label on the judgement corresponds to the security clearance of the user.

Heap, $H ::= \emptyset \mid H, x \overset{q}{\mapsto} a$

$\boxed{[H]a \Longrightarrow_S^q [H']a'}$ *(Selected Heap Step Rules)*

$$\textsc{HeapStep-Var}\ \frac{q \neq 0}{[H_1, x \overset{r+q}{\mapsto} a, H_2]x \Longrightarrow_S^q [H_1, x \overset{r}{\mapsto} a, H_2]a} \qquad \textsc{HeapStep-Discard}\ \frac{[H]a \Longrightarrow_S^q [H']a' \quad q <: q'}{[H]a \Longrightarrow_S^{q'} [H']a'}$$

$$\textsc{HeapStep-AppL}\ \frac{[H]b \Longrightarrow_{S \cup fv\ a}^q [H']b'}{[H]b\ a^r \Longrightarrow_S^q [H']b'\ a^r} \qquad \textsc{HeapStep-AppBeta}\ \frac{y\ \textit{fresh} \qquad b' = b\{y/x\}}{[H](\lambda^r x : A.b)\ a^r \Longrightarrow_S^q [H, y \overset{q \cdot r}{\mapsto} a]b'}$$

$$\textsc{HeapStep-LetPairBeta}\ \frac{\begin{array}{c} x'\ \textit{fresh} \qquad y'\ \textit{fresh} \\ b' = b\{x'/x\}\{y'/y\} \end{array}}{[H]\mathbf{let}_{q_0}\ (x^r, y) = ({a_1}^r, a_2)\ \mathbf{in}\ b \Longrightarrow_S^q [H, x'\ ^{q \cdot q_0 \cdot r}\!\!\mapsto a_1, y'\ \overset{q \cdot q_0}{\mapsto} a_2]b'}$$

Fig. 5. Heap Semantics for LDC (Excerpt)

Now, we consider some of the rules presented in Figure 5. The most interesting of the rules is rule HeapStep-Var. From a resource usage perspective, this rule may be read as: to step q copies of x, we need to look-up the value of x for q times, thereby using up q resources. From an information flow perspective, this rule may be read as: the data pointed to by x, residing at security level $r \sqcap q$, is visible to q since $r \sqcap q \sqsubseteq q$. Note that since $r \sqcap q \sqsubseteq r$, the security level of the assignment cannot go down in the updated heap. However, it can always remain the same because $(r \sqcap q) \sqcap q = r \sqcap q$. This rule includes the precondition $q \neq 0$ because at world 0, usage and flow constraints are not meaningful.

The rule HeapStep-Discard is also interesting. From a resource usage perspective, it enables discarding of resources, whenever permitted. From an information flow perspective, it corresponds to information remaining visible to an observer as the observer's security clearance goes up.

With these rules, let us consider some reductions that go through and some that don't.

$[x \overset{1}{\mapsto} \mathbf{true}]x \Longrightarrow^{1}_{S} [x \overset{0}{\mapsto} \mathbf{true}]\mathbf{true}$ but not $[x \overset{0}{\mapsto} \mathbf{true}]x \Longrightarrow^{1}_{S} [x \overset{0}{\mapsto} \mathbf{true}]\mathbf{true}$

$[x \overset{2}{\mapsto} \mathbf{true}]x \Longrightarrow^{2}_{S} [x \overset{0}{\mapsto} \mathbf{true}]\mathbf{true}$ but not $[x \overset{1}{\mapsto} \mathbf{true}]x \Longrightarrow^{2}_{S} [x \overset{0}{\mapsto} \mathbf{true}]\mathbf{true}$

$[x \overset{\mathbf{L}}{\mapsto} \mathbf{true}]x \Longrightarrow^{\mathbf{M_1}}_{S} [x \overset{\mathbf{L}}{\mapsto} \mathbf{true}]\mathbf{true}$ but not $[x \overset{\mathbf{M_2}}{\mapsto} \mathbf{true}]x \Longrightarrow^{\mathbf{L}}_{S} [x \overset{\mathbf{M_2}}{\mapsto} \mathbf{true}]\mathbf{true}$

4.2 Correctness of Usage and Flow

Looking at the rules in Figure 5, we observe that they enforce fairness of resource usage. The only rule that allows usage of resources is rule HeapStep-Var. This rule ensures that a look-up goes through only when the environment can provide adequate resources. It also takes away the necessary resources from the environment after a successful look-up. The rules in Figure 5 also ensure security of information flow. The only rule that allows information to flow from heap to program is again rule HeapStep-Var. This rule ensures that information can flow through only when the user has the necessary permission.

The following two lemma formalize the arguments presented above. The first lemma says that a definition that is not available at some point during reduction does not become available at a later point. The second lemma says that an unavailable definition does not play any role in reduction. Note that in case of information flow, the constraint $\neg(\exists q_0, r = q + q_0)$ is equivalent to $\neg(r \sqsubseteq q)$. (Here, $|H|$ denotes the length of H.)

Lemma 10 (Unchanged). *If* $[H_1, x \overset{r}{\mapsto} a, H_2]c \Longrightarrow^{q}_{S} [H_1', x \overset{r'}{\mapsto} a, H_2']c'$ *(where* $|H_1| = |H_1'|$*) and* $\neg(\exists q_0, r = q + q_0)$*, then* $r' = r$.

Lemma 11 (Irrelevant). *If* $[H_1, x \overset{r}{\mapsto} a, H_2]c \Longrightarrow^{q}_{S \cup fv\, b} [H_1', x \overset{r'}{\mapsto} a, H_2']c'$ *(where* $|H_1| = |H_1'|$*) and* $\neg(\exists q_0, r = q + q_0)$*, then* $[H_1, x \overset{r}{\mapsto} b, H_2]c \Longrightarrow^{q}_{S \cup fv\, a}$ $[H_1', x \overset{r'}{\mapsto} b, H_2']c'$.

These lemmas, in conjunction with the soundness theorem we present next, show correctness of usage and flow analyses in LDC.

4.3 Soundness with respect to Heap Semantics

The key idea behind usage analysis through heap semantics is that, if a heap contains the right amount of resources to evaluate some number of copies of a term, as judged by the type system, then the evaluation of that many number of copies of the term in that heap does not get stuck. Since the heap-based reduction rules enforce fairness of resource usage, this would mean that the type system accounts resource usage correctly.

The key idea behind dependency analysis through heap semantics is similar. If a heap sets the right access permissions for a user, as judged by the type system, then the evaluation, in that heap, of any program visible to that user does not get stuck. Since the reduction rules enforce security of information flow, this would mean that the type system allows only secure flows.

The compatibility relation, $H \models \Gamma$, between a heap H and a context Γ, formalizes the idea that the heap H contains the right amount of resources or has set the right access permissions for evaluating any term type-checked in context Γ. The compatibility relation [17] is defined below:

$\boxed{H \models \Gamma}$ *(Compatibility)*

$$\textsc{HeapCompat-Empty}\quad \frac{}{\emptyset \models \emptyset} \qquad \textsc{HeapCompat-Cons}\quad \frac{H \models \Gamma_1 + \Gamma_2 \quad \Gamma_2 \vdash a :^q A}{H, x \overset{q}{\mapsto} a \models \Gamma_1, x :^q A}$$

The soundness theorem stated next says that if a heap H is compatible with a context Γ, then the evaluation, starting with heap H, of a term type-checked in context Γ does not get stuck.

Theorem 5 (Soundness). *If $H \models \Gamma$ and $\Gamma \vdash a :^q A$ and $q \neq 0$, then either a is a value or there exists H', Γ', a' such that $[H]a \Longrightarrow_S^q [H']a'$ and $H' \models \Gamma'$ and $\Gamma' \vdash a' :^q A$.*

Below, we present some corollaries of this theorem.

Lemma 12 (No Use). *In LDC($\mathcal{Q}_{\mathbb{N}}$): Let $\emptyset \vdash f :^1 {}^0A \to A$. Then, for any $\emptyset \vdash a_1 :^0 A$ and $\emptyset \vdash a_2 :^0 A$, the terms $f\ {a_1}^0$ and $f\ {a_2}^0$ have the same operational behaviour.*

The above lemma also holds in LDC($\mathcal{L}$) with 0 and 1 replaced by **H** and **L** respectively. In LDC($\mathcal{L}$), this lemma shows *non-interference* of high-security inputs in low-security outputs.

Lemma 13 (Single Use). *In LDC($\mathbb{N}_{\geq}$): Let $\emptyset \vdash f :^1 {}^1A \to A$. Then, for any $\emptyset \vdash a :^1 A$, the term $f\ a^1$ uses a at most once during reduction.*

Now that we have seen the syntax and semantics of simply-typed version of LDC, we move on to its Pure Type System (PTS) version.

5 Linearity and Dependency Analyses in PTS

A Pure Type System (PTS) is characterized by a tuple, $(\mathcal{S}, \mathcal{A}, \mathcal{R})$, where $\mathcal{S}$ is a set of sorts, $\mathcal{A}$ is a set of axioms and $\mathcal{R}$ is a ternary relation between sorts [6]. Many type systems like simply-typed λ-calculus, System F, System F^{ω}, Calculus of Constructions, Type-in-Type, etc. may be seen as PTSs. Note that a PTS need not be normalizing, for example, Type-in-Type allows nonterminating computations. We parametrize LDC over an abstract PTS so that it may be instantiated to particular PTSs as required.

5.1 Simple Version Vs PTS Version

The PTS version of LDC is similar to its simply-typed version. As far as types and terms are concerned, we just need to add to them the sorts in $\mathcal{S}$ and generalize ${}^{r}A \to B, {}^{r}A \times B$ and $A \,\&\, B$ to $\Pi x :^{r} A.B, \Sigma x :^{r} A.B$ and $(x : A) \,\&\, B$ respectively. For resource usage and information flow analyses, we use the same parametrizing structures, i.e. $\mathcal{Q}_{\mathbb{N}}$ and $\mathcal{L}$ respectively. However, there is an important distinction between these two versions. In the PTS version, we need to extend our analyses from terms to both types and terms.

The key idea behind our extension is that the analysis for types and terms can be carried out separately. This idea is inspired by recent developments in graded dependent type systems [29,4,17,31]. McBride [29] first noted that linearity and dependent types can be smoothly combined by distinguishing between 'contemplative use' of resources in types and 'consumptive use' of resources in terms. Atkey [4] carried this work forward with the calculus QTT, where types live in a resource-agnostic world and terms live in a resource-aware world. Choudhury et. al. [17] presented an alternative system, GraD, where both types and terms live in a resource-aware world but resources used by a type are zeroed-out while calculating resources used by terms of that type. Moon et. al. [31] presented yet another alternative system, Grtt, where resources used by types and terms are tracked simultaneously but separately. In its analysis, GraD is more uniform than QTT and much simpler than Grtt. So LDC analyses usage in types à la GraD.

We now look at the type system of the calculus.

5.2 Type System of LDC

The typing and equality rules appear in Figures 6 and 7 respectively. There are a few things to note:

- We use these rules for both resource usage and information flow analyses.
- The judgment $\Delta \vdash_0 a : A$ is shorthand for the judgement $\Gamma \vdash a :^0 A$ where $\lfloor \Gamma \rfloor = \Delta$ and $\overline{\Gamma}$ is a 0 vector. Note that this judgement is essentially the standard typing judgement $\Delta \vdash a : A$ because in world 0, neither resource usage nor information flow constraints are meaningful.

$\boxed{\Gamma \vdash a :^q A}$ *(PTS version)*

$$\text{PTS-Axiom}\ \frac{c : s \in \mathcal{A}}{\emptyset \vdash c :^q s} \qquad \text{PTS-Var}\ \frac{\Delta \vdash_0 A : s \qquad \lfloor \Gamma \rfloor = \Delta}{0 \cdot \Gamma, x :^q A \vdash x :^q A} \qquad \text{PTS-Weak}\ \frac{\Gamma \vdash a :^q A \qquad \Delta \vdash_0 B : s \qquad \lfloor \Gamma \rfloor = \Delta}{\Gamma, y :^0 B \vdash a :^q A}$$

$$\text{PTS-Pi}\ \frac{\Gamma_1 \vdash A :^q s_1 \quad \Gamma_2, x :^{r_0} A \vdash B :^q s_2 \quad \mathcal{R}(s_1, s_2, s_3) \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 + \Gamma_2 \vdash \Pi x :^r A.B :^q s_3} \qquad \text{PTS-Lam}\ \frac{\Gamma, x :^{q \cdot r} A \vdash b :^q B \quad \Delta \vdash_0 \Pi x :^r A.B : s \quad \lfloor \Gamma \rfloor = \Delta}{\Gamma \vdash \lambda^r x : A.b :^q \Pi x :^r A.B}$$

$$\text{PTS-App}\ \frac{\Gamma_1 \vdash b :^q \Pi x :^r A.B \quad \Gamma_2 \vdash a :^{q \cdot r} A \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 + \Gamma_2 \vdash b\ a^r :^q B\{a/x\}} \qquad \text{PTS-Conv}\ \frac{\Gamma \vdash a :^q A \quad \Delta \vdash_0 B : s \quad A =_\beta B \quad \lfloor \Gamma \rfloor = \Delta}{\Gamma \vdash a :^q B} \qquad \text{PTS-SubL}\ \frac{\Gamma' \vdash a :^q A \qquad \Gamma <: \Gamma'}{\Gamma \vdash a :^q A}$$

$$\text{PTS-SubR}\ \frac{\Gamma \vdash a :^q A \qquad q <: q'}{\Gamma \vdash a :^{q'} A} \qquad \text{PTS-WPair}\ \frac{\Delta \vdash_0 \Sigma x :^r A_1.A_2 : s \quad \Gamma_1 \vdash a_1 :^{q \cdot r} A_1 \quad \Gamma_2 \vdash a_2 :^q A_2\{a_1/x\} \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor = \Delta}{\Gamma_1 + \Gamma_2 \vdash (a_1{}^r, a_2) :^q \Sigma x :^r A_1.A_2}$$

$$\text{PTS-LetPair}\ \frac{\Delta, z : \Sigma x :^r A_1.A_2 \vdash_0 B : s \quad \Gamma_1 \vdash a :^{q \cdot q_0} \Sigma x :^r A_1.A_2 \quad \Gamma_2, x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2 \vdash b :^q B\{(x^r, y)/z\} \quad q_0 <: 1 \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor = \Delta}{\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^q B\{a/z\}}$$

$$\text{PTS-SPair}\ \frac{\Delta \vdash_0 (x : A_1)\ \&\ A_2 : s \quad \Gamma \vdash a_1 :^q A_1 \quad \Gamma \vdash a_2 :^q A_2\{a_1/x\} \quad \lfloor \Gamma \rfloor = \Delta}{\Gamma \vdash (a_1; a_2) :^q (x : A_1)\ \&\ A_2}$$

$$\text{PTS-Proj1}\ \frac{\Delta \vdash_0 (x : A_1)\ \&\ A_2 : s \quad \Gamma \vdash a :^q (x : A_1)\ \&\ A_2 \quad \lfloor \Gamma \rfloor = \Delta}{\Gamma \vdash \mathbf{proj}_1\ a :^q A_1} \qquad \text{PTS-Sum}\ \frac{\Gamma_1 \vdash A_1 :^q s \quad \Gamma_2 \vdash A_2 :^q s \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 + \Gamma_2 \vdash A_1 + A_2 :^q s} \qquad \text{PTS-Inj1}\ \frac{\Delta \vdash_0 A_1 + A_2 : s \quad \Gamma \vdash a_1 :^q A_1 \quad \lfloor \Gamma \rfloor = \Delta}{\Gamma \vdash \mathbf{inj}_1\ a_1 :^q A_1 + A_2}$$

$$\text{PTS-Case}\ \frac{\Delta, z : A_1 + A_2 \vdash_0 B : s \quad \Gamma_1 \vdash a :^{q \cdot q_0} A_1 + A_2 \quad \Gamma_2, x_1 :^{q \cdot q_0} A_1 \vdash b_1 :^q B\{\mathbf{inj}_1\ x_1/z\} \quad \Gamma_2, x_2 :^{q \cdot q_0} A_2 \vdash b_2 :^q B\{\mathbf{inj}_2\ x_2/z\} \quad q_0 <: 1 \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor = \Delta}{\Gamma_1 + \Gamma_2 \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B\{a/z\}}$$

Fig. 6. Type System for LDC (Excerpt)

$\boxed{A =_\beta B}$ *(Definitional Equality)*

$$\textsc{Eq-PiCong}\ \frac{A_1 =_\beta A_2 \qquad B_1 =_\beta B_2}{\Pi x :^r A_1.B_1 =_\beta \Pi x :^r A_2.B_2} \qquad \textsc{Eq-LamCong}\ \frac{A_1 =_\beta A_2 \qquad b_1 =_\beta b_2}{\lambda^r x : A_1.b_1 =_\beta \lambda^r x : A_2.b_2}$$

$$\textsc{Eq-AppCong}\ \frac{b_1 =_\beta b_2 \qquad a_1 =_\beta a_2}{b_1\ {a_1}^r =_\beta b_2\ {a_2}^r}$$

Fig. 7. Equality Rules for LDC (Excerpt)

- We track usage and flow in terms and types separately. The rule PTS-Var illustrates this principle nicely. The type A may use some resources or be visible at some low security level. But while type-checking a term of type A, we zero-out the requirements of A or set A to the highest security level. This principle also applies to several other rules, for example, rule PTS-Lam, rule PTS-WPair, etc.
- The rule PTS-Pi shows how we track usage and flow in types. This rule brings out an important aspect of our analysis: not only do we separate the analysis in types and terms but also we allow a term and its type to treat the same bound variable differently. The annotation on the type, r in this case, shows how the bound variable is used in the body of a term having that type. This annotation is *not* related to how the bound variable is used in the body of the type itself.
  Let us consider an example: the polymorphic identity type, $\Pi x :^0 s.\Pi y :^1 x.x$, uses the bound variable x in its body but a function having this type (e.g. polymorphic identity function $\lambda^0 x : s.\lambda^1 y : x.y$) can not use the bound variable x in its body.
- We use β-equivalence for equality in rule PTS-Conv. It is a congruent, equivalence relation closed under β-reduction of terms. Some of the equality rules appear in Figure 7. They are mostly standard. However, the congruence rules Eq-PiCong, Eq-LamCong, and Eq-AppCong need to check that the grade annotations on the terms being equated match up.

Next, we look at the metatheory of the calculus.

5.3 Metatheory of LDC

The PTS version of LDC satisfies the PTS analogues of all the lemmas and theorems satisfied by the simply-typed version, presented in Sections 3.2 and 3.4. The PTS version also enjoys the same standard operational semantics as the simply-typed version. Further, the PTS version is type-sound with respect to this semantics.

Next, we state the PTS analogues of some of the crucial lemmas and theorems presented in Sections 3.2 and 3.4.

Lemma 14 (Weakening). *If* $\Gamma_1, \Gamma_2 \vdash a :^q A$ *and* $\Delta_1 \vdash_0 C : s$ *and* $\lfloor \Gamma_1 \rfloor = \Delta_1$, *then* $\Gamma_1, z :^0 C, \Gamma_2 \vdash a :^q A$.

Lemma 15 (Substitution). *If* $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$ *and* $\Gamma \vdash c :^{r_0} C$ *and* $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$, *then* $\Gamma_1 + \Gamma, \Gamma_2\{c/z\} \vdash a\{c/z\} :^q A\{c/z\}$.

Theorem 6 (Preservation). *If* $\Gamma \vdash a :^q A$ *and* $\vdash a \rightsquigarrow a'$, *then* $\Gamma \vdash a' :^q A$.

Theorem 7 (Progress). *If* $\emptyset \vdash a :^q A$, *then either* a *is a value or there exists* a' *such that* $\vdash a \rightsquigarrow a'$.

Next, we consider heap semantics for LDC.

5.4 Heap Semantics for LDC

The PTS version of LDC enjoys the same heap reduction relation as its simply-typed counterpart. However, the PTS version presents a challenge with regard to well-typedness of terms during reduction. In the simply-typed version, we could delay substitutions in a term by loading them into the heap without being concerned about how it might affect the type of that term. In the PTS version, delayed substitutions may cause the term to 'lag behind' the type. We consider an example from Choudhury et. al. [17] that illustrates this point.

The polymorphic identity function, $\lambda^0 x : s.\lambda^1 y : x.y$, has type $\Pi x :^0 s.\Pi y :^1 x.x$. Instantiating the function at **Unit**, we get $(\lambda^0 x : s.\lambda^1 y : x.y)\,\mathbf{Unit}^0$ of type $\Pi y :^1 \mathbf{Unit}.\mathbf{Unit}$. Now, $[\emptyset](\lambda^0 x : s.\lambda^1 y : x.y)\,\mathbf{Unit}^0 \Longrightarrow_S^1 [x \overset{0}{\mapsto} \mathbf{Unit}]\lambda^1 y : x.y$. Unless we look at the definition in the heap, we have no reason to believe that $\lambda^1 y : x.y$ has type $\Pi y :^1 \mathbf{Unit}.\mathbf{Unit}$. The delayed substitution $x \overset{0}{\mapsto} \mathbf{Unit}$ causes the term $\lambda^1 y : x.y$ to lag behind the type $\Pi y :^1 \mathbf{Unit}.\mathbf{Unit}$. Note that this challenge arises due to delayed substitution only and is independent of usage and flow analyses.

To overcome this challenge, Choudhury et. al. [17] use the following strategy. First, they extend their type system with contexts that allow definitions. Then, they show that the extended calculus is sound with respect to heap semantics. Thereafter, they prove the original calculus equivalent to the extended calculus. Using this equivalence, they conclude that the original calculus is sound with respect to heap semantics. We use the same strategy for LDC. Owing to space constraints, we omit the details in the main body of the paper. The interested reader may please refer to Appendix F.

LDC is sound with respect to heap semantics. Note the statement below is the same as its simply-typed counterpart.

Theorem 8 (Soundness). *If* $H \models \Gamma$ *and* $\Gamma \vdash a :^q A$ *and* $q \neq 0$, *then either* a *is a value or there exists* H', Γ', a' *such that* $[H]a \Longrightarrow_S^q [H']a'$ *and* $H' \models \Gamma'$ *and* $\Gamma' \vdash a' :^q A$.

Below, we present some corollaries of this theorem.

Theorem 9. *In LDC($\mathcal{Q}_{\mathbb{N}}$): If $\emptyset \vdash f :^1 \Pi x :^0 s.x$ and $\emptyset \vdash_0 A : s$, then $f\ A^0$ must diverge.*

Theorem 10. *In LDC($\mathcal{Q}_{\mathbb{N}}$): In a strongly normalizing PTS, if $\emptyset \vdash f :^1 \Pi x :^0 s.\Pi y :^1 x.x$ and $\emptyset \vdash_0 A : s$ and $\emptyset \vdash a :^1 A$, then $f\ A^0\ a^1 =_\beta a$.*

6 Adding Unrestricted Use

Till now, we used LDC for analysing exact use, bounded use and dependency. In this section, we use the calculus for analysing unrestricted use as well. Unrestricted use, referred to by ω, is different from exact and bounded use, referred to by $n \in \mathbb{N}$, in two ways:

- ω is an additive annihilator, meaning $\omega + q = q + \omega = \omega$, for $q \in \mathbb{N} \cup \{\omega\}$. No $n \in \mathbb{N}$ is an additive annihilator.
- ω is a multiplicative annihilator (almost) as well, meaning $\omega \cdot q = q \cdot \omega = \omega$ for $q \in (\mathbb{N} - \{0\}) \cup \{\omega\}$. No $n \in \mathbb{N} - \{0\}$ is a multiplicative annihilator.

To accommodate this behaviour of ω, we need to make a change to our type system. But before we make this change, let us fix our preordered semirings:

- $\mathbb{N}^\omega_=$, that contains the preordered semiring $\mathbb{N}_=$ and ω with $\omega <: q$ for all q
- $\mathbb{N}^\omega_\geq$, that contains the preordered semiring $\mathbb{N}_\geq$ and ω with $\omega <: q$ for all q
- $\mathcal{Q}_{\text{Lin}}$ and $\mathcal{Q}_{\text{Aff}}$, described in Section 2.4

We use $\mathcal{Q}^\omega_{\mathbb{N}}$ to denote an arbitrary member of the above set of semirings. Next, we discuss the change necessary as we move from LDC($\mathcal{Q}_{\mathbb{N}}$) to LDC($\mathcal{Q}^\omega_{\mathbb{N}}$).

6.1 A Problem and its Solution

When unrestricted use is allowed, the type systems in Figures 2 and 6 cannot enforce fairness of resource usage. Consider the following 'unfair' derivation allowed by the simple type system:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x :^\omega A \vdash x :^\omega A}\ \text{ST-Var} \quad \dfrac{}{x :^\omega A \vdash x :^\omega A}\ \text{ST-Var}}{x :^\omega A \vdash (x^1, x) :^\omega {}^1A \times A}\ \text{ST-WPair}}{\emptyset \vdash \lambda^1 x : A.(x^1, x) :^\omega {}^1A \to {}^1A \times A}\ \text{ST-Lam}}{\emptyset \vdash \lambda^1 x : A.(x^1, x) :^1 {}^1A \to {}^1A \times A}\ \text{ST-SubR}$$

The judgement $\emptyset \vdash \lambda^1 x : A.(x^1, x) :^1 {}^1A \to {}^1A \times A$ is unfair because it allows copying of resources. Carefully observing the derivation, we find that the unfairness arises when ω 'tricks' the ST-Lam rule into believing that the term uses x (at most) once.

This unfairness leads to a failure in type soundness. To see how, consider the term: $y :^1 A \vdash (\lambda^1 x : A.(x^1, x))\ y^1 :^1 {}^1A \times A$ that type-checks via the above

derivation and rule ST-App. This term steps to: $\vdash (\lambda^1 x : A.(x^1, x))\, y^1 \rightsquigarrow (y^1, y)$. But then, we have unsoundness because: $y :^1 A \not\vdash (y^1, y) :^1 {}^1A \times A$. Therefore, to ensure type soundness, we need to modify rule ST-Lam and rule PTS-Lam.

We modify these rules as follows:

$$\textsc{ST-LamOmega}\quad \frac{\Gamma, x :^{q \cdot r} A \vdash b :^q B \qquad q = \omega \Rightarrow r = \omega \qquad q_0 \neq 0}{q_0 \cdot \Gamma \vdash \lambda^r x : A.b :^{q_0 \cdot q} {}^rA \to B}$$

$$\textsc{PTS-LamOmega}\quad \frac{\Gamma, x :^{q \cdot r} A \vdash b :^q B \qquad \Delta \vdash_0 \Pi x :^r A.B : s \qquad \lfloor \Gamma \rfloor = \Delta \qquad q = \omega \Rightarrow r = \omega \qquad q_0 \neq 0}{q_0 \cdot \Gamma \vdash \lambda^r x : A.b :^{q_0 \cdot q} \Pi x :^r A.B}$$

There are several points to note here:

- Rules ST-LamOmega and PTS-LamOmega are generalizations of rules ST-Lam and PTS-Lam respectively.
- These rules impose the constraint: $q = \omega \Rightarrow r = \omega$. This way ω won't be able to 'trick' the Lambda-rule into believing that functions use their arguments less than what they actually do. In particular, with these modified rules, the above unfair derivation won't go through.
- The constraint $q = \omega \Rightarrow r = \omega$, while required for blocking unfair derivations, also blocks some fair ones like the one below:

  $$\frac{x :^\omega A \vdash x :^\omega A}{\emptyset \vdash \lambda^1 x : A.x :^\omega {}^1A \to A}$$

  To allow such derivations, while still imposing this constraint, rules ST-LamOmega and PTS-LamOmega multiply the conclusion judgement by q_0. This multiplication helps these rules allow the above derivation as:

  $$\frac{x :^1 A \vdash x :^1 A}{\emptyset \vdash \lambda^1 x : A.x :^\omega {}^1A \to A}$$

- The side condition $q_0 \neq 0$ makes sure that a meaningful judgement is not turned into a meaningless one. Recall that judgements in 0-world are meaningless, as far as usage and dependency analyses are concerned.
- The rules ST-LamOmega and PTS-LamOmega may seem specific to usage analysis because the constraint $q = \omega \Rightarrow r = \omega$ does not have an analogue in dependency analysis. This, however, is a minor issue because we can rephrase this constraint as: $(q + q = q) \wedge (q \cdot r = q) \Rightarrow r <: q$.
  Wrt usage analysis, the above two constraints are equivalent.
  Wrt dependency analysis, the latter constraint is a tautology because $q \sqcup r = q \Rightarrow r \sqsubseteq q$. So rules ST-LamOmega and PTS-LamOmega make sense from the perspective of dependency analysis as well. Further, when viewed as rules analysing dependency, rules ST-LamOmega and PTS-LamOmega are equivalent to rules ST-Lam and PTS-Lam respectively.

– Rule ST-LamOmega can also be equivalently replaced with the following two simpler rules (and similarly for rule PTS-LamOmega):

$$\frac{\Gamma, x :^{q \cdot r} A \vdash b :^{q} B \quad q = \omega \Rightarrow r = \omega}{\Gamma \vdash \lambda^{r} x : A.b :^{q\ r} A \to B}\ \text{ST-LamOmega0} \qquad \frac{\omega \cdot \Gamma \vdash a :^{q} A \quad q \neq 0}{\omega \cdot \Gamma \vdash a :^{\omega} A}\ \text{ST-Omega}$$

The above modification is the only change that we need to make to the type systems presented in Figures 2 and 6 in order to track unrestricted use.

With this modification, LDC($\mathcal{Q}^{\omega}_{\mathbb{N}}$) satisfies all the lemmas and theorems satisfied by LDC($\mathcal{Q}_{\mathbb{N}}$). In particular, LDC($\mathcal{Q}^{\omega}_{\mathbb{N}}$) satisfies type soundness (Theorems 6 and 7) and heap soundness (Theorem 8). We state this property as a theorem below.

Theorem 11. *LDC($\mathcal{Q}^{\omega}_{\mathbb{N}}$) satisfies type soundness and heap soundness.*

Thus, LDC is a general linearity and dependency calculus. For tracking linearity in LDC, we can use any of the $\mathcal{Q}^{\omega}_{\mathbb{N}}$s. For tracking dependency in LDC, we can use any lattice. For tracking linearity and dependency simultaneously in LDC, we can use the cartesian product of these structures.

7 LDC vs. Standard Linear and Dependency Calculi

In section 3.5, we discussed how Sealing Calculus, a standard dependency calculus, embeds into LDC($\mathcal{L}$). Now, we compare LDC with a standard linear calculus, the Linear Nonlinear (LNL) λ-calculus of Benton [8]. The LNL λ-calculus tracks just linear and unrestricted use and is simply-typed. So we compare it with simply-typed LDC($\mathcal{Q}_{\text{Lin}}$).

The LNL calculus employs two types of contexts and two types of typing judgements. The two types of contexts, linear and nonlinear, correspond to assumptions at grades 1 and ω respectively in LDC($\mathcal{Q}_{\text{Lin}}$). The two types of judgements, linear and nonlinear, correspond to derivations in worlds 1 and ω respectively in LDC($\mathcal{Q}_{\text{Lin}}$). Note that in LNL calculus, linear and nonlinear contexts are denoted by Γ and Θ respectively; linear and nonlinear judgements are written as $\Theta; \Gamma \vdash_{\mathcal{L}} e : A$ and $\Theta \vdash_{\mathcal{C}} t : X$ respectively. The calculus contains standard intuitionistic types and linear types; it also contains two type constructors, F and G, via which the linear and the nonlinear worlds interact. The calculus uses A, B for linear types; X, Y for nonlinear types; a, b for linear variables; x, y for nonlinear variables; e, f for linear terms; s, t for nonlinear terms.

We present the translation function from LNL λ-calculus [8] to LDC($\mathcal{Q}_{\text{Lin}}$) in Figure 8. This translation preserves typing and meaning.

Theorem 12. *If $\Theta; \Gamma \vdash_{\mathcal{L}} e : A$, then $\overline{\Theta}^{\omega}, \overline{\Gamma}^{1} \vdash \overline{e} :^{1} \overline{A}$.*
If $\Theta \vdash_{\mathcal{C}} t : X$, then $\overline{\Theta}^{\omega} \vdash \overline{t} :^{\omega} \overline{X}$.
If $e =_{\beta} f$, then $\overline{e} =_{\beta} \overline{f}$. If $s =_{\beta} t$ then $\overline{s} =_{\beta} \overline{t}$.

$$\overline{\mathbf{1}} = \mathbf{Unit} \qquad \overline{X \times Y} = \overline{X} \,\&\, \overline{Y} \qquad \overline{I} = \mathbf{Unit} \qquad \overline{A \otimes B} = {}^{1}\overline{A} \times \overline{B}$$
$$\overline{X \to Y} = {}^{\omega}\overline{X} \to \overline{Y} \qquad \overline{G\ A} = \overline{A} \qquad \overline{A \multimap B} = {}^{1}\overline{A} \to \overline{B} \qquad \overline{F\ X} = {}^{\omega}\overline{X} \times \mathbf{Unit}$$

$$\overline{x} = x \qquad \overline{a} = a \qquad \overline{()} = \mathbf{unit} \qquad \overline{*} = \mathbf{unit}$$
$$\overline{(s,t)} = (\overline{s};\overline{t}) \qquad \overline{e \otimes f} = (\overline{e}^{1}, \overline{f})$$
$$\overline{\mathbf{fst}(s)} = \mathbf{proj}_1\ \overline{s} \qquad \overline{\mathbf{let}\ a \otimes b = e\ \mathbf{in}\ f} = \mathbf{let}_1\ (a^1, b) = \overline{e}\ \mathbf{in}\ \overline{f}$$
$$\overline{\mathbf{snd}(s)} = \mathbf{proj}_2\ \overline{s} \qquad \overline{\mathbf{let} * = e\ \mathbf{in}\ f} = \mathbf{let}_1\ \mathbf{unit} = \overline{e}\ \mathbf{in}\ \overline{f}$$
$$\overline{\lambda x : X.s} = \lambda^{\omega} x : \overline{X}.\overline{s} \qquad \overline{\lambda a : A.e} = \lambda^{1} a : \overline{A}.\overline{e} \qquad \overline{s\ t} = \overline{s}\ \overline{t}^{\omega} \qquad \overline{e\ f} = \overline{e}\ \overline{f}^{1}$$
$$\overline{G\ e} = \overline{e} \qquad \overline{\mathbf{derelict}\ s} = \overline{s}$$
$$\overline{F\ s} = (\overline{s}^{\omega}, \mathbf{unit}) \qquad \overline{\mathbf{let}\ Fx = e\ \mathbf{in}\ f} = \mathbf{let}_1\ (x^{\omega}, y) = \overline{e}\ \mathbf{in}\ \overline{f} \quad (y \text{ fresh})$$

Fig. 8. Type and term translation from LNL λ-calculus to LDC($\mathcal{Q}_{\text{Lin}}$)

Here, $\overline{\Gamma}^{1}$ and $\overline{\Theta}^{\omega}$ denote Γ and Θ, with the types translated, and assumptions held at 1 and ω respectively. Further, $\_ =_{\beta} \_$ denotes the beta equivalence relation on the terms of LNL calculus [7].

Note that a translation in the other direction from LDC($\mathcal{Q}_{\text{Lin}}$) to LNL calculus would fail because the latter does not model 0-use. Next, we compare LDC with GRAD [17] and DDC$^{\top}$ [16], standard dependent calculi for tracking usage and flow respectively. GRAD is a general coeffect calculus parametrized by an arbitrary partially-ordered semiring. LDC($\mathcal{Q}^{\omega}_{\mathbb{N}}$), on the other hand, is a linearity calculus parametrized by specific preordered semirings, i.e. $\mathcal{Q}^{\omega}_{\mathbb{N}}$s. When compared over these semirings, we can show that LDC subsumes GRAD. We can also show that over arbitrary lattices, LDC subsumes DDC$^{\top}$.

Theorem 13. *With $\mathcal{Q}^{\omega}_{\mathbb{N}}$ as the parametrizing structure, if $\Gamma \vdash a : A$ in* GRAD*, then $\Gamma \vdash a :^{1} A$ in LDC. Further, if $\vdash a \rightsquigarrow a'$ in* GRAD*, then $\vdash a \rightsquigarrow a'$ in LDC.*

Theorem 14. *With $\mathcal{L}$ as the parametrizing structure, if $\Gamma \vdash a :^{\ell} A$ in DDC$^{\top}$, then $\Gamma \vdash a :^{\ell} A$ in LDC. Further, if $\vdash a \rightsquigarrow a'$ in DDC$^{\top}$, then $\vdash a \rightsquigarrow a'$ in LDC.*

8 Conclusion

We have shown that linearity and dependency analyses can be systematically unified and combined into a single calculus. We presented, LDC, a general linearity and dependency calculus parametrized by an arbitrary PTS. We showed that linearity and dependency analyses in LDC are correct using a heap semantics. We also showed that LDC subsumes standard calculi for linearity and dependency analyses. In this paper, we focused on the syntactic properties of LDC. In a future work, we plan to explore the semantic properties of the calculus. In particular, we want to find out how semantic models of LDC compare with the categorical models of linear and dependency type systems.

References

1. Abadi, M., Banerjee, A., Heintze, N., Riecke, J.G.: A core calculus of dependency. In: Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. p. 147–160. POPL '99, Association for Computing Machinery, New York, NY, USA (1999). https://doi.org/10.1145/292540.292555, https://doi.org/10.1145/292540.292555
2. Abel, A., Scherer, G.: On irrelevance and algorithmic equality in predicative type theory. Logical Methods in Computer Science **8**(1) (mar 2012). https://doi.org/10.2168/lmcs-8(1:29)2012, https://doi.org/10.2168%2Flmcs-8%281%3A29%292012
3. Abramsky, S.: Computational interpretations of linear logic. Theoretical Computer Science **111**(1), 3–57 (1993). https://doi.org/https://doi.org/10.1016/0304-3975(93)90181-R, https://www.sciencedirect.com/science/article/pii/030439759390181R
4. Atkey, R.: Syntax and semantics of quantitative type theory. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science. p. 56–65. LICS '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3209108.3209189, https://doi.org/10.1145/3209108.3209189
5. Barber, A.: Dual intuitionistic linear logic. Tech. rep., Edinburgh, Scotland (1996)
6. Barendregt, H.P.: Lambda Calculi with Types, p. 117–309. Oxford University Press, Inc., USA (1993)
7. Benton, N., Wadler, P.: Linear logic, monads and the lambda calculus. In: Proceedings 11th Annual IEEE Symposium on Logic in Computer Science. pp. 420–431 (1996). https://doi.org/10.1109/LICS.1996.561458
8. Benton, P.N.: A mixed linear and non-linear logic: Proofs, terms and models (extended abstract). In: Selected Papers from the 8th International Workshop on Computer Science Logic. p. 121–135. CSL '94, Springer-Verlag, Berlin, Heidelberg (1994)
9. Benton, P.N., Bierman, G.M., Paiva, V.d., Hyland, M.: A term calculus for intuitionistic linear logic. In: Proceedings of the International Conference on Typed Lambda Calculi and Applications. p. 75–90. TLCA '93, Springer-Verlag, Berlin, Heidelberg (1993)
10. Bernardy, J.P., Boespflug, M., Newton, R.R., Peyton Jones, S., Spiwack, A.: Linear haskell: practical linearity in a higher-order polymorphic language. In: Principles of Programming Languages 2018 (POPL 2018) (January 2018)
11. Birkhoff, G.: Lattice Theory. American Mathematical Society, Providence, 3rd edn. (1967)
12. Brady, E.: Idris 2: Quantitative Type Theory in Practice. In: Møller, A., Sridharan, M. (eds.) 35th European Conference on Object-Oriented Programming (ECOOP 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 194, pp. 9:1–9:26. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). https://doi.org/10.4230/LIPIcs.ECOOP.2021.9, https://drops.dagstuhl.de/opus/volltexte/2021/14052
13. Brunel, A., Gaboardi, M., Mazza, D., Zdancewic, S.: A core quantitative coeffect calculus. In: Proceedings of the 23rd European Symposium on Programming Languages and Systems - Volume 8410. p. 351–370. Springer-Verlag, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54833-8_19, https://doi.org/10.1007/978-3-642-54833-8_19

14. Caires, L., Pfenning, F., Toninho, B.: Linear logic propositions as session types. Mathematical Structures in Computer Science **26**(3), 367–423 (2016). https://doi.org/10.1017/S0960129514000218
15. Calcagno, C., Taha, W., Huang, L., Leroy, X.: Implementing multi-stage languages using asts, gensym, and reflection. In: Pfenning, F., Smaragdakis, Y. (eds.) Generative Programming and Component Engineering. pp. 57–76. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
16. Choudhury, P., Eades, H., Weirich, S.: A dependent dependency calculus. In: Sergey, I. (ed.) Programming Languages and Systems. pp. 403–430. Springer International Publishing, Cham (2022)
17. Choudhury, P., Eades III, H., Eisenberg, R.A., Weirich, S.: A graded dependent type system with a usage-aware semantics. Proc. ACM Program. Lang. **5**(POPL) (Jan 2021). https://doi.org/10.1145/3434331, https://doi.org/10.1145/3434331
18. Davies, R.: A temporal logic approach to binding-time analysis. J. ACM **64**(1) (mar 2017). https://doi.org/10.1145/3011069, https://doi.org/10.1145/3011069
19. Denning, D.E.: A lattice model of secure information flow. Commun. ACM **19**(5), 236–243 (May 1976). https://doi.org/10.1145/360051.360056, https://doi.org/10.1145/360051.360056
20. Gaboardi, M., Katsumata, S.y., Orchard, D., Breuvart, F., Uustalu, T.: Combining effects and coeffects via grading. SIGPLAN Not. **51**(9), 476–489 (sep 2016). https://doi.org/10.1145/3022670.2951939, https://doi.org/10.1145/3022670.2951939
21. Ghica, D.R., Smith, A.I.: Bounded linear types in a resource semiring. In: Shao, Z. (ed.) Programming Languages and Systems. pp. 331–350. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
22. Girard, J.Y.: Linear logic. Theoretical Computer Science **50**(1), 1–101 (1987). https://doi.org/https://doi.org/10.1016/0304-3975(87)90045-4, https://www.sciencedirect.com/science/article/pii/0304397587900454
23. Girard, J.Y., Scedrov, A., Scott, P.J.: Bounded linear logic: a modular approach to polynomial-time computability. Theoretical Computer Science **97**(1), 1–66 (1992). https://doi.org/https://doi.org/10.1016/0304-3975(92)90386-T, https://www.sciencedirect.com/science/article/pii/030439759290386T
24. Goguen, J.A., Meseguer, J.: Security policies and security models. In: 1982 IEEE Symposium on Security and Privacy. pp. 11–11 (1982)
25. Gomard, C.K., Jones, N.D.: A partial evaluator for the untyped lambda-calculus. Journal of Functional Programming **1**(1), 21–69 (1991). https://doi.org/10.1017/S0956796800000058
26. Heintze, N., Riecke, J.G.: The slam calculus: Programming with secrecy and integrity. In: Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. p. 365–377. POPL '98, Association for Computing Machinery, New York, NY, USA (1998). https://doi.org/10.1145/268946.268976, https://doi.org/10.1145/268946.268976
27. Katsumata, S.y.: A double category theoretic analysis of graded linear exponential comonads. In: Baier, C., Dal Lago, U. (eds.) Foundations of Software Science and Computation Structures. pp. 110–127. Springer International Publishing, Cham (2018)
28. Marshall, D., Vollmer, M., Orchard, D.: Linearity and uniqueness: An entente cordiale. In: Sergey, I. (ed.) Programming Languages and Systems. pp. 346–375. Springer International Publishing, Cham (2022)

29. McBride, C.: I Got Plenty o' Nuttin', pp. 207–233. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-30936-1_12, https://doi.org/10.1007/978-3-319-30936-1_12
30. Moggi, E.: Notions of computation and monads. Information and Computation **93**(1), 55–92 (1991), https://www.sciencedirect.com/science/article/pii/0890540191900524, selections from 1989 IEEE Symposium on Logic in Computer Science
31. Moon, B., Eades III, H., Orchard, D.: Graded modal dependent type theory. In: Yoshida, N. (ed.) Programming Languages and Systems. pp. 462–490. Springer International Publishing, Cham (2021)
32. Myers, A.C.: Jflow: Practical mostly-static information flow control. In: Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. p. 228–241. POPL '99, Association for Computing Machinery, New York, NY, USA (1999). https://doi.org/10.1145/292540.292561, https://doi.org/10.1145/292540.292561
33. Orchard, D., Liepelt, V.B., Eades III, H.: Quantitative program reasoning with graded modal types. Proc. ACM Program. Lang. **3**(ICFP) (Jul 2019). https://doi.org/10.1145/3341714, https://doi.org/10.1145/3341714
34. Petricek, T., Orchard, D., Mycroft, A.: Coeffects: A calculus of context-dependent computation. In: Proceedings of International Conference on Functional Programming. ICFP 2014 (2014)
35. Pitts, A.M.: Nominal Sets: Names and Symmetry in Computer Science. Cambridge University Press, USA (2013)
36. Shikuma, N., Igarashi, A.: Proving noninterference by a fully complete translation to the simply typed λ-calculus. In: Proceedings of the 11th Asian Computing Science Conference on Advances in Computer Science: Secure Software and Related Issues. p. 301–315. ASIAN'06, Springer-Verlag, Berlin, Heidelberg (2006)
37. STEFAN, D., MAZIÈRES, D., MITCHELL, J.C., RUSSO, A.: Flexible dynamic information flow control in the presence of exceptions. Journal of Functional Programming **27**, e5 (2017). https://doi.org/10.1017/S0956796816000241
38. Turner, D.N., Wadler, P.: Operational interpretations of linear logic. Theor. Comput. Sci. **227**(1–2), 231–248 (sep 1999). https://doi.org/10.1016/S0304-3975(99)00054-7, https://doi.org/10.1016/S0304-3975(99)00054-7
39. Volpano, D., Irvine, C., Smith, G.: A sound type system for secure flow analysis. J. Comput. Secur. **4**(2–3), 167–187 (jan 1996)
40. Wadler, P.: Linear types can change the world! In: PROGRAMMING CONCEPTS AND METHODS. North (1990)

A Join Not Derivable in Graded Type Systems

Proposition 1. *Graded type systems [21,13,34,4,33,2,17] cannot derive a monadic join.*

Proof. The mentioned type systems vary slightly in their design. However, all of them contain a core graded calculus. Below, we first present this Core Graded Calculus, GCore, and thereafter show that GCore cannot derive a monadic join.

GCore is parametrized by an arbitrary preordered semiring $\mathcal{Q} = (Q, +, \cdot, 0, 1 <:)$. The parametrized calculus is referred to as GCore($\mathcal{Q}$). The grammar and the typing rules of GCore($\mathcal{Q}$) appear in Figures 9 and 10 respectively. The operations on contexts are defined as in Section 3.1.

$$
\begin{aligned}
\text{grades}, q \in Q &::= 0 \mid 1 \mid q_1 + q_2 \mid q_1 \cdot q_2 \mid \ldots \\
\text{types}, A, B &::= \mathbf{Bool} \mid A \multimap B \mid !_q\, A \\
\text{terms}, a, b &::= x \mid \lambda x : A.b \mid b\ a \mid !_q\ a \mid \mathbf{let}\ !_q x = a\ \mathbf{in}\ b \\
&\quad \mid \mathbf{true} \mid \mathbf{false} \mid \mathbf{if}\ b\ \mathbf{then}\ a_1\ \mathbf{else}\ a_2 \\
\text{contexts}, \Gamma &::= \emptyset \mid \Gamma, x :^q A
\end{aligned}
$$

Fig. 9. Grammar of GCore($\mathcal{Q}$)

Now, towards contradiction, assume that GCore($\mathcal{Q}$) can derive a monadic join. Then, for any A, there exists a closed non-constant function of type $!_{q_1}!_{q_2} A \to !_{q_1 \cdot q_2} A$ for all $q_1, q_2 \in \mathcal{Q}$. By model-theoretic arguments, we shall show that for some $\mathcal{Q}$, A, q_1 and q_2, any such function must be constant.

Fix $\mathcal{Q}$ to be the following semiring. The underlying set is $\{0, 1, k_1, k_2\}$ and the semiring operations and relation are defined as follows:

$$
\begin{aligned}
q_1 + q_2 &= k_2 \text{ if } q_1, q_2 \notin \{0\} \\
q_1 \cdot q_2 &= k_2 \text{ if } q_1, q_2 \notin \{0, 1\} \\
q_1 <: q_2 &\triangleq q_1 = q_2
\end{aligned}
$$

Note here that $k_1 \cdot k_1 = k_2$. Below, we shall show that there exists no non-constant function of type $!_{k_1}\, !_{k_1}\, \mathbf{Bool} \to !_{k_2}\, \mathbf{Bool}$ in GCore($\mathcal{Q}$).

Towards contradiction, suppose such a function, $\emptyset \vdash f :!_{k_1}\, !_{k_1}\, \mathbf{Bool} \to !_{k_2}\, \mathbf{Bool}$ exists. Then, $f\,(!_{k_1}\, !_{k_1}\, \mathbf{true})$ and $f\,(!_{k_1}\, !_{k_1}\, \mathbf{false})$ reduce to different values (Assume a call-by-value reduction). Without loss of generality, say $f\,(!_{k_1}\, !_{k_1}\, \mathbf{true}) \rightsquigarrow^* !_{k_2}\, \mathbf{true}$ and $f\ (!_{k_1}\ !_{k_1}\ \mathbf{false}) \rightsquigarrow^*\ !_{k_2}\ \mathbf{false}$ ($\rightsquigarrow^*$ is the multistep call-by-value reduction relation). Now, for any sound interpretation, $[\![\_]\!]_{\mathcal{M}}$, of GCore($\mathcal{Q}$) in a model $\mathcal{M}$, we should have: $[\![f\ (!_{k_1}\ !_{k_1}\ \mathbf{true})]\!]_{\mathcal{M}} = [\![!_{k_2}\, \mathbf{true}]\!]_{\mathcal{M}}$ and $[\![f\ (!_{k_1}\ !_{k_1}\ \mathbf{false})]\!]_{\mathcal{M}} = [\![!_{k_2}\, \mathbf{false}]\!]_{\mathcal{M}}$. We construct a sound model of GCore($\mathcal{Q}$) where these equalities lead to a contradiction.

$\boxed{\Gamma \vdash a : A}$ *(Typing Rules)*

$$\textsc{GCore-Var}\ \frac{}{0 \cdot \Gamma_1, x :^1 A, 0 \cdot \Gamma_2 \vdash x : A} \qquad \textsc{GCore-Lam}\ \frac{\Gamma, x :^1 A \vdash b : B}{\Gamma \vdash \lambda x : A.b : A \multimap B} \qquad \textsc{GCore-App}\ \frac{\Gamma_1 \vdash b : A \multimap B \quad \Gamma_2 \vdash a : A \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 + \Gamma_2 \vdash b\ a : B}$$

$$\textsc{GCore-Bang}\ \frac{\Gamma \vdash a : A}{q \cdot \Gamma \vdash !_q\ a :!_q A} \qquad \textsc{GCore-LetBang}\ \frac{\Gamma_1 \vdash a :!_q A \quad \Gamma_2, x :^q A \vdash b : B \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 + \Gamma_2 \vdash \mathbf{let}\ !_q x = a\ \mathbf{in}\ b : B} \qquad \textsc{GCore-True}\ \frac{}{0 \cdot \Gamma \vdash \mathbf{true} : \mathbf{Bool}}$$

$$\textsc{GCore-False}\ \frac{}{0 \cdot \Gamma \vdash \mathbf{false} : \mathbf{Bool}} \qquad \textsc{GCore-If}\ \frac{\Gamma_1 \vdash b : \mathbf{Bool} \quad \Gamma_2 \vdash a_1 : A \quad \Gamma_2 \vdash a_2 : A \quad \lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor}{\Gamma_1 + \Gamma_2 \vdash \mathbf{if}\ b\ \mathbf{then}\ a_1\ \mathbf{else}\ a_2 : A}$$

$$\textsc{GCore-Sub}\ \frac{\Gamma_1 \vdash a : A \qquad \Gamma_2 <: \Gamma_1}{\Gamma_2 \vdash a : A}$$

Fig. 10. Typing rules for GCore($\mathcal{Q}$)

Let **Set** be the category of sets and functions. Note that **Set** is a monoidal category with the monoidal product given by cartesian product. Now, any $\mathcal{Q}$-graded linear exponential comonad on **Set** provides a sound interpretation of GCore($\mathcal{Q}$) [27]. We define !, a $\mathcal{Q}$-graded linear exponential comonad on **Set**, as follows:

$$!(0) = !(k_1) = *$$
$$!(1) = !(k_2) = \mathbf{Id}$$

Here, **Id** is the identity functor and $*$ is the functor that maps every object to the terminal object. The morphisms associated with ! are as expected.

Now, interpreting using !, we have:

$$[\![f\ (!_{k_1}\ !_{k_1}\ \mathbf{true})]\!]_{(\mathbf{Set},!)} = [\![f\ (!_{k_1}\ !_{k_1}\ \mathbf{false})]\!]_{(\mathbf{Set},!)} = \mathrm{app} \circ \langle [\![f]\!]_{(\mathbf{Set},!)}, \langle\rangle \rangle$$

But $[\![!_{k_2}\ \mathbf{true}]\!]_{(\mathbf{Set},!)} \neq [\![!_{k_2}\ \mathbf{false}]\!]_{(\mathbf{Set},!)}$. A contradiction.

B Linearity Analysis in Simply-Typed LDC

Lemma 16 (Multiplication (Lemma 1)). *If $\Gamma \vdash a :^q A$, then $r_0 \cdot \Gamma \vdash a :^{r_0 \cdot q} A$.*

Proof. By induction on $\Gamma \vdash a :^q A$.

- Rule ST-Var. Have: $0 \cdot \Gamma_1, x :^q A, 0 \cdot \Gamma_2 \vdash x :^q A$.
  Need to show: $0 \cdot \Gamma_1, x :^{r_0 \cdot q} A, 0 \cdot \Gamma_2 \vdash x :^{r_0 \cdot q} A$.
  This case follows by rule ST-Var.
- Rule ST-Lam. Have: $\Gamma \vdash \lambda^r x : A.b :^q {}^rA \to B$ where $\Gamma, x :^{q \cdot r} A \vdash b :^q B$.
  Need to show: $r_0 \cdot \Gamma \vdash \lambda^r x : A.b :^{r_0 \cdot q} {}^rA \to B$.
  By IH, $r_0 \cdot \Gamma, x :^{r_0 \cdot (q \cdot r)} A \vdash b :^{r_0 \cdot q} B$.
  By associativity of multiplication, $r_0 \cdot (q \cdot r) = (r_0 \cdot q) \cdot r$.
  This case, then, follows by rule ST-Lam.
- Rule ST-App. Have: $\Gamma_1 + \Gamma_2 \vdash b\ a^r :^q B$ where $\Gamma_1 \vdash b :^q {}^rA \to B$ and $\Gamma_2 \vdash a :^{q \cdot r} A$.
  Need to show: $r_0 \cdot (\Gamma_1 + \Gamma_2) \vdash b\ a^r :^{r_0 \cdot q} B$.
  By IH, $r_0 \cdot \Gamma_1 \vdash b :^{r_0 \cdot q} {}^rA \to B$ and $r_0 \cdot \Gamma_2 \vdash a :^{r_0 \cdot (q \cdot r)} A$.
  This case, then, follows by rule ST-App, using associative and distributive properties.
- Rule ST-WPair. Have: $\Gamma_1 + \Gamma_2 \vdash ({a_1}^r, a_2) :^q {}^rA_1 \times A_2$ where $\Gamma_1 \vdash a_1 :^{q \cdot r} A_1$ and $\Gamma_2 \vdash a_2 :^q A_2$.
  Need to show: $r_0 \cdot (\Gamma_1 + \Gamma_2) \vdash ({a_1}^r, a_2) :^{r_0 \cdot q} {}^rA_1 \times A_2$.
  By IH, $r_0 \cdot \Gamma_1 \vdash a_1 :^{r_0 \cdot (q \cdot r)} A_1$ and $r_0 \cdot \Gamma_2 \vdash a_2 :^{r_0 \cdot q} A_2$.
  This case, then, follows by rule ST-WPair, using associative and distributive properties.
- Rule ST-LetPair. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0} {}^rA_1 \times A_2$ and $\Gamma_2, x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2 \vdash b :^q B$.
  Need to show: $r_0 \cdot (\Gamma_1 + \Gamma_2) \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^{r_0 \cdot q} B$.
  By IH, $r_0 \cdot \Gamma_1 \vdash a :^{r_0 \cdot (q \cdot q_0)} {}^rA_1 \times A_2$ and $r_0 \cdot \Gamma_2, x :^{r_0 \cdot (q \cdot q_0 \cdot r)} A_1, y :^{r_0 \cdot (q \cdot q_0)} A_2 \vdash b :^{r_0 \cdot q} B$.
  This case, then, follows by rule ST-LetPair, using associative and distributive properties.
- Rule ST-Unit. Have: $0 \cdot \Gamma \vdash \mathbf{unit} :^q \mathbf{Unit}$.
  Need to show: $r_0 \cdot (0 \cdot \Gamma) \vdash \mathbf{unit} :^{r_0 \cdot q} \mathbf{Unit}$.
  This case follows by rule ST-Unit, using associative and identity properties.
- Rule ST-LetUnit. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0} \mathbf{Unit}$ and $\Gamma_2 \vdash b :^q B$.
  Need to show: $r_0 \cdot (\Gamma_1 + \Gamma_2) \vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b :^{r_0 \cdot q} B$.
  By IH, $r_0 \cdot \Gamma_1 \vdash a :^{r_0 \cdot (q \cdot q_0)} \mathbf{Unit}$ and $r_0 \cdot \Gamma_2 \vdash b :^{r_0 \cdot q} B$.
  This case, then, follows by rule ST-LetUnit, using associative and distributive properties.
- Rule ST-SPair. Have: $\Gamma \vdash (a_1; a_2) :^q A_1 \,\&\, A_2$ where $\Gamma \vdash a_1 :^q A_1$ and $\Gamma \vdash a_2 :^q A_2$.
  Need to show: $r_0 \cdot \Gamma \vdash (a_1; a_2) :^{r_0 \cdot q} A_1 \,\&\, A_2$.
  By IH, $r_0 \cdot \Gamma \vdash a_1 :^{r_0 \cdot q} A_1$ and $r_0 \cdot \Gamma \vdash a_2 :^{r_0 \cdot q} A_2$.
  This case, then, follows by rule ST-SPair.
- Rule ST-Proj1. Have: $\Gamma \vdash \mathbf{proj}_1\ a :^q A_1$ where $\Gamma \vdash a :^q A_1 \,\&\, A_2$.
  Need to show: $r_0 \cdot \Gamma \vdash \mathbf{proj}_1\ a :^{r_0 \cdot q} A_1$.
  By IH, $r_0 \cdot \Gamma \vdash a :^{r_0 \cdot q} A_1 \,\&\, A_2$.
  This case, then, follows by rule ST-Proj1.

– Rule ST-Proj2. Similar to rule ST-Proj1.
– Rule ST-Inj1. Have $\Gamma \vdash \mathbf{inj}_1\ a_1 :^q A_1 + A_2$ where $\Gamma \vdash a_1 :^q A_1$.
Need to show: $r_0 \cdot \Gamma \vdash \mathbf{inj}_1\ a_1 :^{r_0 \cdot q} A_1 + A_2$.
By IH, $r_0 \cdot \Gamma \vdash a_1 :^{r_0 \cdot q} A_1$.
This case, then, follows by rule ST-Inj1.
– Rule ST-Inj2. Similar to rule ST-Inj1.
– Rule ST-Case. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0} A_1 + A_2$ and $\Gamma_2, x_1 :^{q \cdot q_0} A_1 \vdash b_1 :^q B$ and $\Gamma_2, x_2 :^{q \cdot q_0} A_2 \vdash b_2 :^q B$.
Need to show: $r_0 \cdot (\Gamma_1 + \Gamma_2) \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^{r_0 \cdot q} B$.
By IH, $r_0 \cdot \Gamma_1 \vdash a :^{r_0 \cdot (q \cdot q_0)} A_1 + A_2$ and $r_0 \cdot \Gamma_2, x_1 :^{r_0 \cdot (q \cdot q_0)} A_1 \vdash b_1 :^{r_0 \cdot q} B$ and $r_0 \cdot \Gamma_2, x_2 :^{r_0 \cdot (q \cdot q_0)} A_2 \vdash b_2 :^{r_0 \cdot q} B$.
This case, then, follows by rule ST-Case, using associative and distributive properties.
– Rule ST-SubL. Have: $\Gamma \vdash a :^q A$ where $\Gamma' \vdash a :^q A$ and $\Gamma <: \Gamma'$.
Need to show: $r_0 \cdot \Gamma \vdash a :^{r_0 \cdot q} A$.
By IH, $r_0 \cdot \Gamma' \vdash a :^{r_0 \cdot q} A$.
Next, $r_0 \cdot \Gamma <: r_0 \cdot \Gamma'$, since $\Gamma <: \Gamma'$.
This case, then, follows by rule ST-SubL.
– Rule ST-SubR. Have: $\Gamma \vdash a :^{q'} A$ where $\Gamma \vdash a :^q A$ and $q <: q'$.
Need to show: $r_0 \cdot \Gamma \vdash a :^{r_0 \cdot q'} A$.
By IH, $r_0 \cdot \Gamma \vdash a :^{r_0 \cdot q} A$.
Next, $r_0 \cdot q <: r_0 \cdot q'$, since $q <: q'$.
This case, then, follows by rule ST-SubR.

Lemma 17 (Factorization (Lemma 2)). *If $\Gamma \vdash a :^q A$ and $q \neq 0$, then there exists Γ' such that $\Gamma' \vdash a :^1 A$ and $\Gamma <: q \cdot \Gamma'$.*

Proof. By induction on $\Gamma \vdash a :^q A$.

– Rule ST-Var. Have: $0 \cdot \Gamma_1, x :^q A, 0 \cdot \Gamma_2 \vdash x :^q A$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash x :^1 A$ and $0 \cdot \Gamma_1, x :^q A, 0 \cdot \Gamma_2 <: q \cdot \Gamma'$.
This case follows by setting $\Gamma' := 0 \cdot \Gamma_1, x :^1 A, 0 \cdot \Gamma_2$.
– Rule ST-Lam. Have: $\Gamma \vdash \lambda^r x : A.b :^q {}^rA \to B$ where $\Gamma, x :^{q \cdot r} A \vdash b :^q B$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash \lambda^r x : A.b :^1 {}^rA \to B$ and $\Gamma <: q \cdot \Gamma'$.
By IH, $\exists \Gamma_1'$ and r_1' such that $\Gamma_1', x :^{r_1'} A \vdash b :^1 B$ and $\Gamma <: q \cdot \Gamma_1'$ and $q \cdot r <: q \cdot r_1'$.
Since $q \cdot r <: q \cdot r_1'$ and $q \neq 0$, therefore $r <: r_1'$.
By rule ST-SubL, $\Gamma_1', x :^r A \vdash b :^1 B$.
This case, then, follows by rule ST-Lam by setting $\Gamma' := \Gamma_1'$.
– Rule ST-App. Have: $\Gamma_1 + \Gamma_2 \vdash b\ a^r :^q B$ where $\Gamma_1 \vdash b :^q {}^rA \to B$ and $\Gamma_2 \vdash a :^{q \cdot r} A$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash b\ a^r :^1 B$ and $\Gamma_1 + \Gamma_2 <: q \cdot \Gamma'$.
There are two cases to consider.
  - $r \neq 0$. Since $q \neq 0$, therefore $q \cdot r \neq 0$.
  By IH, $\exists \Gamma_1', \Gamma_2'$ such that $\Gamma_1' \vdash b :^1 {}^rA \to B$ and $\Gamma_2' \vdash a :^1 A$ and $\Gamma_1 <: q \cdot \Gamma_1'$ and $\Gamma_2 <: (q \cdot r) \cdot \Gamma_2'$.

By Lemma 16, $r \cdot \Gamma_2' \vdash a :^r A$.
By rule ST-App, $\Gamma_1' + r \cdot \Gamma_2' \vdash b\ a^r :^1 B$.
This case, then, follows by setting $\Gamma' := \Gamma_1' + r \cdot \Gamma_2'$.

- $r = 0$. Then, $\Gamma_2 \vdash a :^0 A$. By Lemma 16, $0 \cdot \Gamma_2 \vdash a :^0 A$.
  By IH, $\exists \Gamma_1'$ such that $\Gamma_1' \vdash b :^1 {}^0A \to B$ and $\Gamma_1 <: q \cdot \Gamma_1'$.
  By rule ST-App, $\Gamma_1' \vdash b\ a^0 :^1 B$ and $\Gamma_1 + \Gamma_2 <: \Gamma_1 <: q \cdot \Gamma_1'$.
  This case, then, follows by setting $\Gamma' := \Gamma_1'$.

– Rule ST-WPair. Have: $\Gamma_1 + \Gamma_2 \vdash ({a_1}^r, a_2) :^q {}^rA_1 \times A_2$ where $\Gamma_1 \vdash a_1 :^{q \cdot r} A_1$ and $\Gamma_2 \vdash a_2 :^q A_2$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash ({a_1}^r, a_2) :^1 {}^rA_1 \times A_2$ and $\Gamma_1 + \Gamma_2 <: q \cdot \Gamma'$.
There are two cases to consider.

- $r \neq 0$. Since $q \neq 0$, therefore $q \cdot r \neq 0$.
  By IH, $\exists \Gamma_1', \Gamma_2'$ such that $\Gamma_1' \vdash a_1 :^1 A_1$ and $\Gamma_2' \vdash a_2 :^1 A_2$ and $\Gamma_1 <: (q \cdot r) \cdot \Gamma_1'$ and $\Gamma_2 <: q \cdot \Gamma_2'$.
  By Lemma 16, $r \cdot \Gamma_1' \vdash a_1 :^r A_1$.
  By rule ST-WPair, $r \cdot \Gamma_1' + \Gamma_2' \vdash ({a_1}^r, a_2) :^1 {}^rA_1 \times A_2$.
  This case, then, follows by setting $\Gamma' := r \cdot \Gamma_1' + \Gamma_2'$.
- $r = 0$. Then, $\Gamma_1 \vdash a_1 :^0 A_1$. By Lemma 16, $0 \cdot \Gamma_1 \vdash a_1 :^0 A_1$.
  By IH, $\exists \Gamma_2'$ such that $\Gamma_2' \vdash a_2 :^1 A_2$ and $\Gamma_2 <: q \cdot \Gamma_2'$.
  By rule ST-WPair, $\Gamma_2' \vdash ({a_1}^0, a_2) :^1 {}^0A_1 \times A_2$ and $\Gamma_1 + \Gamma_2 <: \Gamma_2 <: q \cdot \Gamma_2'$.
  This case, then, follows by setting $\Gamma' := \Gamma_2'$.

– Rule ST-LetPair. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0} {}^rA_1 \times A_2$ and $\Gamma_2, x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2 \vdash b :^q B$ and $q_0 <: 1$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^1 B$ and $\Gamma_1 + \Gamma_2 <: q \cdot \Gamma'$.
Since $q \neq 0$ and $q_0 <: 1$, therefore $q \cdot q_0 \neq 0$.
By IH, $\exists \Gamma_1', \Gamma_2'$ and r', q' such that $\Gamma_1' \vdash a :^1 {}^rA_1 \times A_2$ and $\Gamma_2', x :^{r'} A_1, y :^{q'} A_2 \vdash b :^1 B$ and $\Gamma_1 <: (q \cdot q_0) \cdot \Gamma_1'$ and $\Gamma_2 <: q \cdot \Gamma_2'$ and $q \cdot q_0 \cdot r <: q \cdot r'$ and $q \cdot q_0 <: q \cdot q'$.
Since $q \neq 0$, therefore $q_0 \cdot r <: r'$ and $q_0 <: q'$.
By rule ST-SubL, $\Gamma_2', x :^{q_0 \cdot r} A_1, y :^{q_0} A_2 \vdash b :^1 B$.
Again, by Lemma 16, $q_0 \cdot \Gamma_1' \vdash a :^{q_0} {}^rA_1 \times A_2$.
By rule ST-LetPair, $q_0 \cdot \Gamma_1' + \Gamma_2' \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^1 B$.
This case, then, follows by setting $\Gamma' := q_0 \cdot \Gamma_1' + \Gamma_2'$.

– Rule ST-Unit. Have: $0 \cdot \Gamma \vdash \mathbf{unit} :^q \mathbf{Unit}$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash \mathbf{unit} :^1 \mathbf{Unit}$ and $0 \cdot \Gamma <: q \cdot \Gamma'$.
This case follows by setting $\Gamma' := 0 \cdot \Gamma$.

– Rule ST-LetUnit. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0} \mathbf{Unit}$ and $\Gamma_2 \vdash b :^q B$ and $q_0 <: 1$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash \mathbf{let}_{q_0} \mathbf{unit} = a \mathbf{in} b :^1 B$ and $\Gamma_1 + \Gamma_2 <: q \cdot \Gamma'$.
By IH, $\exists \Gamma_1', \Gamma_2'$ such that $\Gamma_1' \vdash a :^1 \mathbf{Unit}$ and $\Gamma_2' \vdash b :^1 B$ and $\Gamma_1 <: (q \cdot q_0) \cdot \Gamma_1'$ and $\Gamma_2 <: q \cdot \Gamma_2'$.
By Lemma 16, $q_0 \cdot \Gamma_1' \vdash a :^{q_0} \mathbf{Unit}$.
By rule ST-LetUnit, $q_0 \cdot \Gamma_1' + \Gamma_2' \vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b :^1 B$.
This case follows by setting $\Gamma' := q_0 \cdot \Gamma_1' + \Gamma_2'$.

– Rule ST-SPair. Have: $\Gamma \vdash (a_1; a_2) :^q A_1 \,\&\, A_2$ where $\Gamma \vdash a_1 :^q A_1$ and $\Gamma \vdash a_2 :^q A_2$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash (a_1; a_2) :^1 A_1 \,\&\, A_2$ and $\Gamma <: q \cdot \Gamma'$.
By IH, $\exists \Gamma_1', \Gamma_2'$ such that $\Gamma_1' \vdash a_1 :^1 A_1$ and $\Gamma_2' \vdash a_2 :^1 A_2$ and $\Gamma <: q \cdot \Gamma_1'$ and $\Gamma <: q \cdot \Gamma_2'$.
Now, for quantities q_1, q_2 define an operator $q_1; q_2 = q_1$ if $q_1 <: q_2$ and q_2 otherwise.
Extending the operator to contexts Γ_1 and Γ_2 where $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma_2 \rfloor$, define $\Gamma_0 := \Gamma_1; \Gamma_2$ as $\lfloor \Gamma_0 \rfloor = \lfloor \Gamma_1 \rfloor$ and $\overline{\Gamma_0} = \overline{\Gamma_1}; \overline{\Gamma_2}$ (defined pointwise).
Let $\Gamma_0' = \Gamma_1'; \Gamma_2'$. Then, $\Gamma_0' \vdash a_1 :^1 A_1$ and $\Gamma_0' \vdash a_2 :^1 A_2$ and $\Gamma <: q \cdot \Gamma_0'$.
This case, thereafter, follows by setting $\Gamma' := \Gamma_0'$.
– Rule ST-Proj1. Have: $\Gamma \vdash \mathbf{proj}_1\ a :^q A_1$ where $\Gamma \vdash a :^q A_1 \,\&\, A_2$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash \mathbf{proj}_1\ a :^1 A_1$ and $\Gamma <: q \cdot \Gamma'$.
By IH, $\exists \Gamma_1'$ such that $\Gamma_1' \vdash a :^1 A_1 \,\&\, A_2$ and $\Gamma <: q \cdot \Gamma_1'$.
This case follows by setting $\Gamma' := \Gamma_1'$.
– Rule ST-Proj2. Similar to rule ST-Proj1.
– Rule ST-Inj1. Have $\Gamma \vdash \mathbf{inj}_1\ a_1 :^q A_1 + A_2$ where $\Gamma \vdash a_1 :^q A_1$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash \mathbf{inj}_1\ a_1 :^1 A_1 + A_2$ and $\Gamma <: q \cdot \Gamma'$.
By IH, $\exists \Gamma_1'$ such that $\Gamma_1' \vdash a_1 :^1 A_1$ and $\Gamma <: q \cdot \Gamma_1'$.
This case follows by setting $\Gamma' := \Gamma_1'$.
– Rule ST-Inj2. Similar to rule ST-Inj1.
– Rule ST-Case. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0} A_1 + A_2$ and $\Gamma_2, x_1 :^{q \cdot q_0} A_1 \vdash b_1 :^q B$ and $\Gamma_2, x_2 :^{q \cdot q_0} A_2 \vdash b_2 :^q B$ and $q_0 <: 1$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash \mathbf{case}_{q_0}\ a\, \mathbf{of}\, x_1.b_1\,;\, x_2.b_2 :^1 B$ and $\Gamma_1 + \Gamma_2 <: q \cdot \Gamma'$.
By IH, $\exists \Gamma_1', \Gamma_{21}', \Gamma_{22}'$ and q_1', q_2' such that $\Gamma_1' \vdash a :^1 A_1 + A_2$ and $\Gamma_{21}', x_1 :^{q_1'} A_1 \vdash b_1 :^1 B$ and $\Gamma_{22}', x_2 :^{q_2'} A_2 \vdash b_2 :^1 B$ and $\Gamma_1 <: (q \cdot q_0) \cdot \Gamma_1'$ and $\Gamma_2 <: q \cdot \Gamma_{21}'$ and $\Gamma_2 <: q \cdot \Gamma_{22}'$ and $q \cdot q_0 <: q \cdot q_1'$ and $q \cdot q_0 <: q \cdot q_2'$.
Since $q \neq 0$, therefore $q_0 <: q_1'$ and $q_0 <: q_2'$.
Now, let $\Gamma_2' := \Gamma_{21}'; \Gamma_{22}'$. Then, $\Gamma_2 <: q \cdot \Gamma_2'$.
By rule ST-SubL, $\Gamma_2', x_1 :^{q_0} A_1 \vdash b_1 :^1 B$ and $\Gamma_2', x_2 :^{q_0} A_2 \vdash b_2 :^1 B$.
By Lemma 16, $q_0 \cdot \Gamma_1' \vdash a :^{q_0} A_1 + A_2$.
This case, then, follows by setting $\Gamma' := q_0 \cdot \Gamma_1' + \Gamma_2'$.
– Rule ST-SubL. Have: $\Gamma \vdash a :^q A$ where $\Gamma_1 \vdash a :^q A$ and $\Gamma <: \Gamma_1$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash a :^1 A$ and $\Gamma <: q \cdot \Gamma'$.
By IH, $\exists \Gamma_1'$ such that $\Gamma_1' \vdash a :^1 A$ and $\Gamma_1 <: q \cdot \Gamma_1'$.
This case, then, follows by setting $\Gamma' := \Gamma_1'$.
– Rule ST-SubR. Have: $\Gamma \vdash a :^{q'} A$ where $\Gamma \vdash a :^q A$ and $q <: q'$.
Need to show: $\exists \Gamma'$ such that $\Gamma' \vdash a :^1 A$ and $\Gamma <: q' \cdot \Gamma'$.
Since $q' \neq 0$, therefore $q \neq 0$.
By IH, $\exists \Gamma_1'$ such that $\Gamma_1' \vdash a :^1 A$ and $\Gamma <: q \cdot \Gamma_1'$.
Now, since $q <: q'$, therefore $q \cdot \Gamma_1' <: q' \cdot \Gamma_1'$.
This case, then, follows by setting $\Gamma' := \Gamma_1'$.

Lemma 18 (Splitting (Lemma 3)). *If $\Gamma \vdash a :^{q_1 + q_2} A$, then there exists Γ_1 and Γ_2 such that $\Gamma_1 \vdash a :^{q_1} A$ and $\Gamma_2 \vdash a :^{q_2} A$ and $\Gamma = \Gamma_1 + \Gamma_2$.*

Proof. If $q_1 + q_2 = 0$, then $\Gamma_1 := 0 \cdot \Gamma$ and $\Gamma_2 := \Gamma$.
Otherwise, by Lemma 17, $\exists \Gamma'$ such that $\Gamma' \vdash a :^1 A$ and $\Gamma <: (q_1 + q_2) \cdot \Gamma'$.
Then, $\Gamma = (q_1 + q_2) \cdot \Gamma' + \Gamma_0$ for some Γ_0.
Now, by Lemma 16, $q_1 \cdot \Gamma' \vdash a :^{q_1} A$ and $q_2 \cdot \Gamma' \vdash a :^{q_2} A$.
By rule ST-SubL, $q_2 \cdot \Gamma' + \Gamma_0 \vdash a :^{q_2} A$.
The lemma follows by setting $\Gamma_1 := q_1 \cdot \Gamma'$ and $\Gamma_2 := q_2 \cdot \Gamma' + \Gamma_0$.

Lemma 19 (Weakening (Lemma 4)). *If $\Gamma_1, \Gamma_2 \vdash a :^q A$, then $\Gamma_1, z :^0 C, \Gamma_2 \vdash a :^q A$.*

Proof. By induction on $\Gamma_1, \Gamma_2 \vdash a :^q A$.

Lemma 20 (Substitution (Lemma 5)). *If $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$ and $\Gamma \vdash c :^{r_0} C$ and $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$, then $\Gamma_1 + \Gamma, \Gamma_2 \vdash a\{c/z\} :^q A$.*

Proof. By induction on $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$.

- Rule ST-Var. There are three cases to consider.
  - $0 \cdot \Gamma_{11}, z :^0 C, 0 \cdot \Gamma_{12}, x :^q A, 0 \cdot \Gamma_2 \vdash x :^q A$. Also, $\Gamma \vdash c :^0 C$ where $\lfloor \Gamma_{11} \rfloor = \lfloor \Gamma \rfloor$.
    Need to show: $\Gamma, 0 \cdot \Gamma_{12}, x :^q A, 0 \cdot \Gamma_2 \vdash x :^q A$.
    Follows by rule ST-Var and rule ST-SubL.
  - $0 \cdot \Gamma_1, x :^q A, 0 \cdot \Gamma_2 \vdash x :^q A$. Also, $\Gamma \vdash a :^q A$ where $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$.
    Need to show: $\Gamma, 0 \cdot \Gamma_2 \vdash a :^q A$.
    Follows by lemma 19.
  - $0 \cdot \Gamma_1, x :^q A, 0 \cdot \Gamma_{21}, z :^0 C, 0 \cdot \Gamma_{22} \vdash x :^q A$. Also, $\Gamma_{31}, x :^r A, \Gamma_{32} \vdash c :^0 C$ where $\lfloor \Gamma_{31} \rfloor = \lfloor \Gamma_1 \rfloor$ and $\lfloor \Gamma_{32} \rfloor = \lfloor \Gamma_{21} \rfloor$.
    Need to show: $\Gamma_{31}, x :^{(q+r)} A, \Gamma_{32}, 0 \cdot \Gamma_{22} \vdash x :^q A$.
    Follows by rule ST-Var and rule ST-SubL.
- Rule ST-Lam. Have: $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash \lambda^r x : A.b :^q {}^rA \to B$ where $\Gamma_1, z :^{r_0} C, \Gamma_2, x :^{q \cdot r} A \vdash b :^q B$. Also, $\Gamma \vdash c :^{r_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
  Need to show: $\Gamma_1 + \Gamma, \Gamma_2 \vdash \lambda^r x : A.b\{c/z\} :^q {}^rA \to B$.
  Follows by IH and rule ST-Lam.
- Rule ST-App. Have: $\Gamma_{11} + \Gamma_{12}, z :^{r_{01}+r_{02}} C, \Gamma_{21} + \Gamma_{22} \vdash b\ a^r :^q B$ where $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{21} \vdash b :^q {}^rA \to B$ and $\Gamma_{12}, z :^{r_{02}} C, \Gamma_{22} \vdash a :^{q \cdot r} A$. Also, $\Gamma \vdash c :^{r_{01}+r_{02}} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_{11} \rfloor$.
  Need to show: $\Gamma_{11} + \Gamma_{12} + \Gamma, \Gamma_{21} + \Gamma_{22} \vdash b\{c/z\}\ a\{c/z\}^r :^q B$.
  By lemma 18, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{r_{01}} C$ and $\Gamma_{32} \vdash c :^{r_{02}} C$ and $\Gamma = \Gamma_{31} + \Gamma_{32}$.
  By IH, $\Gamma_{11} + \Gamma_{31}, \Gamma_{21} \vdash b\{c/z\} :^q {}^rA \to B$ and $\Gamma_{12} + \Gamma_{32}, \Gamma_{22} \vdash a\{c/z\} :^{q \cdot r} A$.
  This case, then, follows by rule ST-App.
- Rule ST-WPair. Have: $\Gamma_{11} + \Gamma_{12}, z :^{r_{01}+r_{02}} C, \Gamma_{21} + \Gamma_{22} \vdash (a_1{}^r, a_2) :^q {}^rA_1 \times A_2$ where $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{21} \vdash a_1 :^{q \cdot r} A_1$ and $\Gamma_{12}, z :^{r_{02}} C, \Gamma_{22} \vdash a_2 :^q A_2$. Also, $\Gamma \vdash c :^{r_{01}+r_{02}} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_{11} \rfloor$.
  Need to show: $\Gamma_{11} + \Gamma_{12} + \Gamma, \Gamma_{21} + \Gamma_{22} \vdash (a_1\{c/z\}^r, a_2\{c/z\}) :^q {}^rA_1 \times A_2$.
  By lemma 18, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{r_{01}} C$ and $\Gamma_{32} \vdash c :^{r_{02}} C$ and $\Gamma = \Gamma_{31} + \Gamma_{32}$.
  By IH, $\Gamma_{11} + \Gamma_{31}, \Gamma_{21} \vdash a_1\{c/z\} :^{q \cdot r} A_1$ and $\Gamma_{12} + \Gamma_{32}, \Gamma_{22} \vdash a_2\{c/z\} :^q A_2$.
  This case, then, follows by rule ST-WPair.

– Rule ST-LetPair. Have: $\Gamma_{11} + \Gamma_{12}, z :^{r_{01}+r_{02}} C, \Gamma_{21} + \Gamma_{22} \vdash \mathbf{let}_{q_0}\,(x^r, y) = a\ \mathbf{in}\ b :^q B$ where $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{21} \vdash a :^{q\cdot q_0}\ ^rA_1 \times A_2$ and $\Gamma_{12}, z :^{r_{02}} C, \Gamma_{22}, x :^{q\cdot q_0\cdot r} A_1, y :^{q\cdot q_0} A_2 \vdash b :^q B$. Also, $\Gamma \vdash c :^{r_{01}+r_{02}} C$ where $\lfloor\Gamma\rfloor = \lfloor\Gamma_{11}\rfloor$.
Need to show: $\Gamma_{11}+\Gamma_{12}+\Gamma, \Gamma_{21}+\Gamma_{22} \vdash \mathbf{let}_{q_0}(x^r, y) = a\{c/z\}\mathbf{in}\,b\{c/z\} :^q B$.
By lemma 18, $\exists\Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{r_{01}} C$ and $\Gamma_{32} \vdash c :^{r_{02}} C$ and $\Gamma = \Gamma_{31} + \Gamma_{32}$.
By IH, $\Gamma_{11} + \Gamma_{31}, \Gamma_{21} \vdash a\{c/z\} :^{q\cdot q_0}\ ^rA_1 \times A_2$ and $\Gamma_{12} + \Gamma_{32}, \Gamma_{22}, x :^{q\cdot q_0\cdot r} A_1, y :^{q\cdot q_0} A_2 \vdash b\{c/z\} :^q B$.
This case, then, follows by rule ST-LetPair.
– Rule ST-Unit. Have: $0 \cdot \Gamma_1, z :^0 C, 0 \cdot \Gamma_2 \vdash \mathbf{unit} :^q \mathbf{Unit}$. Also $\Gamma \vdash c :^0 C$.
Need to show: $\Gamma, 0 \cdot \Gamma_2 \vdash \mathbf{unit} :^q \mathbf{Unit}$.
Follows by rule ST-Unit and rule ST-SubL.
– Rule ST-LetUnit. Have: $\Gamma_{11} + \Gamma_{12}, z :^{r_{01}+r_{02}} C, \Gamma_{21} + \Gamma_{22} \vdash \mathbf{let}_{q_0}\,\mathbf{unit} = a\ \mathbf{in}\ b :^q B$ where $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{21} \vdash a :^{q\cdot q_0} \mathbf{Unit}$ and $\Gamma_{12}, z :^{r_{02}} C, \Gamma_{22} \vdash b :^q B$. Also, $\Gamma \vdash c :^{r_{01}+r_{02}} C$ where $\lfloor\Gamma\rfloor = \lfloor\Gamma_{11}\rfloor$.
Need to show: $\Gamma_{11}+\Gamma_{12}+\Gamma, \Gamma_{21}+\Gamma_{22} \vdash \mathbf{let}_{q_0}\,\mathbf{unit} = a\{c/z\}\,\mathbf{in}\,b\{c/z\} :^q B$.
By lemma 18, $\exists\Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{r_{01}} C$ and $\Gamma_{32} \vdash c :^{r_{02}} C$ and $\Gamma = \Gamma_{31} + \Gamma_{32}$.
By IH, $\Gamma_{11} + \Gamma_{31}, \Gamma_{21} \vdash a\{c/z\} :^{q\cdot q_0} \mathbf{Unit}$ and $\Gamma_{12} + \Gamma_{32}, \Gamma_{22} \vdash b\{c/z\} :^q B$.
This case, then, follows by rule ST-LetUnit.
– Rule ST-SPair. Have: $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash (a_1; a_2) :^q A_1 \,\&\, A_2$ where $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a_1 :^q A_1$ and $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a_2 :^q A_2$. Also, $\Gamma \vdash c :^{r_0} C$ where $\lfloor\Gamma\rfloor = \lfloor\Gamma_1\rfloor$.
Need to show: $\Gamma_1 + \Gamma, \Gamma_2 \vdash (a_1\{c/z\}; a_2\{c/z\}) :^q A_1 \,\&\, A_2$.
By IH, $\Gamma_1 + \Gamma, \Gamma_2 \vdash a_1\{c/z\} :^q A_1$ and $\Gamma_1 + \Gamma, \Gamma_2 \vdash a_2\{c/z\} :^q A_2$.
This case, then, follows by rule ST-SPair.
– Rule ST-Proj1. Have: $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash \mathbf{proj}_1\, a :^q A_1$ where $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A_1 \,\&\, A_2$. Also, $\Gamma \vdash c :^{r_0} C$ where $\lfloor\Gamma\rfloor = \lfloor\Gamma_1\rfloor$.
Need to show: $\Gamma_1 + \Gamma, \Gamma_2 \vdash \mathbf{proj}_1\, a\{c/z\} :^q A_1$.
By IH, $\Gamma_1 + \Gamma, \Gamma_2 \vdash a\{c/z\} :^q A_1 \,\&\, A_2$.
This case, then, follows by rule ST-Proj1.
– Rule ST-Proj2. Similar to rule ST-Proj1.
– Rule ST-Inj1. Have: $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash \mathbf{inj}_1\ a_1 :^q A_1 + A_2$ where $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a_1 :^q A_1$. Also, $\Gamma \vdash c :^{r_0} C$ where $\lfloor\Gamma\rfloor = \lfloor\Gamma_1\rfloor$.
Need to show: $\Gamma_1 + \Gamma, \Gamma_2 \vdash \mathbf{inj}_1\ a_1\{c/z\} :^q A_1 + A_2$.
By IH, $\Gamma_1 + \Gamma, \Gamma_2 \vdash a_1\{c/z\} :^q A_1$.
This case, then, follows by rule ST-Inj1.
– Rule ST-Inj2. Similar to rule ST-Inj1.
– Rule ST-Case. Have: $\Gamma_{11}+\Gamma_{12}, z :^{r_{01}+r_{02}} C, \Gamma_{21}+\Gamma_{22} \vdash \mathbf{case}_{q_0}\, a\mathbf{of}\, x_1.b_1; x_2.b_2 :^q B$ where $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{21} \vdash a :^{q\cdot q_0} A_1 + A_2$ and $\Gamma_{12}, z :^{r_{02}} C, \Gamma_{22}, x_1 :^{q\cdot q_0} A_1 \vdash b_1 :^q B$ and $\Gamma_{12}, z :^{r_{02}} C, \Gamma_{22}, x_2 :^{q\cdot q_0} A_2 \vdash b_2 :^q B$. Also, $\Gamma \vdash c :^{r_{01}+r_{02}} C$ where $\lfloor\Gamma\rfloor = \lfloor\Gamma_{11}\rfloor$.
Need to show: $\Gamma_{11}+\Gamma_{12}+\Gamma, \Gamma_{21}+\Gamma_{22} \vdash \mathbf{case}_{q_0}\, a\{c/z\}\mathbf{of}\, x_1.b_1\{c/z\}; x_2.b_2\{c/z\} :^q B$.
By lemma 18, $\exists\Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{r_{01}} C$ and $\Gamma_{32} \vdash c :^{r_{02}} C$ and

$\Gamma = \Gamma_{31} + \Gamma_{32}$.
By IH, $\Gamma_{11} + \Gamma_{31}, \Gamma_{21} \vdash a\{c/z\} :^{q \cdot q_0} A_1 + A_2$ and $\Gamma_{12} + \Gamma_{32}, \Gamma_{22}, x_1 :^{q \cdot q_0} A_1 \vdash b_1\{c/z\} :^q B$ and $\Gamma_{12} + \Gamma_{32}, \Gamma_{22}, x_2 :^{q \cdot q_0} A_2 \vdash b_2\{c/z\} :^q B$.
This case, then, follows by rule ST-Case.

– Rule ST-SubL. Have: $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$ where $\Gamma_1', z :^{r_0'} C, \Gamma_2' \vdash a :^q A$ where $\Gamma_1 <: \Gamma_1'$ and $r_0 <: r_0'$ and $\Gamma_2 <: \Gamma_2'$. Also, $\Gamma \vdash c :^{r_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
Need to show: $\Gamma_1 + \Gamma, \Gamma_2 \vdash a\{c/z\} :^q A$.
Since $r_0 <: r_0'$, by rule ST-SubR, $\Gamma \vdash c :^{r_0'} C$.
By IH, $\Gamma_1' + \Gamma, \Gamma_2' \vdash a\{c/z\} :^q A$.
This case, then, follows by rule ST-SubL.

– Rule ST-SubR. Have: $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^{q'} A$ where $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$ and $q <: q'$. Also, $\Gamma \vdash c :^{r_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
Need to show: $\Gamma_1 + \Gamma, \Gamma_2 \vdash a\{c/z\} :^{q'} A$.
By IH, $\Gamma_1 + \Gamma, \Gamma_2 \vdash a\{c/z\} :^q A$.
This case, then, follows by rule ST-SubR.

Theorem 15 (Preservation (Theorem 1)). *If $\Gamma \vdash a :^q A$ and $\vdash a \rightsquigarrow a'$, then $\Gamma \vdash a' :^q A$.*

Proof. By induction on $\Gamma \vdash a :^q A$ and inversion on $\vdash a \rightsquigarrow a'$.

– Rule ST-App. Have: $\Gamma_1 + \Gamma_2 \vdash b\ a^r :^q B$ where $\Gamma_1 \vdash b :^q {}^rA \to B$ and $\Gamma_2 \vdash a :^{q \cdot r} A$.
Let $\vdash b\ a^r \rightsquigarrow c$. By inversion:

- $\vdash b\ a^r \rightsquigarrow b'\ a^r$ when $\vdash b \rightsquigarrow b'$.
Need to show: $\Gamma_1 + \Gamma_2 \vdash b'\ a^r :^q B$.
Follows by IH and rule ST-App.
- $b = \lambda^r x : A'.b'$ and $\vdash b\ a^r \rightsquigarrow b'\{a/x\}$.
Need to show: $\Gamma_1 + \Gamma_2 \vdash b'\{a/x\} :^q B$.
By inversion on $\Gamma_1 \vdash \lambda^r x : A'.b' :^q {}^rA \to B$, we get $A' = A$ and $\Gamma_1, x :^{q_0 \cdot r} A \vdash b' :^{q_0} B$ for some $q_0 <: q$.
Now, there are two cases to consider.
  * $q_0 = 0$. Since $q_0 <: q$, so $q = 0$.
  Then, by the substitution lemma, $\Gamma_1 + \Gamma_2 \vdash b'\{a/x\} :^0 B$.
  * $q_0 \neq 0$. By lemma 17, $\exists \Gamma_1'$ and r' such that $\Gamma_1', x :^{r'} A \vdash b' :^1 B$ and $\Gamma_1 <: q_0 \cdot \Gamma_1'$ and $q_0 \cdot r <: q_0 \cdot r'$.
  Since $q_0 \neq 0$, therefore $r <: r'$. Hence, by rule ST-SubL, $\Gamma_1', x :^r A \vdash b' :^1 B$.
  Now, by lemma 16, $q \cdot \Gamma_1', x :^{q \cdot r} A \vdash b' :^q B$. By rule ST-SubL, $\Gamma_1, x :^{q \cdot r} A \vdash b' :^q B$.
  This case, then, follows by the substitution lemma.

– Rule ST-LetPair. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0} {}^rA_1 \times A_2$ and $\Gamma_2, x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2 \vdash b :^q B$.
Let $\vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \rightsquigarrow c$. By inversion:

- $\vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \rightsquigarrow \mathbf{let}_{q_0}\ (x^r, y) = a'\ \mathbf{in}\ b$ when $\vdash a \rightsquigarrow a'$.
  Need to show: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ (x^r, y) = a'\ \mathbf{in}\ b :^q B$.
  Follows by IH and rule ST-LetPair.
- $\vdash \mathbf{let}_{q_0}\ (x^r, y) = ({a_1}^r, a_2)\ \mathbf{in}\ b \rightsquigarrow b\{a_1/x\}\{a_2/y\}$.
  Need to show: $\Gamma_1 + \Gamma_2 \vdash b\{a_1/x\}\{a_2/y\} :^q B$.
  By inversion on $\Gamma_1 \vdash ({a_1}^r, a_2) :^{q \cdot q_0}\ {}^r A_1 \times A_2$, we have:
  $\exists \Gamma_{11}, \Gamma_{12}$ such that $\Gamma_{11} \vdash a_1 :^{q \cdot q_0 \cdot r} A_1$ and $\Gamma_{12} \vdash a_2 :^{q \cdot q_0} A_2$ and $\Gamma_1 = \Gamma_{11} + \Gamma_{12}$.
  This case, then, follows by applying the substitution lemma twice.

– Rule ST-LetUnit. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0} \mathbf{Unit}$ and $\Gamma_2 \vdash b :^q B$.
  Let $\vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b \rightsquigarrow c$. By inversion:
  - $\vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b \rightsquigarrow \mathbf{let}_{q_0}\ \mathbf{unit} = a'\ \mathbf{in}\ b$ when $\vdash a \rightsquigarrow a'$.
    Need to show: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a'\ \mathbf{in}\ b :^q B$.
    Follows by IH and rule ST-LetUnit.
  - $\vdash \mathbf{let}_{q_0}\ \mathbf{unit} = \mathbf{unit}\ \mathbf{in}\ b \rightsquigarrow b$.
    Need to show: $\Gamma_1 + \Gamma_2 \vdash b :^q B$.
    Follows by rule ST-SubL.

– Rule ST-Proj1. Have: $\Gamma \vdash \mathbf{proj}_1\ a :^q A_1$ where $\Gamma \vdash a :^q A_1 \,\&\, A_2$.
  Let $\vdash \mathbf{proj}_1\ a \rightsquigarrow c$. By inversion:
  - $\vdash \mathbf{proj}_1\ a \rightsquigarrow \mathbf{proj}_1\ a'$ when $\vdash a \rightsquigarrow a'$.
    Need to show: $\Gamma \vdash \mathbf{proj}_1\ a' :^q A_1$.
    Follows by IH and rule ST-Proj1.
  - $\vdash \mathbf{proj}_1\ (a_1; a_2) \rightsquigarrow a_1$.
    Need to show: $\Gamma \vdash a_1 :^q A_1$.
    Follows by inversion on $\Gamma \vdash (a_1; a_2) :^q A_1 \,\&\, A_2$.

– Rule ST-Proj2. Similar to rule ST-Proj1.

– Rule ST-Case. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0} A_1 + A_2$ and $\Gamma_2, x_1 :^{q \cdot q_0} A_1 \vdash b_1 :^q B$ and $\Gamma_2, x_2 :^{q \cdot q_0} A_2 \vdash b_2 :^q B$.
  Let $\vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow c$. By inversion:
  - $\vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow \mathbf{case}_{q_0}\ a'\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2$ when $\vdash a \rightsquigarrow a'$.
    Need to show: $\Gamma_1 + \Gamma_2 \vdash \mathbf{case}_{q_0}\ a'\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B$.
    Follows by IH and rule ST-Case.
  - $\vdash \mathbf{case}_{q_0}\ (\mathbf{inj}_1\ a_1)\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_1\{a_1/x_1\}$.
    Need to show $\Gamma_1 + \Gamma_2 \vdash b_1\{a_1/x_1\} :^q B$.
    By inversion on $\Gamma_1 \vdash \mathbf{inj}_1\ a_1 :^{q \cdot q_0} A_1 + A_2$, we have $\Gamma_1 \vdash a_1 :^{q \cdot q_0} A_1$.
    This case, then, follows by applying the substitution lemma.
  - $\vdash \mathbf{case}_{q_0}\ (\mathbf{inj}_2\ a_2)\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_2\{a_2/x_2\}$.
    Similar to the previous case.

– Rules ST-SubL and ST-SubR. Follows by IH.

Theorem 16 (Progress (Theorem 2)). *If $\emptyset \vdash a :^q A$, then either a is a value or there exists a' such that $\vdash a \rightsquigarrow a'$.*

Proof. By induction on $\emptyset \vdash a :^q A$.

- Rule ST-Var. Does not apply since the context here is empty.
- Rule ST-App. Have: $\Gamma_1 + \Gamma_2 \vdash b\ a^r :^q B$ where $\Gamma_1 \vdash b :^q {}^rA \to B$ and $\Gamma_2 \vdash a :^{q \cdot r} A$.
  Need to show: $\exists c, \vdash b\ a^r \rightsquigarrow c$.
  By IH, b is either a value or $\vdash b \rightsquigarrow b'$.
  If b is a value, then $b = \lambda^r x : A.b'$ for some b'. Therefore, $\vdash b\ a^r \rightsquigarrow b'\{a/x\}$.
  Otherwise, $\vdash b\ a^r \rightsquigarrow b'\ a^r$.
- Rule ST-LetPair. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0}\ {}^rA_1 \times A_2$ and $\Gamma_2, x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2 \vdash b :^q B$.
  Need to show: $\exists c, \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \rightsquigarrow c$.
  By IH, a is either a value or $\vdash a \rightsquigarrow a'$.
  If a is a value, then $a = ({a_1}^r, a_2)$. Therefore, $\vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \rightsquigarrow b\{a_1/x\}\{a_2/y\}$.
  Otherwise, $\vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \rightsquigarrow \mathbf{let}_{q_0}\ (x^r, y) = a'\ \mathbf{in}\ b$.
- Rule ST-LetUnit. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0} \mathbf{Unit}$ and $\Gamma_2 \vdash b :^q B$.
  Need to show: $\exists c, \vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b \rightsquigarrow c$.
  By IH, a is either a value or $\vdash a \rightsquigarrow a'$.
  If a is a value, then $a = \mathbf{unit}$. Therefore, $\vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b \rightsquigarrow b$.
  Otherwise, $\vdash \mathbf{let}_{q_0}\ \mathbf{unit} = a\ \mathbf{in}\ b \rightsquigarrow \mathbf{let}_{q_0}\ \mathbf{unit} = a'\ \mathbf{in}\ b$.
- Rule ST-Proj1. Have: $\Gamma \vdash \mathbf{proj}_1\ a :^q A_1$ where $\Gamma \vdash a :^q A_1 \,\&\, A_2$.
  Need to show: $\exists c, \vdash \mathbf{proj}_1\ a \rightsquigarrow c$.
  By IH, a is either a value or $\vdash a \rightsquigarrow a'$.
  If a is a value, then $a = (a_1; a_2)$. Therefore, $\vdash \mathbf{proj}_1\ a \rightsquigarrow a_1$.
  Otherwise, $\vdash \mathbf{proj}_1\ a \rightsquigarrow \mathbf{proj}_1\ a'$.
- Rule ST-Proj2. Similar to rule ST-Proj1.
- Rule ST-Case. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B$ where $\Gamma_1 \vdash a :^{q \cdot q_0} A_1 + A_2$ and $\Gamma_2, x_1 :^{q \cdot q_0} A_1 \vdash b_1 :^q B$ and $\Gamma_2, x_2 :^{q \cdot q_0} A_2 \vdash b_2 :^q B$.
  Need to show: $\exists c, \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow c$.
  By IH, a is either a value or $\vdash a \rightsquigarrow a'$.
  If a is a value, then $a = \mathbf{inj}_1\ a_1$ or $a = \mathbf{inj}_2\ a_2$.
  Then, $\vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_1\{a_1/x_1\}$ or $\vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_2\{a_2/x_2\}$.
  Otherwise, $\vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow \mathbf{case}_{q_0}\ a'\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2$.
- Rules ST-SubL and ST-SubR. Follows by IH.
- Rules ST-Lam, ST-WPair, ST-Unit, ST-SPair, ST-Inj1, and ST-Inj2. The terms typed by these rules are values.

C Dependency Analysis in Simply-Typed LDC

Lemma 21 (Multiplication (Lemma 6)). *If $\Gamma \vdash a :^\ell A$, then $m_0 \sqcup \Gamma \vdash a :^{m_0 \sqcup \ell} A$.*

Proof. By induction on $\Gamma \vdash a :^\ell A$.

– Rule ST-VarD. Have: $0 \sqcup \Gamma_1, x :^{\ell} A, 0 \sqcup \Gamma_2 \vdash x :^{\ell} A$.
Need to show: $0 \sqcup \Gamma_1, x :^{m_0 \sqcup \ell} A, 0 \sqcup \Gamma_2 \vdash x :^{m_0 \sqcup \ell} A$.
This case follows by rule ST-VarD.

– Rule ST-LamD. Have: $\Gamma \vdash \lambda^m x : A.b :^{\ell}\ ^m A \to B$ where $\Gamma, x :^{\ell \sqcup m} A \vdash b :^{\ell} B$.
Need to show: $m_0 \sqcup \Gamma \vdash \lambda^m x : A.b :^{m_0 \sqcup \ell}\ ^m A \to B$.
By IH, $m_0 \sqcup \Gamma, x :^{m_0 \sqcup (\ell \sqcup m)} A \vdash b :^{m_0 \sqcup \ell} B$.
This case, then, follows by rule ST-LamD using associativity of $\sqcup$.

– Rule ST-AppD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash b\ a^m :^{\ell} B$ where $\Gamma_1 \vdash b :^{\ell}\ ^m A \to B$ and $\Gamma_2 \vdash a :^{\ell \sqcup m} A$.
Need to show: $m_0 \sqcup (\Gamma_1 \sqcap \Gamma_2) \vdash b\ a^m :^{m_0 \sqcup \ell} B$.
By IH, $m_0 \sqcup \Gamma_1 \vdash b :^{m_0 \sqcup \ell}\ ^m A \to B$ and $m_0 \sqcup \Gamma_2 \vdash a :^{m_0 \sqcup (\ell \sqcup m)} A$.
By rule ST-AppD, using associativity of $\sqcup$, $(m_0 \sqcup \Gamma_1) \sqcap (m_0 \sqcup \Gamma_2) \vdash b\, a^m :^{m_0 \sqcup \ell} B$.
Now, for elements ℓ_1, ℓ_2 and ℓ_3 of any lattice, $\ell_1 \sqcup (\ell_2 \sqcap \ell_3) \sqsubseteq (\ell_1 \sqcup \ell_2) \sqcap (\ell_1 \sqcup \ell_3)$.
This case, then, follows by rule ST-SubLD, using the above relation.

– Rule ST-WPairD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash ({a_1}^m, a_2) :^{\ell}\ ^m A_1 \times A_2$ where $\Gamma_1 \vdash a_1 :^{\ell \sqcup m} A_1$ and $\Gamma_2 \vdash a_2 :^{\ell} A_2$.
Need to show: $m_0 \sqcup (\Gamma_1 \sqcap \Gamma_2) \vdash ({a_1}^m, a_2) :^{m_0 \sqcup \ell}\ ^m A_1 \times A_2$.
By IH, $m_0 \sqcup \Gamma_1 \vdash a_1 :^{m_0 \sqcup (\ell \sqcup m)} A_1$ and $m_0 \sqcup \Gamma_2 \vdash a_2 :^{m_0 \sqcup \ell} A_2$.
By rule ST-WPairD, using associativity of $\sqcup$, $(m_0 \sqcup \Gamma_1) \sqcap (m_0 \sqcup \Gamma_2) \vdash ({a_1}^m, a_2) :^{m_0 \sqcup \ell}\ ^m A_1 \times A_2$.
For ℓ_1, ℓ_2, ℓ_3, we have, $\ell_1 \sqcup (\ell_2 \sqcap \ell_3) \sqsubseteq (\ell_1 \sqcup \ell_2) \sqcap (\ell_1 \sqcup \ell_3)$.
This case follows by rule ST-SubLD, using the above relation.

– Rule ST-LetPairD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{let}\ (x^m, y) = a\ \mathbf{in}\ b :^{\ell} B$ where $\Gamma_1 \vdash a :^{\ell}\ ^m A_1 \times A_2$ and $\Gamma_2, x :^{\ell \sqcup m} A_1, y :^{\ell} A_2 \vdash b :^{\ell} B$.
Need to show: $m_0 \sqcup (\Gamma_1 \sqcap \Gamma_2) \vdash \mathbf{let}\ (x^m, y) = a\ \mathbf{in}\ b :^{m_0 \sqcup \ell} B$.
By IH, $m_0 \sqcup \Gamma_1 \vdash a :^{m_0 \sqcup \ell}\ ^m A_1 \times A_2$ and $m_0 \sqcup \Gamma_2, x :^{m_0 \sqcup (\ell \sqcup m)} A_1, y :^{m_0 \sqcup \ell} A_2 \vdash b :^{m_0 \sqcup \ell} B$.
This case follows by rules ST-LetPairD and ST-SubLD, using associativity of $\sqcup$ and the distributive inequality.

– Rule ST-UnitD. Have: $0 \sqcup \Gamma \vdash \mathbf{unit} :^{\ell} \mathbf{Unit}$.
Need to show: $0 \sqcup \Gamma \vdash \mathbf{unit} :^{m_0 \sqcup \ell} \mathbf{Unit}$.
Follows by rule ST-UnitD.

– Rule ST-LetUnitD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{let\ unit} = a\ \mathbf{in}\ b :^{\ell} B$ where $\Gamma_1 \vdash a :^{\ell} \mathbf{Unit}$ and $\Gamma_2 \vdash b :^{\ell} B$.
Need to show: $m_0 \sqcup (\Gamma_1 \sqcap \Gamma_2) \vdash \mathbf{let\ unit} = a\ \mathbf{in}\ b :^{m_0 \sqcup \ell} B$.
By IH, $m_0 \sqcup \Gamma_1 \vdash a :^{m_0 \sqcup \ell} \mathbf{Unit}$ and $m_0 \sqcup \Gamma_2 \vdash b :^{m_0 \sqcup \ell} B$.
This case follows by rules ST-LetUnitD and ST-SubLD, using the distributive inequality.

– Rule ST-SPairD. Have: $\Gamma \vdash (a_1; a_2) :^{\ell} A_1 \,\&\, A_2$ where $\Gamma \vdash a_1 :^{\ell} A_1$ and $\Gamma \vdash a_2 :^{\ell} A_2$.
Need to show: $m_0 \sqcup \Gamma \vdash (a_1; a_2) :^{m_0 \sqcup \ell} A_1 \,\&\, A_2$.
By IH, $m_0 \sqcup \Gamma \vdash a_1 :^{m_0 \sqcup \ell} A_1$ and $m_0 \sqcup \Gamma \vdash a_2 :^{m_0 \sqcup \ell} A_2$.
This case, then, follows by rule ST-SPairD.

– Rule ST-Proj1D. Have: $\Gamma \vdash \mathbf{proj}_1\ a :^{\ell} A_1$ where $\Gamma \vdash a :^{\ell} A_1 \,\&\, A_2$.
Need to show: $m_0 \sqcup \Gamma \vdash \mathbf{proj}_1\ a :^{m_0 \sqcup \ell} A_1$.
By IH, $m_0 \sqcup \Gamma \vdash a :^{m_0 \sqcup \ell} A_1 \,\&\, A_2$.
This case, then, follows by rule ST-Proj1D.
– Rule ST-Proj2D. Similar to rule ST-Proj1D.
– Rule ST-Inj1D. Have: $\Gamma \vdash \mathbf{inj}_1\ a_1 :^{\ell} A_1 + A_2$ where $\Gamma \vdash a_1 :^{\ell} A_1$.
Need to show: $m_0 \sqcup \Gamma \vdash \mathbf{inj}_1\ a_1 :^{m_0 \sqcup \ell} A_1 + A_2$.
By IH, $m_0 \sqcup \Gamma \vdash a_1 :^{m_0 \sqcup \ell} A_1$.
This case, then, follows by rule ST-Inj1D.
– Rule ST-Inj2D. Similar to rule ST-Inj1D.
– Rule ST-CaseD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{case}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^{\ell} B$ where $\Gamma_1 \vdash a :^{\ell} A_1 + A_2$ and $\Gamma_2, x_1 :^{\ell} A_1 \vdash b_1 :^{\ell} B$ and $\Gamma_2, x_2 :^{\ell} A_2 \vdash b_2 :^{\ell} B$.
Need to show: $m_0 \sqcup (\Gamma_1 \sqcap \Gamma_2) \vdash \mathbf{case}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^{m_0 \sqcup \ell} B$.
By IH, $m_0 \sqcup \Gamma_1 \vdash a_1 :^{m_0 \sqcup \ell} A_1 + A_2$ and $m_0 \sqcup \Gamma_2, x_1 :^{m_0 \sqcup \ell} A_1 \vdash b_1 :^{m_0 \sqcup \ell} B$ and $m_0 \sqcup \Gamma_2, x_2 :^{m_0 \sqcup \ell} A_2 \vdash b_2 :^{m_0 \sqcup \ell} B$.
This case follows by rules ST-CaseD and ST-SubLD using the distributive inequality.
– Rule ST-SubLD. Have: $\Gamma \vdash a :^{\ell} A$ where $\Gamma' \vdash a :^{\ell} A$ and $\Gamma \sqsubseteq \Gamma'$.
Need to show: $m_0 \sqcup \Gamma \vdash a :^{m_0 \sqcup \ell} A$.
By IH, $m_0 \sqcup \Gamma' \vdash a :^{m_0 \sqcup \ell} A$.
Since $\Gamma \sqsubseteq \Gamma'$, so $m_0 \sqcup \Gamma \sqsubseteq m_0 \sqcup \Gamma'$.
This case follows by rule ST-SubLD.
– Rule ST-SubRD. Have: $\Gamma \vdash a :^{\ell'} A$ where $\Gamma \vdash a :^{\ell} A$ and $\ell \sqsubseteq \ell'$.
Need to show: $m_0 \sqcup \Gamma \vdash a :^{m_0 \sqcup \ell'} A$.
By IH, $m_0 \sqcup \Gamma \vdash a :^{m_0 \sqcup \ell} A$.
Since $\ell \sqsubseteq \ell'$, so $m_0 \sqcup \ell \sqsubseteq m_0 \sqcup \ell'$.
This case follows by rule ST-SubRD.

Lemma 22 (Splitting (Lemma 7)). *If $\Gamma \vdash a :^{q_1 \sqcap q_2} A$, then there exists Γ_1 and Γ_2 such that $\Gamma_1 \vdash a :^{q_1} A$ and $\Gamma_2 \vdash a :^{q_2} A$ and $\Gamma = \Gamma_1 \sqcap \Gamma_2$.*

Proof. Have: $\Gamma \vdash a :^{q_1 \sqcap q_2} A$. By rule ST-SubRD, $\Gamma \vdash a :^{q_1} A$ and $\Gamma \vdash a :^{q_2} A$. The lemma follows by setting $\Gamma_1 := \Gamma$ and $\Gamma_2 := \Gamma$.

Lemma 23 (Weakening (Lemma 8)). *If $\Gamma_1, \Gamma_2 \vdash a :^{\ell} A$, then $\Gamma_1, z :^{\top} C, \Gamma_2 \vdash a :^{\ell} A$.*

Proof. By induction on $\Gamma_1, \Gamma_2 \vdash a :^{\ell} A$.

Lemma 24 (Substitution (Lemma 9)). *If $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a :^{\ell} A$ and $\Gamma \vdash c :^{m_0} C$ and $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$, then $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash a\{c/z\} :^{\ell} A$.*

Proof. By induction on $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a :^{\ell} A$.

– Rule ST-VarD. There are three cases to consider.
  • $0 \sqcup \Gamma_{11}, z :^{0} C, 0 \sqcup \Gamma_{12}, x :^{\ell} A, 0 \sqcup \Gamma_2 \vdash x :^{\ell} A$. Also, $\Gamma \vdash c :^{0} C$ where $\lfloor \Gamma_{11} \rfloor = \lfloor \Gamma \rfloor$.
  Need to show: $\Gamma, 0 \sqcup \Gamma_{12}, x :^{\ell} A, 0 \sqcup \Gamma_2 \vdash x :^{\ell} A$.
  Follows by rule ST-VarD and rule ST-SubLD.

- $0 \sqcup \Gamma_1, x :^{\ell} A, 0 \sqcup \Gamma_2 \vdash x :^{\ell} A$. Also, $\Gamma \vdash a :^{\ell} A$ where $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$.
  Need to show: $\Gamma, 0 \sqcup \Gamma_2 \vdash a :^{\ell} A$.
  Follows by lemma 23.
- $0 \sqcup \Gamma_1, x :^{\ell} A, 0 \sqcup \Gamma_{21}, z :^{0} C, 0 \sqcup \Gamma_{22} \vdash x :^{\ell} A$. Also, $\Gamma_{31}, x :^{m} A, \Gamma_{32} \vdash c :^{0} C$ where $\lfloor \Gamma_{31} \rfloor = \lfloor \Gamma_1 \rfloor$ and $\lfloor \Gamma_{32} \rfloor = \lfloor \Gamma_{21} \rfloor$.
  Need to show: $\Gamma_{31}, x :^{(\ell \sqcap m)} A, \Gamma_{32}, 0 \sqcup \Gamma_{22} \vdash x :^{\ell} A$.
  Follows by rule ST-VarD and rule ST-SubLD.

– Rule ST-LamD. Have: $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash \lambda^m x : A.b :^{\ell} {}^m A \to B$ where $\Gamma_1, z :^{m_0} C, \Gamma_2, x :^{\ell \sqcup m} A \vdash b :^{\ell} B$. Also, $\Gamma \vdash c :^{m_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
  Need to show: $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash \lambda^m x : A.b\{c/z\} :^{\ell} {}^m A \to B$.
  Follows by IH and rule ST-LamD.
– Rule ST-AppD. Have: $\Gamma_{11} \sqcap \Gamma_{12}, z :^{m_{01} \sqcap m_{02}} C, \Gamma_{21} \sqcap \Gamma_{22} \vdash b\, a^m :^{\ell} B$ where $\Gamma_{11}, z :^{m_{01}} C, \Gamma_{21} \vdash b :^{\ell} {}^m A \to B$ and $\Gamma_{12}, z :^{m_{02}} C, \Gamma_{22} \vdash a :^{\ell \sqcup m} A$. Also, $\Gamma \vdash c :^{m_{01} \sqcap m_{02}} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_{11} \rfloor$.
  Need to show: $\Gamma_{11} \sqcap \Gamma_{12} \sqcap \Gamma, \Gamma_{21} \sqcap \Gamma_{22} \vdash b\{c/z\}\ a\{c/z\}^m :^{\ell} B$.
  By lemma 22, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{m_{01}} C$ and $\Gamma_{32} \vdash c :^{m_{02}} C$ and $\Gamma = \Gamma_{31} \sqcap \Gamma_{32}$.
  By IH, $\Gamma_{11} \sqcap \Gamma_{31}, \Gamma_{21} \vdash b\{c/z\} :^{\ell} {}^m A \to B$ and $\Gamma_{12} \sqcap \Gamma_{32}, \Gamma_{22} \vdash a\{c/z\} :^{\ell \sqcup m} A$.
  This case, then, follows by rule ST-AppD.
– Rule ST-WPairD. Have: $\Gamma_{11} \sqcap \Gamma_{12}, z :^{m_{01} \sqcap m_{02}} C, \Gamma_{21} \sqcap \Gamma_{22} \vdash (a_1{}^m, a_2) :^{\ell} {}^m A_1 \times A_2$ where $\Gamma_{11}, z :^{m_{01}} C, \Gamma_{21} \vdash a_1 :^{\ell \sqcup m} A_1$ and $\Gamma_{12}, z :^{m_{02}} C, \Gamma_{22} \vdash a_2 :^{\ell} A_2$. Also, $\Gamma \vdash c :^{m_{01} \sqcap m_{02}} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_{11} \rfloor$.
  Need to show: $\Gamma_{11} \sqcap \Gamma_{12} \sqcap \Gamma, \Gamma_{21} \sqcap \Gamma_{22} \vdash (a_1\{c/z\}^r, a_2\{c/z\}) :^{\ell} {}^m A_1 \times A_2$.
  By lemma 22, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{m_{01}} C$ and $\Gamma_{32} \vdash c :^{m_{02}} C$ and $\Gamma = \Gamma_{31} \sqcap \Gamma_{32}$.
  By IH, $\Gamma_{11} \sqcap \Gamma_{31}, \Gamma_{21} \vdash a_1\{c/z\} :^{\ell \sqcup m} A_1$ and $\Gamma_{12} \sqcap \Gamma_{32}, \Gamma_{22} \vdash a_2\{c/z\} :^{\ell} A_2$.
  This case, then, follows by rule ST-WPairD.
– Rule ST-LetPairD. Have: $\Gamma_{11} \sqcap \Gamma_{12}, z :^{m_{01} \sqcap m_{02}} C, \Gamma_{21} \sqcap \Gamma_{22} \vdash \mathbf{let}(x^m, y) = a\ \mathbf{in}\ b :^{\ell} B$ where $\Gamma_{11}, z :^{m_{01}} C, \Gamma_{21} \vdash a :^{\ell} {}^m A_1 \times A_2$ and $\Gamma_{12}, z :^{m_{02}} C, \Gamma_{22}, x :^{\ell \sqcup m} A_1, y :^{\ell} A_2 \vdash b :^{\ell} B$. Also, $\Gamma \vdash c :^{m_{01} \sqcap m_{02}} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_{11} \rfloor$.
  Need to show: $\Gamma_{11} \sqcap \Gamma_{12} \sqcap \Gamma, \Gamma_{21} \sqcap \Gamma_{22} \vdash \mathbf{let}\,(x^m, y) = a\{c/z\}\,\mathbf{in}\,b\{c/z\} :^{\ell} B$.
  By lemma 22, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{m_{01}} C$ and $\Gamma_{32} \vdash c :^{m_{02}} C$ and $\Gamma = \Gamma_{31} \sqcap \Gamma_{32}$.
  By IH, $\Gamma_{11} \sqcap \Gamma_{31}, \Gamma_{21} \vdash a\{c/z\} :^{\ell} {}^m A_1 \times A_2$ and $\Gamma_{12} \sqcap \Gamma_{32}, \Gamma_{22}, x :^{\ell \sqcup m} A_1, y :^{\ell} A_2 \vdash b\{c/z\} :^{\ell} B$.
  This case, then, follows by rule ST-LetPairD.
– Rule ST-UnitD. Have: $0 \sqcup \Gamma_1, z :^{0} C, 0 \sqcup \Gamma_2 \vdash \mathbf{unit} :^{\ell} \mathbf{Unit}$. Also $\Gamma \vdash c :^{0} C$.
  Need to show: $\Gamma, 0 \sqcup \Gamma_2 \vdash \mathbf{unit} :^{\ell} \mathbf{Unit}$.
  Follows by rule ST-UnitD and rule ST-SubLD.
– Rule ST-LetUnitD. Have: $\Gamma_{11} \sqcap \Gamma_{12}, z :^{m_{01} \sqcap m_{02}} C, \Gamma_{21} \sqcap \Gamma_{22} \vdash \mathbf{let\,unit} = a\ \mathbf{in}\ b :^{\ell} B$ where $\Gamma_{11}, z :^{m_{01}} C, \Gamma_{21} \vdash a :^{\ell} \mathbf{Unit}$ and $\Gamma_{12}, z :^{m_{02}} C, \Gamma_{22} \vdash b :^{\ell} B$. Also, $\Gamma \vdash c :^{m_{01} \sqcap m_{02}} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_{11} \rfloor$.
  Need to show: $\Gamma_{11} \sqcap \Gamma_{12} \sqcap \Gamma, \Gamma_{21} \sqcap \Gamma_{22} \vdash \mathbf{let\,unit} = a\{c/z\}\,\mathbf{in}\,b\{c/z\} :^{\ell} B$.
  By lemma 22, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{m_{01}} C$ and $\Gamma_{32} \vdash c :^{m_{02}} C$ and

$\Gamma = \Gamma_{31} \sqcap \Gamma_{32}$.
By IH, $\Gamma_{11} \sqcap \Gamma_{31}, \Gamma_{21} \vdash a\{c/z\} :^{\ell} \mathbf{Unit}$ and $\Gamma_{12} \sqcap \Gamma_{32}, \Gamma_{22} \vdash b\{c/z\} :^{\ell} B$.
This case, then, follows by rule ST-LetUnitD.

- Rule ST-SPairD. Have: $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash (a_1; a_2) :^{\ell} A_1 \,\&\, A_2$ where $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a_1 :^{\ell} A_1$ and $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a_2 :^{\ell} A_2$. Also, $\Gamma \vdash c :^{m_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
  Need to show: $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash (a_1\{c/z\}; a_2\{c/z\}) :^{\ell} A_1 \,\&\, A_2$.
  By IH, $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash a_1\{c/z\} :^{\ell} A_1$ and $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash a_2\{c/z\} :^{\ell} A_2$.
  This case, then, follows by rule ST-SPairD.
- Rule ST-Proj1D. Have: $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash \mathbf{proj}_1\ a :^{\ell} A_1$ where $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a :^{\ell} A_1 \,\&\, A_2$. Also, $\Gamma \vdash c :^{m_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
  Need to show: $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash \mathbf{proj}_1\ a\{c/z\} :^{\ell} A_1$.
  By IH, $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash a\{c/z\} :^{\ell} A_1 \,\&\, A_2$.
  This case, then, follows by rule ST-Proj1D.
- Rule ST-Proj2D. Similar to rule ST-Proj1D.
- Rule ST-Inj1D. Have: $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash \mathbf{inj}_1\ a_1 :^{\ell} A_1 + A_2$ where $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a_1 :^{\ell} A_1$. Also, $\Gamma \vdash c :^{m_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
  Need to show: $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash \mathbf{inj}_1\ a_1\{c/z\} :^{\ell} A_1 + A_2$.
  By IH, $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash a_1\{c/z\} :^{\ell} A_1$.
  This case, then, follows by rule ST-Inj1D.
- Rule ST-Inj2D. Similar to rule ST-Inj1D.
- Rule ST-CaseD. Have: $\Gamma_{11} \sqcap \Gamma_{12}, z :^{m_{01} \sqcap m_{02}} C, \Gamma_{21} \sqcap \Gamma_{22} \vdash \mathbf{case} a \mathbf{of} x_1.b_1; x_2.b_2 :^{\ell} B$ where $\Gamma_{11}, z :^{m_{01}} C, \Gamma_{21} \vdash a :^{\ell} A_1 + A_2$ and $\Gamma_{12}, z :^{m_{02}} C, \Gamma_{22}, x_1 :^{\ell} A_1 \vdash b_1 :^{\ell} B$ and $\Gamma_{12}, z :^{m_{02}} C, \Gamma_{22}, x_2 :^{\ell} A_2 \vdash b_2 :^{\ell} B$. Also, $\Gamma \vdash c :^{m_{01} \sqcap m_{02}} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_{11} \rfloor$.
  Need to show: $\Gamma_{11} \sqcap \Gamma_{12} \sqcap \Gamma, \Gamma_{21} \sqcap \Gamma_{22} \vdash \mathbf{case} a\{c/z\} \mathbf{of} x_1.b_1\{c/z\}; x_2.b_2\{c/z\} :^{\ell} B$.
  By lemma 22, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{m_{01}} C$ and $\Gamma_{32} \vdash c :^{m_{02}} C$ and $\Gamma = \Gamma_{31} \sqcap \Gamma_{32}$.
  By IH, $\Gamma_{11} \sqcap \Gamma_{31}, \Gamma_{21} \vdash a\{c/z\} :^{\ell} A_1 + A_2$ and $\Gamma_{12} \sqcap \Gamma_{32}, \Gamma_{22}, x_1 :^{\ell} A_1 \vdash b_1\{c/z\} :^{\ell} B$ and $\Gamma_{12} \sqcap \Gamma_{32}, \Gamma_{22}, x_2 :^{\ell} A_2 \vdash b_2\{c/z\} :^{\ell} B$.
  This case, then, follows by rule ST-CaseD.
- Rule ST-SubLD. Have: $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a :^{\ell} A$ where $\Gamma_1', z :^{m_0'} C, \Gamma_2' \vdash a :^{\ell} A$ where $\Gamma_1 \sqsubseteq \Gamma_1'$ and $m_0 \sqsubseteq m_0'$ and $\Gamma_2 \sqsubseteq \Gamma_2'$. Also, $\Gamma \vdash c :^{m_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
  Need to show: $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash a\{c/z\} :^{\ell} A$.
  Since $m_0 \sqsubseteq m_0'$, by rule ST-SubRD, $\Gamma \vdash c :^{m_0'} C$.
  By IH, $\Gamma_1' \sqcap \Gamma, \Gamma_2' \vdash a\{c/z\} :^{\ell} A$.
  This case, then, follows by rule ST-SubLD.
- Rule ST-SubRD. Have: $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a :^{\ell'} A$ where $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a :^{\ell} A$ and $\ell \sqsubseteq \ell'$. Also, $\Gamma \vdash c :^{m_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
  Need to show: $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash a\{c/z\} :^{\ell'} A$.
  By IH, $\Gamma_1 \sqcap \Gamma, \Gamma_2 \vdash a\{c/z\} :^{\ell} A$.
  This case, then, follows by rule ST-SubRD.

Lemma 25 (Restricted Upgrading). *If $\Gamma_1, x :^{m} A, \Gamma_2 \vdash b :^{\ell} B$ and $\ell_0 \sqsubseteq \ell$, then $\Gamma_1, x :^{\ell_0 \sqcup m} A, \Gamma_2 \vdash b :^{\ell} B$.*

Proof. By Lemma 21, $\ell_0 \sqcup \Gamma_1, x :^{\ell_0 \sqcup m} A, \ell_0 \sqcup \Gamma_2 \vdash b :^{\ell_0 \sqcup \ell} B$.
Since $\ell_0 \sqsubseteq \ell$, $\ell_0 \sqcup \Gamma_1, x :^{\ell_0 \sqcup m} A, \ell_0 \sqcup \Gamma_2 \vdash b :^{\ell} B$.
By rule ST-SubLD, $\Gamma_1, x :^{\ell_0 \sqcup m} A, \Gamma_2 \vdash b :^{\ell} B$ because $\Gamma_1 \sqsubseteq \ell_0 \sqcup \Gamma_1$ and $\Gamma_2 \sqsubseteq \ell_0 \sqcup \Gamma_2$.

Theorem 17 (Preservation (Theorem 3)). *If $\Gamma \vdash a :^{\ell} A$ and $\vdash a \rightsquigarrow a'$, then $\Gamma \vdash a' :^{\ell} A$.*

Proof. By induction on $\Gamma \vdash a :^{\ell} A$ and inversion on $\vdash a \rightsquigarrow a'$.

- Rule ST-AppD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash b\ a^m :^{\ell} B$ where $\Gamma_1 \vdash b :^{\ell}\ ^m A \to B$ and $\Gamma_2 \vdash a :^{\ell \sqcup m} A$.
  Let $\vdash b\ a^m \rightsquigarrow c$. By inversion:
  - $\vdash b\ a^m \rightsquigarrow b'\ a^m$ when $\vdash b \rightsquigarrow b'$.
    Need to show: $\Gamma_1 \sqcap \Gamma_2 \vdash b'\ a^m :^{\ell} B$.
    Follows by IH and rule ST-AppD.
  - $b = \lambda^m x : A'.b'$ and $\vdash b\ a^m \rightsquigarrow b'\{a/x\}$.
    Need to show: $\Gamma_1 \sqcap \Gamma_2 \vdash b'\{a/x\} :^{\ell} B$.
    By inversion on $\Gamma_1 \vdash \lambda^m x : A'.b' :^{\ell}\ ^m A \to B$, we get $A' = A$ and $\Gamma_1, x :^{\ell_0 \sqcup m} A \vdash b' :^{\ell_0} B$ for some $\ell_0 \sqsubseteq \ell$.
    By rule ST-SubRD, $\Gamma_1, x :^{\ell_0 \sqcup m} A \vdash b' :^{\ell} B$.
    By lemma 25, $\Gamma_1, x :^{\ell \sqcup m} A \vdash b' :^{\ell} B$.
    This case, then, follows by the substitution lemma.
- Rule ST-LetPairD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{let}\ (x^m, y) = a\ \mathbf{in}\ b :^{\ell} B$ where $\Gamma_1 \vdash a :^{\ell}\ ^m A_1 \times A_2$ and $\Gamma_2, x :^{\ell \sqcup m} A_1, y :^{\ell} A_2 \vdash b :^{\ell} B$.
  Let $\vdash \mathbf{let}\ (x^m, y) = a\ \mathbf{in}\ b \rightsquigarrow c$. By inversion:
  - $\vdash \mathbf{let}\ (x^m, y) = a\ \mathbf{in}\ b \rightsquigarrow \mathbf{let}\ (x^m, y) = a'\ \mathbf{in}\ b$ when $\vdash a \rightsquigarrow a'$.
    Need to show: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{let}\ (x^m, y) = a'\ \mathbf{in}\ b :^{\ell} B$.
    Follows by IH and rule ST-LetPairD.
  - $\vdash \mathbf{let}\ (x^m, y) = (a_1{}^m, a_2)\ \mathbf{in}\ b \rightsquigarrow b\{a_1/x\}\{a_2/y\}$.
    Need to show: $\Gamma_1 \sqcap \Gamma_2 \vdash b\{a_1/x\}\{a_2/y\} :^{\ell} B$.
    By inversion on $\Gamma_1 \vdash (a_1{}^m, a_2) :^{\ell}\ ^m A_1 \times A_2$, we have:
    $\exists \Gamma_{11}, \Gamma_{12}$ such that $\Gamma_{11} \vdash a_1 :^{\ell \sqcup m} A_1$ and $\Gamma_{12} \vdash a_2 :^{\ell} A_2$ and $\Gamma_1 = \Gamma_{11} \sqcap \Gamma_{12}$.
    This case, then, follows by applying the substitution lemma twice.
- Rule ST-LetUnitD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{let\ unit} = a\ \mathbf{in}\ b :^{\ell} B$ where $\Gamma_1 \vdash a :^{\ell}$ **Unit** and $\Gamma_2 \vdash b :^{\ell} B$.
  Let $\vdash \mathbf{let\ unit} = a\ \mathbf{in}\ b \rightsquigarrow c$. By inversion:
  - $\vdash \mathbf{let\ unit} = a\ \mathbf{in}\ b \rightsquigarrow \mathbf{let\ unit} = a'\ \mathbf{in}\ b$ when $\vdash a \rightsquigarrow a'$.
    Need to show: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{let\ unit} = a'\ \mathbf{in}\ b :^{\ell} B$.
    Follows by IH and rule ST-LetUnitD.
  - $\vdash \mathbf{let\ unit} = \mathbf{unit}\ \mathbf{in}\ b \rightsquigarrow b$.
    Need to show: $\Gamma_1 \sqcap \Gamma_2 \vdash b :^{\ell} B$.
    Follows by rule ST-SubLD.
- Rule ST-Proj1D. Have: $\Gamma \vdash \mathbf{proj}_1\ a :^{\ell} A_1$ where $\Gamma \vdash a :^{\ell} A_1 \,\&\, A_2$.
  Let $\vdash \mathbf{proj}_1\ a \rightsquigarrow c$. By inversion:

- $\vdash \mathbf{proj}_1\ a \rightsquigarrow \mathbf{proj}_1\ a'$ when $\vdash a \rightsquigarrow a'$.
  Need to show: $\Gamma \vdash \mathbf{proj}_1\ a' :^\ell A_1$.
  Follows by IH and rule ST-Proj1D.
- $\vdash \mathbf{proj}_1\ (a_1; a_2) \rightsquigarrow a_1$.
  Need to show: $\Gamma \vdash a_1 :^\ell A_1$.
  Follows by inversion on $\Gamma \vdash (a_1; a_2) :^\ell A_1 \,\&\, A_2$.

– Rule ST-Proj2D. Similar to rule ST-Proj1D.
– Rule ST-CaseD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{case}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^\ell B$ where $\Gamma_1 \vdash a :^\ell A_1 + A_2$ and $\Gamma_2, x_1 :^\ell A_1 \vdash b_1 :^\ell B$ and $\Gamma_2, x_2 :^\ell A_2 \vdash b_2 :^\ell B$.
  Let $\vdash \mathbf{case}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow c$. By inversion:
  - $\vdash \mathbf{case}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow \mathbf{case}\ a'\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2$ when $\vdash a \rightsquigarrow a'$.
    Need to show: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{case}\ a'\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B$.
    Follows by IH and rule ST-CaseD.
  - $\vdash \mathbf{case}\ (\mathbf{inj}_1\ a_1)\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_1\{a_1/x_1\}$.
    Need to show $\Gamma_1 \sqcap \Gamma_2 \vdash b_1\{a_1/x_1\} :^q B$.
    By inversion on $\Gamma_1 \vdash \mathbf{inj}_1\ a_1 :^\ell A_1 + A_2$, we have: $\Gamma_1 \vdash a_1 :^\ell A_1$.
    This case, then, follows by applying the substitution lemma.
  - $\vdash \mathbf{case}\ (\mathbf{inj}_2\ a_2)\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_2\{a_2/x_2\}$.
    Similar to the previous case.

– Rules ST-SubLD and ST-SubRD. Follows by IH.

Theorem 18 (Progress (Theorem 4)). *If $\emptyset \vdash a :^\ell A$, then either a is a value or there exists a' such that $\vdash a \rightsquigarrow a'$.*

Proof. By induction on $\emptyset \vdash a :^\ell A$.

– Rule ST-VarD. Does not apply since the context here is empty.
– Rule ST-AppD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash b\ a^m :^\ell B$ where $\Gamma_1 \vdash b :^{\ell\ m} A \to B$ and $\Gamma_2 \vdash a :^{\ell \sqcup m} A$.
  Need to show: $\exists c, \vdash b\ a^m \rightsquigarrow c$.
  By IH, b is either a value or $\vdash b \rightsquigarrow b'$.
  If b is a value, then $b = \lambda^m x : A.b'$ for some b'. Therefore, $\vdash b\, a^m \rightsquigarrow b'\{a/x\}$.
  Otherwise, $\vdash b\ a^m \rightsquigarrow b'\ a^m$.
– Rule ST-LetPairD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{let}\ (x^m, y) = a\ \mathbf{in}\ b :^\ell B$ where $\Gamma_1 \vdash a :^{\ell\ m} A_1 \times A_2$ and $\Gamma_2, x :^{\ell \sqcup m} A_1, y :^\ell A_2 \vdash b :^\ell B$.
  Need to show: $\exists c, \vdash \mathbf{let}\ (x^m, y) = a\ \mathbf{in}\ b \rightsquigarrow c$.
  By IH, a is either a value or $\vdash a \rightsquigarrow a'$.
  If a is a value, then $a = ({a_1}^m, a_2)$. Therefore, $\vdash \mathbf{let}\ (x^m, y) = a\ \mathbf{in}\ b \rightsquigarrow b\{a_1/x\}\{a_2/y\}$.
  Otherwise, $\vdash \mathbf{let}\ (x^m, y) = a\ \mathbf{in}\ b \rightsquigarrow \mathbf{let}\ (x^m, y) = a'\ \mathbf{in}\ b$.
– Rule ST-LetUnitD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{let\ unit} = a\ \mathbf{in}\ b :^\ell B$ where $\Gamma_1 \vdash a :^\ell \mathbf{Unit}$ and $\Gamma_2 \vdash b :^\ell B$.
  Need to show: $\exists c, \vdash \mathbf{let\ unit} = a\ \mathbf{in}\ b \rightsquigarrow c$.
  By IH, a is either a value or $\vdash a \rightsquigarrow a'$.
  If a is a value, then $a = \mathbf{unit}$. Therefore, $\vdash \mathbf{let\ unit} = a\ \mathbf{in}\ b \rightsquigarrow b$.
  Otherwise, $\vdash \mathbf{let\ unit} = a\ \mathbf{in}\ b \rightsquigarrow \mathbf{let\ unit} = a'\ \mathbf{in}\ b$.

- Rule ST-Proj1D. Have: $\Gamma \vdash \mathbf{proj}_1\ a :^\ell A_1$ where $\Gamma \vdash a :^\ell A_1 \,\&\, A_2$.
  Need to show: $\exists c, \vdash \mathbf{proj}_1\ a \rightsquigarrow c$.
  By IH, a is either a value or $\vdash a \rightsquigarrow a'$.
  If a is a value, then $a = (a_1; a_2)$. Therefore, $\vdash \mathbf{proj}_1\ a \rightsquigarrow a_1$.
  Otherwise, $\vdash \mathbf{proj}_1\ a \rightsquigarrow \mathbf{proj}_1\ a'$.
- Rule ST-Proj2D. Similar to rule ST-Proj1D.
- Rule ST-CaseD. Have: $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{case}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^\ell B$ where $\Gamma_1 \vdash a :^\ell A_1 + A_2$ and $\Gamma_2, x_1 :^\ell A_1 \vdash b_1 :^\ell B$ and $\Gamma_2, x_2 :^\ell A_2 \vdash b_2 :^\ell B$.
  Need to show: $\exists c, \vdash \mathbf{case}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow c$.
  By IH, a is either a value or $\vdash a \rightsquigarrow a'$.
  If a is a value, then $a = \mathbf{inj}_1\ a_1$ or $a = \mathbf{inj}_2\ a_2$.
  Then, $\vdash \mathbf{case}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_1\{a_1/x_1\}$ or $\vdash \mathbf{case}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_2\{a_2/x_2\}$.
  Otherwise, $\vdash \mathbf{case}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow \mathbf{case}\ a'\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2$.
- Rules ST-SubLD and ST-SubRD. Follows by IH.
- Rules ST-LamD, ST-WPairD, ST-UnitD, ST-SPairD, ST-Inj1D, and ST-Inj2D. The terms typed by these rules are values.

D Heap Semantics For Simply-Typed LDC

Lemma 26 (Similarity). *If $[H]a \Longrightarrow_S^q [H']a'$, then $a\{H\} = a'\{H'\}$ or $\vdash a\{H\} \rightsquigarrow a'\{H'\}$. Here, $a\{H\}$ denotes the term obtained by substituting in a the definitions in H, in reverse order.*

Proof. By induction on $[H]a \Longrightarrow_S^q [H']a'$.

Lemma 27 (Unchanged (Lemma 10)). *If $[H_1, x \overset{r}{\mapsto} a, H_2]c \Longrightarrow_S^q [H_1', x \overset{r'}{\mapsto} a, H_2']c'$ (where $|H_1| = |H_1'|$) and $\neg(\exists q_0, r = q + q_0)$, then $r' = r$.*

Proof. By induction on $[H_1, x \overset{r}{\mapsto} a, H_2]c \Longrightarrow_S^q [H_1', x \overset{r'}{\mapsto} a, H_2']c'$.

Lemma 28 (Irrelevant (Lemma 11)). *If $[H_1, x \overset{r}{\mapsto} a, H_2]c \Longrightarrow_{S \cup fv\, b}^q [H_1', x \overset{r'}{\mapsto} a, H_2']c'$ (where $|H_1| = |H_1'|$) and $\neg(\exists q_0, r = q + q_0)$, then $[H_1, x \overset{r}{\mapsto} b, H_2]c \Longrightarrow_{S \cup fv\, a}^q [H_1', x \overset{r'}{\mapsto} b, H_2']c'$.*

Proof. By induction on $[H_1, x \overset{r}{\mapsto} a, H_2]c \Longrightarrow_{S \cup fv\, b}^q [H_1', x \overset{r'}{\mapsto} a, H_2']c'$.

Lemma 29. *If $H \models \Gamma_1 + \Gamma_2$ and $\Gamma_2 \vdash a :^q A$ and $q \neq 0$, then either a is a value or there exists H', Γ', a' such that:*

- $[H]a \Longrightarrow_S^q [H']a'$
- $H' \models \Gamma_1 + \Gamma_2'$
- $\Gamma_2' \vdash a' :^q A$

(Note that $+$ is overloaded here: $\Gamma_1 + \Gamma_2$ denotes addition of contexts Γ_1 and Γ_2 after padding them as necessary.)

Proof. By induction on $\Gamma_2 \vdash a :^q A$.

- Rule ST-Var. Have $0 \cdot \Gamma_{21}, x :^{q_2} A, 0 \cdot \Gamma_{22} \vdash x :^{q_2} A$.
  Further, $H \models (\Gamma_{11}, x :^{q_1} A, \Gamma_{12}) + (0 \cdot \Gamma_{21}, x :^{q_2} A, 0 \cdot \Gamma_{22})$.
  By inversion, $\exists H_1, q, a, H_2, \Gamma_0$ and q_0 such that $H = H_1, x \overset{q}{\mapsto} a, H_2$ and $\Gamma_0 \vdash a :^q A$ and $q = q_1 + q_2 + q_0$.
  By lemma 18, $\exists \Gamma_{01}, \Gamma_{02}$ such that $\Gamma_{01} \vdash a :^{q_0+q_1} A$ and $\Gamma_{02} \vdash a :^{q_2} A$ and $\Gamma_0 = \Gamma_{01} + \Gamma_{02}$.
  Then, we have,
  - $[H_1, x \overset{q_0+q_1+q_2}{\mapsto} a, H_2]x \Longrightarrow_S^{q_2} [H_1, x \overset{q_0+q_1}{\mapsto} a, H_2]a$.
  - $H_1, x \overset{q_0+q_1}{\mapsto} a, H_2 \models (\Gamma_{11}, x :^{q_1} A, \Gamma_{12}) + (\Gamma_{02}, x :^0 A, 0 \cdot \Gamma_{22})$.
  - $\Gamma_{02}, x :^0 A, 0 \cdot \Gamma_{22} \vdash a :^{q_2} A$.
- Rule ST-App. Have: $\Gamma_{21} + \Gamma_{22} \vdash b\ a^r :^q B$ where $\Gamma_{21} \vdash b :^q {}^rA \to B$ and $\Gamma_{22} \vdash a :^{q \cdot r} A$.
  Further, $H \models \Gamma_1 + (\Gamma_{21} + \Gamma_{22})$.
  By IH, if b steps, then $[H]b \Longrightarrow_{S \cup fv\, a}^q [H']b'$ and $\Gamma'_{21} \vdash b' :^q {}^rA \to B$ and $H' \models \Gamma_1 + (\Gamma'_{21} + \Gamma_{22})$.
  Then, $[H]b\ a^r \Longrightarrow_S^q [H']b'\ a^r$ and $\Gamma'_{21} + \Gamma_{22} \vdash b'\ a^r :^q B$ and $H' \models \Gamma_1 + (\Gamma'_{21} + \Gamma_{22})$.

  Otherwise, b is a value. By inversion $b = \lambda^r x : A.b'$.
  Also, by inversion, $\Gamma_{21}, x :^{q \cdot r} A \vdash b' :^q B$.
  Then, $[H](\lambda^r x : A.b')\ a \Longrightarrow_S^q [H, x \overset{q \cdot r}{\mapsto} a]b'$ (assuming x *fresh*).
  Further, $H, x \overset{q \cdot r}{\mapsto} a \models (\Gamma_1 + \Gamma_{21}), x :^{q \cdot r} A$.
- Rule ST-LetPair. Have: $\Gamma_{21} + \Gamma_{22} \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^q B$ where $\Gamma_{21} \vdash a :^{q \cdot q_0} {}^rA_1 \times A_2$ and $\Gamma_{22}, x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2 \vdash b :^q B$.
  Further, $H \models \Gamma_1 + (\Gamma_{21} + \Gamma_{22})$.
  By IH, if a steps, then $[H]a \Longrightarrow_{S \cup fv\, b}^{q \cdot q_0} [H']a'$ and $H' \models \Gamma_1 + (\Gamma'_{21} + \Gamma_{22})$ and $\Gamma'_{21} \vdash a' :^{q \cdot q_0} {}^rA_1 \times A_2$.
  So, $[H]\mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \Longrightarrow_S^{q \cdot q_0} [H']\mathbf{let}_{q_0}\ (x^r, y) = a'\ \mathbf{in}\ b$.
  By rule HeapStep-Discard, $[H]\mathbf{let}_{q_0}(x^r, y) = a\mathbf{in}b \Longrightarrow_S^q [H']\mathbf{let}_{q_0}(x^r, y) = a'\ \mathbf{in}\ b$. ($\because q_0 <: 1$)
  And by rule ST-LetPair, $\Gamma'_{21} + \Gamma_{22} \vdash \mathbf{let}_{q_0}\ (x^r, y) = a'\ \mathbf{in}\ b :\ B$.

  Otherwise, a is a value. By inversion, $a = ({a_1}^r, a_2)$.
  Further, $\exists \Gamma_{211}, \Gamma_{212}$ such that $\Gamma_{211} \vdash a_1 :^{q \cdot q_0 \cdot r} A_1$ and $\Gamma_{212} \vdash a_2 :^{q \cdot q_0} A_2$ and $\Gamma_{21} = \Gamma_{211} + \Gamma_{212}$.
  Now, $[H]\mathbf{let}_{q_0}\ (x^r, y) = ({a_1}^r, a_2)\ \mathbf{in}\ b \Longrightarrow_S^q [H, x \overset{q \cdot q_0 \cdot r}{\mapsto} a_1, y \overset{q \cdot q_0}{\mapsto} a_2]b$ (assuming x, y *fresh*).
  And, $H, x \overset{q \cdot q_0 \cdot r}{\mapsto} a_1, y \overset{q \cdot q_0}{\mapsto} a_2 \models (\Gamma_1 + \Gamma_{22}), x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2$.
- Rule ST-Case. Have: $\Gamma_{21} + \Gamma_{22} \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B$ where $\Gamma_{21} \vdash a :^{q \cdot q_0} A_1 + A_2$ and $\Gamma_{22}, x_1 :^{q \cdot q_0} A_1 \vdash b_1 :^q B$ and $\Gamma_{22}, x_2 :^{q \cdot q_0} A_2 \vdash b_2 :^q B$.
  Further, $H \models \Gamma_1 + (\Gamma_{21} + \Gamma_{22})$.
  By IH, if a steps, then $[H]a \Longrightarrow_{S \cup fv\, b_1 \cup fv\, b_2}^{q \cdot q_0} [H']a'$ and $H' \models \Gamma_1 + (\Gamma'_{21} + \Gamma_{22})$ and $\Gamma'_{21} \vdash a' :^{q \cdot q_0} {}^rA_1 \times A_2$.

So, $[H]\mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \Longrightarrow_S^{q\cdot q_0} [H']\mathbf{case}_{q_0}\ a'\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2$.
By rule HeapStep-Discard, $[H]\mathbf{case}_{q_0}\, a \mathbf{of} x_1.b_1; x_2.b_2 \Longrightarrow_S^{q} [H']\mathbf{case}_{q_0}\, a' \mathbf{of} x_1.b_1; x_2.b_2$.
$(\because q_0 <: 1)$
And by rule ST-Case, $\Gamma'_{21} + \Gamma_{22} \vdash \mathbf{case}_{q_0}\ a'\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B$.

Otherwise, a is a value. By inversion, $a = \mathbf{inj}_1\ a_1$ or $a = \mathbf{inj}_2\ a_2$.
Say $a = \mathbf{inj}_1\ a_1$. Then, $\Gamma_{21} \vdash a_1 :^{q\cdot q_0} A_1$.
Now, $[H]\mathbf{case}_{q_0}\ (\mathbf{inj}_1\ a_1)\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \Longrightarrow_S^{q} [H, x_1 \overset{q\cdot q_0}{\mapsto} a_1]b_1$ (assuming x_1 *fresh*).
And, $H, x_1 \overset{q\cdot q_0}{\mapsto} a_1 \models (\Gamma_1 + \Gamma_{22}), x_1 :^{q\cdot q_0} A_1$.
The case when $a = \mathbf{inj}_2\ a_2$ is similar.

Theorem 19 (Soundness (Theorem 5)). *If $H \models \Gamma$ and $\Gamma \vdash a :^q A$ and $q \neq 0$, then either a is a value or there exists H', Γ', a' such that $[H]a \Longrightarrow_S^q [H']a'$ and $H' \models \Gamma'$ and $\Gamma' \vdash a' :^q A$.*

Proof. Use lemma 29 with $\Gamma_1 := 0 \cdot \Gamma$ and $\Gamma_2 := \Gamma$.

Lemma 30. *If $H \models \Gamma_1 \sqcap \Gamma_2$ and $\Gamma_2 \vdash a :^\ell A$ and $l \neq \top$, then either a is a value or there exists H', Γ', a' such that:*

- $[H]a \Longrightarrow_S^\ell [H']a'$
- $H' \models \Gamma_1 \sqcap \Gamma'_2$
- $\Gamma'_2 \vdash a' :^\ell A$

Proof. By induction on $\Gamma_2 \vdash a :^\ell A$. Follow lemma 29.

Theorem 20 (Soundness (Theorem 5)). *If $H \models \Gamma$ and $\Gamma \vdash a :^\ell A$ and $l \neq \top$, then either a is a value or there exists H', Γ', a' such that $[H]a \Longrightarrow_S^\ell [H']a'$ and $H' \models \Gamma'$ and $\Gamma' \vdash a' :^\ell A$.*

Proof. Use lemma 30 with $\Gamma_1 := 0 \sqcup \Gamma$ and $\Gamma_2 := \Gamma$.

Lemma 31 (No Use (Lemma 12)). *In LDC($\mathcal{Q}_\mathbb{N}$): Let $\emptyset \vdash f :^1 {}^0A \to A$. Then, for any $\emptyset \vdash a_1 :^0 A$ and $\emptyset \vdash a_2 :^0 A$, the terms $f\ {a_1}^0$ and $f\ {a_2}^0$ have the same operational behaviour.*

Proof. To see why, we consider the reduction of $f\ {a_1}^0$ and $f\ {a_2}^0$.

Let $[H]a \Longrightarrow_{q,j}^* [H']a'$ denote a reduction of j steps where H and a are the initial heap and term; H' and a' are the final heap and term; and q is the quantity or level of the reduction.

For some j, H and b, we have, $[\emptyset]f\,{a_1}^0 \Longrightarrow_{1,j}^* [H](\lambda^0 x.b)\,{a_1}^0$ and $[\emptyset]f\,{a_2}^0 \Longrightarrow_{1,j}^* [H](\lambda^0 x.b)\ {a_2}^0$.

Then, $[H](\lambda^0 x.b)\ {a_1}^0 \Longrightarrow_S^1 [H, x \overset{0}{\mapsto} a_1]b$ and $[H](\lambda^0 x.b)\ {a_2}^0 \Longrightarrow_S^1 [H, x \overset{0}{\mapsto} a_2]b$ (for x *fresh*).

But, if $[H, x \overset{0}{\mapsto} a_1]b \Longrightarrow_{1,k}^* [H', x \overset{0}{\mapsto} a_1, H'']b'$, then $[H, x \overset{0}{\mapsto} a_2]b \Longrightarrow_{1,k}^* [H', x \overset{0}{\mapsto} a_2, H'']b'$, for any k (By Lemma 11).

Therefore, $f\ {a_1}^0$ and $f\ {a_2}^0$ have the same operational behavior.

Lemma 32 (Single Use (Lemma 13)). *In LDC($\mathbb{N}_{\geq}$): Let $\emptyset \vdash f :^1 {}^1A \to A$. Then, for any $\emptyset \vdash a :^1 A$, the term $f\ a^1$ uses a at most once during reduction.*

Proof. To see why, consider the reduction of $f\ a^1$.

For some j, H and b, we have, $[\emptyset]f\ a^1 \Longrightarrow^*_{1,j} [H](\lambda^1 x.b)\ a^1$.

Then, $[H](\lambda^1 x.b)\ a^1 \Longrightarrow^1_S [H, x \overset{1}{\mapsto} a]b$ (for x *fresh*).

Now, b may reduce to a value without ever looking-up the value of x or b may look-up the value of x exactly once. A single use of x will change its allowed usage from 1 to 0, making it essentially unusable thereafter. Hence, a cannot be used more than once.

E Linearity Analysis in PTS Version of LDC

Lemma 33 (Multiplication). *If $\Gamma \vdash a :^q A$, then $r_0 \cdot \Gamma \vdash a :^{r_0 \cdot q} A$.*

Proof. By induction on $\Gamma \vdash a :^q A$. Follow the proof of lemma 16.

Lemma 34 (Factorization). *If $\Gamma \vdash a :^q A$ and $q \neq 0$, then there exists Γ' such that $\Gamma' \vdash a :^1 A$ and $\Gamma <: q \cdot \Gamma'$.*

Proof. By induction on $\Gamma \vdash a :^q A$. Follow the proof of lemma 17.

Lemma 35 (Splitting). *If $\Gamma \vdash a :^{q_1+q_2} A$, then there exists Γ_1 and Γ_2 such that $\Gamma_1 \vdash a :^{q_1} A$ and $\Gamma_2 \vdash a :^{q_2} A$ and $\Gamma = \Gamma_1 + \Gamma_2$.*

Proof. If $q_1 + q_2 = 0$, then $\Gamma_1 := 0 \cdot \Gamma$ and $\Gamma_2 := \Gamma$.
Otherwise, by Lemma 34, $\exists \Gamma'$ such that $\Gamma' \vdash a :^1 A$ and $\Gamma <: (q_1 + q_2) \cdot \Gamma'$.
Then, $\Gamma = (q_1 + q_2) \cdot \Gamma' + \Gamma_0$ for some Γ_0.
Now, by Lemma 33, $q_1 \cdot \Gamma' \vdash a :^{q_1} A$ and $q_2 \cdot \Gamma' \vdash a :^{q_2} A$.
By rule PTS-SubL, $q_2 \cdot \Gamma' + \Gamma_0 \vdash a :^{q_2} A$.
The lemma follows by setting $\Gamma_1 := q_1 \cdot \Gamma'$ and $\Gamma_2 := q_2 \cdot \Gamma' + \Gamma_0$.

Lemma 36 (Weakening (Lemma 14)). *If $\Gamma_1, \Gamma_2 \vdash a :^q A$ and $\Delta_1 \vdash_0 C : s$ and $\lfloor \Gamma_1 \rfloor = \Delta_1$, then $\Gamma_1, z :^0 C, \Gamma_2 \vdash a :^q A$.*

Proof. By induction on $\Gamma_1, \Gamma_2 \vdash a :^q A$.

Lemma 37 (Substitution (Lemma 15)). *If $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$ and $\Gamma \vdash c :^{r_0} C$ and $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$, then $\Gamma_1 + \Gamma, \Gamma_2\{c/z\} \vdash a\{c/z\} :^q A\{c/z\}$.*

Proof. By induction on $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$.

- Rule PTS-Var. Have: $0 \cdot \Gamma_1, x :^q A \vdash x :^q A$ where $\Delta \vdash_0 A : s$ and $\lfloor \Gamma_1 \rfloor = \Delta$. There are two cases to consider.
  - Have: $0 \cdot \Gamma_1, x :^q A \vdash x :^q A$ and $\Gamma \vdash a :^q A$ where $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$.
    Need to show: $\Gamma \vdash a :^q A$. Follows from what's given.

- Have: $0 \cdot \Gamma_{11}, z :^0 C, 0 \cdot \Gamma_{12}, x :^q A \vdash x :^q A$ and $\Gamma \vdash c :^0 C$ where $\lfloor \Gamma_{11} \rfloor = \lfloor \Gamma \rfloor$.
  Need to show: $\Gamma, 0 \cdot \Gamma_{12}\{c/z\}, x :^q A\{c/z\} \vdash x :^q A\{c/z\}$.
  Follows by rules PTS-Var and PTS-SubL.

– Rule PTS-Weak. Have: $\Gamma_1, y :^0 B \vdash a :^q A$ where $\Gamma_1 \vdash a :^q A$ and $\Delta \vdash_0 B : s$ and $\lfloor \Gamma_1 \rfloor = \Delta$. There are two cases to consider.
  - Have: $\Gamma_1, y :^0 B \vdash a :^q A$ and $\Gamma \vdash b :^0 B$ where $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$.
    Need to show: $\Gamma_1 + \Gamma \vdash a\{b/y\} :^q A\{b/y\}$.
    Since $y \notin$ fv a and $y \notin$ fv A, need to show: $\Gamma_1 + \Gamma \vdash a :^q A$.
    This case follows by rule PTS-SubL.
  - Have: $\Gamma_{11}, z :^{r_0} C, \Gamma_{12}, y :^0 B \vdash a :^q A$ and $\Gamma \vdash c :^{r_0} C$ where $\lfloor \Gamma_{11} \rfloor = \lfloor \Gamma \rfloor$.
    Need to show: $\Gamma_{11} + \Gamma, \Gamma_{12}\{c/z\}, y :^0 B\{c/z\} \vdash a\{c/z\} :^q A\{c/z\}$.
    Follows by IH and rule PTS-Weak.

– Rule PTS-Pi. Have: $\Gamma_{11} + \Gamma_{21}, z :^{r_{01}+r_{02}} C, \Gamma_{12} + \Gamma_{22} \vdash \Pi x :^r A.B :^q s_3$ where $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{12} \vdash A :^q s_1$ and $\Gamma_{21}, z :^{r_{02}} C, \Gamma_{22}, x :^{q_0} A \vdash B :^q s_2$ and $\mathcal{R}(s_1, s_2, s_3)$.
  Further, $\Gamma \vdash c :^{r_{01}+r_{02}} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_{11} \rfloor$.
  By lemma 35, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{r_{01}} C$ and $\Gamma_{32} \vdash c :^{r_{02}} C$ and $\Gamma_{31} + \Gamma_{32} = \Gamma$.
  By IH, $\Gamma_{11} + \Gamma_{31}, \Gamma_{12}\{c/z\} \vdash A\{c/z\} :^q s_1$ and $\Gamma_{21} + \Gamma_{32}, \Gamma_{22}\{c/z\}, x :^{q_0} A\{c/z\} \vdash B\{c/z\} :^q s_2$.
  This case, then, follows by rule PTS-Pi.

– Rule PTS-Lam. Have: $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash \lambda^r x : A.b :^q \Pi x :^r A.B$ where $\Gamma_1, z :^{r_0} C, \Gamma_2, x :^{q \cdot r} A \vdash b :^q B$.
  Further, $\Gamma \vdash c :^{r_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
  Need to show: $\Gamma_1 + \Gamma, \Gamma_2\{c/z\} \vdash \lambda^r x : A\{c/z\}.b\{c/z\} :^q \Pi x :^r A\{c/z\}.B\{c/z\}$.
  By IH, $\Gamma_1 + \Gamma, \Gamma_2\{c/z\}, x :^{q \cdot r} A\{c/z\} \vdash b\{c/z\} :^q B\{c/z\}$.
  This case, then, follows by rule PTS-Lam.

– Rule PTS-App. Have: $\Gamma_{11} + \Gamma_{21}, z :^{r_{01}+r_{02}} C, \Gamma_{12} + \Gamma_{22} \vdash b\ a^r :^q B\{a/x\}$ where
  $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{12} \vdash b :^q \Pi x :^r A.B$ and $\Gamma_{21}, z :^{r_{02}} C, \Gamma_{22} \vdash a :^{q \cdot r} A$.
  Further, $\Gamma \vdash c :^{r_{01}+r_{02}} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_{11} \rfloor$.
  By lemma 35, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{r_{01}} C$ and $\Gamma_{32} \vdash c :^{r_{02}} C$ and $\Gamma_{31} + \Gamma_{32} = \Gamma$.
  By IH, $\Gamma_{11} + \Gamma_{31}, \Gamma_{12}\{c/z\} \vdash b\{c/z\} :^q \Pi x :^r A\{c/z\}.B\{c/z\}$
  and $\Gamma_{21} + \Gamma_{32}, \Gamma_{22}\{c/z\} \vdash a\{c/z\} :^{q \cdot r} A\{c/z\}$.
  This case, then, follows by rule PTS-App.

– Rule PTS-Conv. Have: $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q B$ where $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$ and $A =_\beta B$.
  Further, $\Gamma \vdash c :^{r_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
  Need to show: $\Gamma_{11} + \Gamma, \Gamma_2\{c/z\} \vdash a\{c/z\} :^q B\{c/z\}$.
  By IH, $\Gamma_{11} + \Gamma, \Gamma_2\{c/z\} \vdash a\{c/z\} :^q A\{c/z\}$.
  Also, since $A =_\beta B$, so $A\{c/z\} =_\beta B\{c/z\}$.
  This case, then, follows by rule PTS-Conv.

– Rules PTS-SubL and PTS-SubR. By IH.

– Rule PTS-WPair. Have: $\Gamma_{11} + \Gamma_{21}, z :^{r_{01}+r_{02}} C, \Gamma_{12} + \Gamma_{22} \vdash (a_1{}^r, a_2) :^q \Sigma x :^r A_1.A_2$ where $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{12} \vdash a_1 :^{q\cdot r} A_1$ and $\Gamma_{21}, z :^{r_{02}} C, \Gamma_{22} \vdash a_2 :^q A_2\{a_1/x\}$.
Further, $\Gamma \vdash c :^{r_{01}+r_{02}} C$ where $\lfloor\Gamma\rfloor = \lfloor\Gamma_{11}\rfloor$.
By lemma 35, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{r_{01}} C$ and $\Gamma_{32} \vdash c :^{r_{02}} C$ and $\Gamma = \Gamma_{31} + \Gamma_{32}$.
By IH, $\Gamma_{11} + \Gamma_{31}, \Gamma_{12}\{c/z\} \vdash a_1\{c/z\} :^{q\cdot r} A_1\{c/z\}$
and $\Gamma_{21} + \Gamma_{32}, \Gamma_{22}\{c/z\} \vdash a_2\{c/z\} :^q A_2\{a_1/x\}\{c/z\}$.
Now, $A_2\{a_1/x\}\{c/z\} = A_2\{c/z\}\{a_1\{c/z\}/x\}$.
This case, then, follows by rule PTS-WPair.
– Rule PTS-LetPair. Have: $\Gamma_{11}+\Gamma_{21}, z :^{r_{01}+r_{02}} C, \Gamma_{12}+\Gamma_{22} \vdash \mathbf{let}_{q_0}(x^r, y) = a \mathbf{\ in\ } b :^q B\{a/v\}$ where
$\Delta, v : \Sigma x :^r A_1.A_2 \vdash_0 B : s$ and $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{12} \vdash a :^{q\cdot q_0} \Sigma x :^r A_1.A_2$ and $\Delta = \lfloor\Gamma_{11}\rfloor, z : C, \lfloor\Gamma_{12}\rfloor$ and $\Gamma_{21}, z :^{r_{02}} C, \Gamma_{22}, x :^{q\cdot q_0\cdot r} A_1, y :^{q\cdot q_0} A_2 \vdash b :^q B\{(x^r, y)/v\}$.
Further, $\Gamma \vdash c :^{r_{01}+r_{02}} C$ where $\lfloor\Gamma\rfloor = \lfloor\Gamma_{11}\rfloor$.
By lemma 35, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{r_{01}} C$ and $\Gamma_{32} \vdash c :^{r_{02}} C$ and $\Gamma = \Gamma_{31} + \Gamma_{32}$.
By IH, $\Gamma_{11} + \Gamma_{31}, \Gamma_{12}\{c/z\} \vdash a\{c/z\} :^{q\cdot q_0} \Sigma x :^r A_1\{c/z\}.A_2\{c/z\}$ and $\Gamma_{21}+\Gamma_{32}, \Gamma_{22}\{c/z\}, x :^{q\cdot q_0\cdot r} A_1\{c/z\}, y :^{q\cdot q_0} A_2\{c/z\} \vdash b\{c/z\} :^q B\{(x^r, y)/v\}\{c/z\}$.
This case, then, follows by rule PTS-LetPair.
– Rules PTS-SPair, PTS-Proj1, and PTS-Proj2. By IH.
– Rules PTS-Sum, PTS-Inj1, and PTS-Inj2. By IH.
– Rule PTS-Case. Have: $\Gamma_{11}+\Gamma_{21}, z :^{r_{01}+r_{02}} C, \Gamma_{12}+\Gamma_{22} \vdash \mathbf{case}_{q_0} a \mathbf{\ of\ } x_1.b_1; x_2.b_2 :^q B\{a/v\}$ where
$\Delta, v : A_1 + A_2 \vdash_0 B : s$ and $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{12} \vdash a :^{q\cdot q_0} A_1 + A_2$ and $\Delta = \lfloor\Gamma_{11}\rfloor, z : C, \lfloor\Gamma_{12}\rfloor$ and
$\Gamma_{21}, z :^{r_{02}} C, \Gamma_{22}, x_1 :^{q\cdot q_0} A_1 \vdash b_1 :^q B\{\mathbf{inj}_1\ x_1/v\}$ and
$\Gamma_{21}, z :^{r_{02}} C, \Gamma_{22}, x_2 :^{q\cdot q_0} A_2 \vdash b_2 :^q B\{\mathbf{inj}_2\ x_2/v\}$.
Further, $\Gamma \vdash c :^{r_{01}+r_{02}} C$ where $\lfloor\Gamma\rfloor = \lfloor\Gamma_{11}\rfloor$.
By lemma 35, $\exists \Gamma_{31}, \Gamma_{32}$ such that $\Gamma_{31} \vdash c :^{r_{01}} C$ and $\Gamma_{32} \vdash c :^{r_{02}} C$ and $\Gamma = \Gamma_{31} + \Gamma_{32}$.
By IH, $\Gamma_{11} + \Gamma_{31}, \Gamma_{12}\{c/z\} \vdash a\{c/z\} :^{q\cdot q_0} A_1\{c/z\} + A_2\{c/z\}$ and
$\Gamma_{21} + \Gamma_{32}, \Gamma_{22}\{c/z\}, x_1 :^{q\cdot q_0} A_1\{c/z\} \vdash b_1\{c/z\} :^q B\{\mathbf{inj}_1\ x_1/v\}\{c/z\}$ and
$\Gamma_{21} + \Gamma_{32}, \Gamma_{22}\{c/z\}, x_2 :^{q\cdot q_0} A_2\{c/z\} \vdash b_2\{c/z\} :^q B\{\mathbf{inj}_2\ x_2/v\}\{c/z\}$.
This case, then, follows by rule PTS-Case.

Theorem 21 (Preservation (Theorem 6)). *If $\Gamma \vdash a :^q A$ and $\vdash a \rightsquigarrow a'$, then $\Gamma \vdash a' :^q A$.*

Proof. By induction on $\Gamma \vdash a :^q A$ and inversion on $\vdash a \rightsquigarrow a'$.

– Rule PTS-App. Have: $\Gamma_1 + \Gamma_2 \vdash b\ a^r :^q B\{a/x\}$ where $\Gamma_1 \vdash b :^q \Pi x :^r A.B$ and $\Gamma_2 \vdash a :^{q\cdot r} A$.
Let $\vdash b\ a^r \rightsquigarrow c$. By inversion:
  • $\vdash b\ a^r \rightsquigarrow b'\ a^r$ when $\vdash b \rightsquigarrow b'$.
  Need to show: $\Gamma_1 + \Gamma_2 \vdash b'\ a^r :^q B\{a/x\}$.
  Follows by IH and rule PTS-App.

- $b = \lambda^r x : A'.b'$ and $\vdash b\ a^r \rightsquigarrow b'\{a/x\}$.
  Need to show: $\Gamma_1 + \Gamma_2 \vdash b'\{a/x\} :^q B\{a/x\}$.
  By inversion on $\Gamma_1 \vdash \lambda^r x : A'.b' :^q \Pi x :^r A.B$, we get $\Gamma_1, x :^{q_0 \cdot r} A'' \vdash b' :^{q_0} B$ for some $q_0 <: q$ and $A'' =_\beta A$.
  Now, there are two cases to consider.
  - $q_0 = 0$. Since $q_0 <: q$, so $q = 0$.
    Then, by the substitution lemma, $\Gamma_1 + \Gamma_2 \vdash b'\{a/x\} :^0 B\{a/x\}$.
  - $q_0 \neq 0$. By lemma 34, $\exists \Gamma_1'$ and r' such that $\Gamma_1', x :^{r'} A'' \vdash b' :^1 B$ and $\Gamma_1 <: q_0 \cdot \Gamma_1'$ and $q_0 \cdot r <: q_0 \cdot r'$.
    Since $q_0 \neq 0$, therefore $r <: r'$. Hence, by rule PTS-SubL, $\Gamma_1', x :^r A'' \vdash b' :^1 B$.
    Now, by lemma 33, $q \cdot \Gamma_1', x :^{q \cdot r} A'' \vdash b' :^q B$. By rule PTS-SubL, $\Gamma_1, x :^{q \cdot r} A'' \vdash b' :^q B$.
    This case, then, follows by rule PTS-Conv and the substitution lemma.

– Rule PTS-LetPair. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^q B\{a/z\}$ where $\Delta, z : \Sigma x :^r A_1.A_2 \vdash_0 B : s$ and $\Gamma_1 \vdash a :^{q \cdot q_0} \Sigma x :^r A_1.A_2$ and $\Gamma_2, x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2 \vdash b :^q B\{(x^r, y)/z\}$.
  Let $\vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \rightsquigarrow c$. By inversion:
  - $\vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \rightsquigarrow \mathbf{let}_{q_0}\ (x^r, y) = a'\ \mathbf{in}\ b$ when $\vdash a \rightsquigarrow a'$.
    Need to show: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ (x^r, y) = a'\ \mathbf{in}\ b :^q B\{a/z\}$.
    Follows by IH and rules PTS-LetPair and PTS-Conv.
  - $\vdash \mathbf{let}_{q_0}\ (x^r, y) = ({a_1}^r, a_2)\ \mathbf{in}\ b \rightsquigarrow b\{a_1/x\}\{a_2/y\}$.
    Need to show: $\Gamma_1 + \Gamma_2 \vdash b\{a_1/x\}\{a_2/y\} :^q B\{({a_1}^r, a_2)/z\}$.
    By inversion on $\Gamma_1 \vdash ({a_1}^r, a_2) :^{q \cdot q_0} \Sigma x :^r A_1.A_2$, we have:
    $\exists \Gamma_{11}, \Gamma_{12}$ such that $\Gamma_{11} \vdash a_1 :^{q \cdot q_0 \cdot r} A_1$ and $\Gamma_{12} \vdash a_2 :^{q \cdot q_0} A_2\{a_1/x\}$ and $\Gamma_1 = \Gamma_{11} + \Gamma_{12}$.
    Applying the substitution lemma, we get $\Gamma_2 + \Gamma_{11}, y :^{q \cdot q_0} A_2\{a_1/x\} \vdash b\{a_1/x\} :^q B\{({a_1}^r, y)/z\}$.
    Applying the lemma again, we get $\Gamma_2 + \Gamma_{11} + \Gamma_{12} \vdash b\{a_1/x\}\{a_2/y\} :^q B\{({a_1}^r, a_2)/z\}$, as required.

– Rule PTS-Case. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B\{a/z\}$ where $\Delta, z : A_1 + A_2 \vdash_0 B : s$ and $\Gamma_1 \vdash a :^{q \cdot q_0} A_1 + A_2$ and $\Gamma_2, x_1 :^{q \cdot q_0} A_1 \vdash b_1 :^q B\{\mathbf{inj}_1\ x_1/z\}$ and $\Gamma_2, x_2 :^{q \cdot q_0} A_2 \vdash b_2 :^q B\{\mathbf{inj}_2\ x_2/z\}$.
  Let $\vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow c$. By inversion:
  - $\vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow \mathbf{case}_{q_0}\ a'\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2$ when $\vdash a \rightsquigarrow a'$.
    Need to show: $\Gamma_1 + \Gamma_2 \vdash \mathbf{case}_{q_0}\ a'\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B\{a/z\}$.
    Follows by IH and rules PTS-Case and PTS-Conv.
  - $\vdash \mathbf{case}_{q_0}\ (\mathbf{inj}_1\ a_1)\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_1\{a_1/x_1\}$.
    Need to show $\Gamma_1 + \Gamma_2 \vdash b_1\{a_1/x_1\} :^q B\{\mathbf{inj}_1\ a_1/z\}$.
    By inversion on $\Gamma_1 \vdash \mathbf{inj}_1\ a_1 :^{q \cdot q_0} A_1 + A_2$, we have $\Gamma_1 \vdash a_1 :^{q \cdot q_0} A_1$.
    This case, then, follows by applying the substitution lemma.
  - $\vdash \mathbf{case}_{q_0}\ (\mathbf{inj}_2\ a_2)\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_2\{a_2/x_2\}$.
    Similar to the previous case.

Theorem 22 (Progress (Theorem 7)). *If $\emptyset \vdash a :^q A$, then either a is a value or there exists a' such that $\vdash a \rightsquigarrow a'$.*

Proof. By induction on $\emptyset \vdash a :^q A$.

- Rule PTS-App. Have: $\Gamma_1 + \Gamma_2 \vdash b\ a^r :^q B\{a/x\}$ where $\Gamma_1 \vdash b :^q \Pi x :^r A.B$ and $\Gamma_2 \vdash a :^{q \cdot r} A$.
  Need to show: $\exists c, \vdash b\ a^r \rightsquigarrow c$.
  By IH, b is either a value or $\vdash b \rightsquigarrow b'$.
  If b is a value, then, by inversion, $b = \lambda^r x : A'.b'$ for some b'. Therefore, $\vdash b\ a^r \rightsquigarrow b'\{a/x\}$.
  Otherwise, $\vdash b\ a^r \rightsquigarrow b'\ a^r$.
- Rule PTS-LetPair. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^q B\{a/z\}$ where $\Delta, z : \Sigma x :^r A_1.A_2 \vdash_0 B : s$ and $\Gamma_1 \vdash a :^{q \cdot q_0} \Sigma x :^r A_1.A_2$ and $\Gamma_2, x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2 \vdash b :^q B\{(x^r, y)/z\}$.
  Need to show: $\exists c, \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \rightsquigarrow c$.
  By IH, a is either a value or $\vdash a \rightsquigarrow a'$.
  If a is a value, then, by inversion, $a = ({a_1}^r, a_2)$. Therefore, $\vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \rightsquigarrow b\{a_1/x\}\{a_2/y\}$.
  Otherwise, $\vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \rightsquigarrow \mathbf{let}_{q_0}\ (x^r, y) = a'\ \mathbf{in}\ b$.
- Rule ST-Case. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 :^q B\{a/z\}$ where $\Delta, z : A_1 + A_2 \vdash_0 B : s$ and $\Gamma_1 \vdash a :^{q \cdot q_0} A_1 + A_2$ and $\Gamma_2, x_1 :^{q \cdot q_0} A_1 \vdash b_1 :^q B\{\mathbf{inj}_1\ x_1/z\}$ and $\Gamma_2, x_2 :^{q \cdot q_0} A_2 \vdash b_2 :^q B\{\mathbf{inj}_2\ x_2/z\}$.
  Need to show: $\exists c, \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow c$.
  By IH, a is either a value or $\vdash a \rightsquigarrow a'$.
  If a is a value, then $a = \mathbf{inj}_1\ a_1$ or $a = \mathbf{inj}_2\ a_2$.
  Then, $\vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_1\{a_1/x_1\}$ or $\vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow b_2\{a_2/x_2\}$.
  Otherwise, $\vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2 \rightsquigarrow \mathbf{case}_{q_0}\ a'\ \mathbf{of}\ x_1.b_1\ ;\ x_2.b_2$.
- The remaining cases follow similarly.

F Heap Soundness for PTS Version of LDC

Recall that to show heap soundness, we need to allow delayed substitution in types. For this purpose, we extend contexts with definitions. These definitions are used to derive type equalities only and do not interact with the analyses. The typing rules for definitions and the conversion rule where the definitions are used are shown in Figure 11. We add these extra rules to the type system of the calculus.

There are a few things to note:

- Definitions do not interact with the analyses, as we see in rules PTS-DefVar and PTS-DefWeak.
- The conversion rule PTS-DefConv checks for equality of types after substituting the definitions in the types. Here, $A\{\Delta\}$ denotes the type A with the definitions in Δ substituted in reverse order. With this rule, we can show: $x = \mathbf{Unit} :^0 s \vdash \lambda^1 y : x.y :^1 \Pi y :^1 \mathbf{Unit}.\mathbf{Unit}$.

context, $\Gamma ::= \emptyset \mid \Gamma, x :^q A \mid \Gamma, x = a :^q A$

$\boxed{\Gamma \vdash a :^q A}$ *(Extra Rules)*

$$\textsc{PTS-DefVar}\ \frac{\Delta \vdash_0 a : A \qquad \lfloor \Gamma \rfloor = \Delta}{0 \cdot \Gamma, x = a :^q A \vdash x :^q A} \qquad \textsc{PTS-DefWeak}\ \frac{\Gamma \vdash a :^q A \qquad \Delta \vdash_0 b : B \qquad \lfloor \Gamma \rfloor = \Delta}{\Gamma, y = b :^0 B \vdash a :^q A} \qquad \textsc{PTS-DefConv}\ \frac{\Gamma \vdash a :^q A \qquad \Delta \vdash_0 B : s \qquad A\{\Delta\} =_\beta B\{\Delta\} \qquad \lfloor \Gamma \rfloor = \Delta}{\Gamma \vdash a :^q B}$$

Fig. 11. Typing Rules With Definitions

The definitions allow us to communicate to the type system the substitutions that have been delayed by the heap. To enable this communication, we need to update the compatibility relation, as shown in Figure 12.

$\boxed{H \models \Gamma}$ *(Updated Compatibility)*

$$\textsc{HeapCompat-DefEmpty}\ \frac{}{\emptyset \models \emptyset} \qquad \textsc{HeapCompat-DefCons}\ \frac{H \models \Gamma_1 + \Gamma_2 \qquad \Gamma_2 \vdash a :^q A}{H, x \overset{q}{\mapsto} a \models \Gamma_1, x = a :^q A}$$

Fig. 12. Compatibility Relation With Definitions

These extensions do not alter the essential character of the underlying system. The following lemma establish the correspondence between the underlying system and the extended system. To distinguish, let us denote the typing and the compatibility judgements of the underlying system as $\Gamma \vdash^{\mathbf{u}} a :^q A$ and $H \models^{\mathbf{u}} \Gamma$ and those of the extended system as $\Gamma \vdash^{\mathbf{e}} a :^q A$ and $H \models^{\mathbf{e}} \Gamma$ respectively. Now, for $H \models^{\mathbf{u}} \Gamma$, let Γ_H be the context Γ with the variables defined according to H. Then, the multi-substitution lemma below says that given a derivation in the extended system, substituting the definitions gives us a derivation in the underlying system. The elaboration lemma says that given a derivation in the underlying system, adding appropriate definitions to the context gives us a derivation in the extended system.

Lemma 38 (Multi-Substitution). *If $H \models^{\mathbf{e}} \Gamma$ and $\Gamma \vdash^{\mathbf{e}} a :^q A$, then $\emptyset \vdash^{\mathbf{u}} a\{\Delta\} :^q A\{\Delta\}$ where $\Delta = \lfloor \Gamma \rfloor$.*

Proof. By induction on $H \models^{\mathbf{e}} \Gamma$.

Lemma 39 (Elaboration). *If $H \models^{\mathbf{u}} \Gamma$ and $\Gamma \vdash^{\mathbf{u}} a :^q A$, then $H \models^{\mathbf{e}} \Gamma_H$ and $\Gamma_H \vdash^{\mathbf{e}} a :^q A$.*

Proof. By induction on $H \models^{\mathbf{u}} \Gamma$.

Lemma 40. *If $H \models \Gamma_1 + \Gamma_2$ and $\Gamma_2 \vdash a :^q A$ and $q \neq 0$, then either a is a value or there exists H', Γ', a' such that:*

- $[H]a \Longrightarrow_S^q [H']a'$
- $H' \models \Gamma_1 + \Gamma_2'$
- $\Gamma_2' \vdash a' :^q A$

Proof. By induction on $\Gamma_2 \vdash a :^q A$.

- Rule PTS-DefVar. Have: $0 \cdot \Gamma_{21}, x = a :^{q_2} A \vdash x :^{q_2} A$ where $\Delta \vdash_0 a : A$ and $\Delta = \lfloor \Gamma_{21} \rfloor$.
  Further, $H \models (\Gamma_{11}, x = a :^{q_1} A) + (0 \cdot \Gamma_{21}, x = a :^{q_2} A)$.
  By inversion, $\exists H_1, \Gamma_0$ such that $H = H_1, x \overset{q_1+q_2}{\mapsto} a$ and $\Gamma_0 \vdash a :^{q_1+q_2} A$.
  By lemma 35, $\exists \Gamma_{01}, \Gamma_{02}$ such that $\Gamma_{01} \vdash a :^{q_1} A$ and $\Gamma_{02} \vdash a :^{q_2} A$ and $\Gamma_0 = \Gamma_{01} + \Gamma_{02}$.
  Then, we have,
  - $[H_1, x \overset{q_1+q_2}{\mapsto} a]x \Longrightarrow_S^{q_2} [H_1, x \overset{q_1}{\mapsto} a]a$.
  - $H_1, x \overset{q_1}{\mapsto} a \models (\Gamma_{11}, x = a :^{q_1} A) + (\Gamma_{02}, x = a :^0 A)$.
  - $\Gamma_{02}, x = a :^0 A \vdash a :^{q_2} A$.
- Rule PTS-DefWeak. Have: $\Gamma_{21}, y = b :^0 B \vdash a :^q A$ where $\Gamma_{21} \vdash a :^q A$ and $\Delta \vdash_0 b : B$ and $\Delta = \lfloor \Gamma_{21} \rfloor$.
  Further, $H \models (\Gamma_{11}, y = b :^{q_1} B) + (\Gamma_{21}, y = b :^0 B)$.
  By inversion, $\exists H_1, \Gamma_0$ such that $H = H_1, y \overset{q_1}{\mapsto} b$ and $\Gamma_0 \vdash b :^{q_1} B$ and $H_1 \models \Gamma_{11} + \Gamma_{21} + \Gamma_0$.
  By IH, $\exists H_1', H_2', \Gamma_{21}', \Gamma_{22}', a'$ such that
  - $[H_1]a \Longrightarrow^q_{S \cup \{y\}} [H_1', H_2']a'$ where $|H_1'| = |H_1|$
  - $H_1', H_2' \models \Gamma_{21}', \Gamma_{22}'$ where $|\Gamma_{21}'| = |\Gamma_{21}|$
  - $\Gamma_{21}', \Gamma_{22}' \vdash a' :^q A$

  Then, we have,
  - $[H_1, y \overset{q_1}{\mapsto} b]a \Longrightarrow_S^q [H_1', y \overset{q_1}{\mapsto} b, H_2']a'$
  - $H_1', y \overset{q_1}{\mapsto} b, H_2' \models ((\Gamma_{11}, y = b :^{q_1} B) + (\Gamma_{21}', y = b :^0 B)), \Gamma_{22}'$
  - $\Gamma_{21}', y = b :^0 B, \Gamma_{22}' \vdash a' :^q A$
- Rule PTS-App. Have: $\Gamma_{21} + \Gamma_{22} \vdash b\ a^r :^q B\{a/x\}$ where $\Gamma_{21} \vdash b :^q \Pi x :^r A.B$ and $\Gamma_{22} \vdash a :^{q \cdot r} A$.
  Further, $H \models \Gamma_1 + (\Gamma_{21} + \Gamma_{22})$.
  If b steps, then this case follows by IH.
  Otherwise, b is a value. By inversion, $b = \lambda^r x : A'.b'$.
  Further, $\exists A_1, B_1$ such that $\Gamma_{21}, x :^{q \cdot r} A_1 \vdash b' :^q B_1$ and $\Pi x :^r A_1.B_1\{\Delta_2\} =_\beta \Pi x :^r A.B\{\Delta_2\}$ where $\Delta_2 = \lfloor \Gamma_{21} \rfloor$.
  Then, by rule PTS-DefConv, $\Gamma_{22} \vdash a :^{q \cdot r} A_1$. Further, $\Gamma_{21}, x = a :^{q \cdot r} A_1 \vdash b' :^q B_1$.
  Now,

- $[H](\lambda^r x : A'.b')\ a^r \Longrightarrow_S^q [H, x \overset{q\cdot r}{\mapsto} a]b'$
- $H, x \overset{q\cdot r}{\mapsto} a \models (\Gamma_1 + \Gamma_{21}), x = a :^{q\cdot r} A_1$
- $\Gamma_{21}, x = a :^{q\cdot r} A_1 \vdash b' :^q B\{a/x\}$ by rule PTS-DefConv.

– Rule PTS-DefConv. Have: $\Gamma_2 \vdash a :^q B$ where $\Gamma_2 \vdash a :^q A$ and $A\{\Delta_2\} =_\beta B\{\Delta_2\}$ and $\Delta_2 = \lfloor\Gamma_2\rfloor$.
Further, $H \models \Gamma_1 + \Gamma_2$.
By IH, $[H]a \Longrightarrow_S^q [H']a'$ and $H' \models \Gamma_2'$ and $\Gamma_2' \vdash a' :^q A$.
Let $\Delta_2' = \lfloor\Gamma_2'\rfloor$. Then, $A\{\Delta_2'\} = A\{\Delta_2\} =_\beta B\{\Delta_2\} = B\{\Delta_2'\}$.
This case, then, follows by rule PTS-DefConv.

– Rule PTS-LetPair. Have: $\Gamma_{21} + \Gamma_{22} \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^q B\{a/z\}$ where $\Delta_2, z : \Sigma x :^r A_1.A_2 \vdash_0 B : s$ and $\Gamma_{21} \vdash a :^{q\cdot q_0} \Sigma x :^r A_1.A_2$ and $\Gamma_{22}, x :^{q\cdot q_0\cdot r} A_1, y :^{q\cdot q_0} A_2 \vdash b :^q B\{(x^r, y)/z\}$ and $\Delta_2 = \lfloor\Gamma_{21}\rfloor$.
Further, $H \models \Gamma_1 + (\Gamma_{21} + \Gamma_{22})$.
If a steps, then $[H]a \Longrightarrow_{S \cup fv\ b}^{q\cdot q_0} [H']a'$ and $H' \models \Gamma_1 + (\Gamma_{21}' + \Gamma_{22})$ and $\Gamma_{21}' \vdash a' :^{q\cdot q_0} \Sigma x :^r A_1.A_2$.
Then,
- $[H]\mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b \Longrightarrow_S^{q\cdot q_0} [H']\mathbf{let}_{q_0}\ (x^r, y) = a'\ \mathbf{in}\ b$.
  By rule HeapStep-Discard, $[H]\mathbf{let}_{q_0}(x^r, y) = a\mathbf{in}b \Longrightarrow_S^q [H']\mathbf{let}_{q_0}(x^r, y) = a'\ \mathbf{in}\ b$. ($\because q_0 <: 1$)
- $H' \models \Gamma_1 + (\Gamma_{21}' + \Gamma_{22})$
- By rule ST-LetPair, $\Gamma_{21}' + \Gamma_{22} \vdash \mathbf{let}_{q_0}\ (x^r, y) = a'\ \mathbf{in}\ b :\ B\{a'/z\}$.
  Now, by lemma 26, $a'\{\Delta_2'\} =_\beta a\{\Delta_2\}$ where $\Delta_2' = \lfloor\Gamma_{21}'\rfloor$.
  Then, $B\{a/z\}\{\Delta_2\} = B\{\Delta_2\}\{a\{\Delta_2\}/z\} = B\{\Delta_2'\}\{a\{\Delta_2\}/z\} =_\beta B\{\Delta_2'\}\{a'\{\Delta_2'\}/z\} = B\{a'/z\}\{\Delta_2'\}$.
  By rule PTS-DefConv, $\Gamma_{21}' + \Gamma_{22} \vdash \mathbf{let}_{q_0}\ (x^r, y) = a'\ \mathbf{in}\ b :\ B\{a/z\}$.

Otherwise, a is a value. By inversion, $a = ({a_1}^r, a_2)$.
Further, $\exists \Gamma_{211}, \Gamma_{212}$ such that $\Gamma_{211} \vdash a_1 :^{q\cdot q_0\cdot r} A_1$ and $\Gamma_{212} \vdash a_2 :^{q\cdot q_0} A_2\{a_1/x\}$ and $\Gamma_{21} = \Gamma_{211} + \Gamma_{212}$.
Then,
- $[H]\mathbf{let}_{q_0}(x^r, y) = ({a_1}^r, a_2)\,\mathbf{in}\,b \Longrightarrow_S^q [H, x \overset{q\cdot q_0\cdot r}{\mapsto} a_1, y \overset{q\cdot q_0}{\mapsto} a_2]b$ (assuming x, y *fresh*)
- $H, x \overset{q\cdot q_0\cdot r}{\mapsto} a_1, y \overset{q\cdot q_0}{\mapsto} a_2 \models (\Gamma_1 + \Gamma_{22}), x = a_1 :^{q\cdot q_0\cdot r} A_1, y = a_2 :^{q\cdot q_0} A_2$
- $\Gamma_{22}, x = a_1 :^{q\cdot q_0\cdot r} A_1, y = a_2 :^{q\cdot q_0} A_2 \vdash b :^q B\{({a_1}^r, a_2)/z\}$.

– Rule PTS-Case. Have: $\Gamma_{21} + \Gamma_{22} \vdash \mathbf{case}_{q_0}\ a\ \mathbf{of}\ x_1.b_1\,;\, x_2.b_2 :^q B\{a/z\}$ where $\Delta_2, z : A_1 + A_2 \vdash_0 B : s$ and $\Gamma_{21} \vdash a :^{q\cdot q_0} A_1 + A_2$ and $\Gamma_{22}, x_1 :^{q\cdot q_0} A_1 \vdash b_1 :^q B\{\mathbf{inj}_1\ x_1/z\}$ and $\Gamma_{22}, x_2 :^{q\cdot q_0} A_2 \vdash b_2 :^q B\{\mathbf{inj}_2\ x_2/z\}$ and $\Delta_2 = \lfloor\Gamma_{21}\rfloor$.
Further, $H \models \Gamma_1 + (\Gamma_{21} + \Gamma_{22})$.
If a steps, then this case follows by IH.
Otherwise, a is a value. By inversion, $a = \mathbf{inj}_1\ a_1$ or $a = \mathbf{inj}_2\ a_2$.
Say $a = \mathbf{inj}_1\ a_1$. Then, $\Gamma_{21} \vdash a_1 :^{q\cdot q_0} A_1$.
Now,
- $[H]\mathbf{case}_{q_0}(\mathbf{inj}_1\, a_1)\mathbf{of}\, x_1.b_1; x_2.b_2 \Longrightarrow_S^q [H, x_1 \overset{q\cdot q_0}{\mapsto} a_1]b_1$ (assuming x_1*fresh*)
- $H, x_1 \overset{q\cdot q_0}{\mapsto} a_1 \models (\Gamma_1 + \Gamma_{22}), x_1 = a_1 :^{q\cdot q_0} A_1$
- $\Gamma_{22}, x_1 = a_1 :^{q\cdot q_0} A_1 \vdash b_1 :^q B\{\mathbf{inj}_1\ a_1/z\}$

The case when $a = \mathbf{inj}_2\ a_2$ is similar.

Theorem 23 (Soundness (Theorem 8)). *If $H \models \Gamma$ and $\Gamma \vdash a :^q A$ and $q \neq 0$, then either a is a value or there exists H', Γ', a' such that $[H]a \Longrightarrow_S^q [H']a'$ and $H' \models \Gamma'$ and $\Gamma' \vdash a' :^q A$.*

Proof. Use lemma 40 with $\Gamma_1 := 0 \cdot \Gamma$ and $\Gamma_2 := \Gamma$.

Theorem 24 (Theorem 9). *In LDC($\mathcal{Q}_\mathbb{N}$): If $\emptyset \vdash f :^1 \Pi x :^0 s.x$ and $\emptyset \vdash_0 A : s$, then $f\ A^0$ must diverge.*

Proof. (Sketch) To see why, let us assume, towards contradiction, that $f\ A^0$ terminates.

Let $A_1 := \mathbf{Unit}$ and $A_2 := \mathbf{Bool}$.

Then, for some j, H and b, we have, $[\emptyset]f\ {A_1}^0 \Longrightarrow_{1,j}^* [H](\lambda^0 x.b)\ {A_1}^0$ and $[\emptyset]f\ {A_2}^0 \Longrightarrow_{1,j}^* [H](\lambda^0 x.b)\ {A_2}^0$.

Again, $[H](\lambda^0 x.b)\ {A_1}^0 \Longrightarrow_S^1 [H, x \overset{0}{\mapsto} A_1]b$ and $[H](\lambda^0 x.b)\ {A_2}^0 \Longrightarrow_S^1 [H, x \overset{0}{\mapsto} A_2]b$ (say $x \notin S$).

But, if $[H, x \overset{0}{\mapsto} A_1]b \Longrightarrow_{1,k}^* [H', x \overset{0}{\mapsto} A_1, H'']b'$, then $[H, x \overset{0}{\mapsto} A_2]b \Longrightarrow_{1,k}^* [H', x \overset{0}{\mapsto} A_2, H'']b'$, for any k (By Lemma 28).

Now, as and when the reduction stops, b' is a value. By soundness, b' must be a value of type A_1 and of type A_2 simultaneously, a contradiction. So $f\ {A_1}^0$, $f\ {A_2}^0$, and therefore $f\ A^0$, must diverge.

As a corollary, we see that in a strongly normalizing PTS, no such f exists.

Using the same argument as above, we can also show that if $\emptyset \vdash f :^{\mathbf{L}} \Pi x :^{\mathbf{H}} s.x$ and $\emptyset \vdash A :^{\mathbf{H}} s$, then $f\ A^{\mathbf{H}}$ must diverge.

Theorem 25 (Theorem 10). *In LDC($\mathcal{Q}_\mathbb{N}$): In a strongly normalizing PTS, if $\emptyset \vdash f :^1 \Pi x :^0 s.\Pi y :^1 x.x$ and $\emptyset \vdash_0 A : s$ and $\emptyset \vdash a :^1 A$, then $f\ A^0\ a^1 =_\beta a$.*

Proof. (Sketch) To see why, let's look at how $f\ A^0\ a^1$ reduces.

Let, $A_1 := \mathbf{Unit}$ and $A_2 := \mathbf{Bool}$. Also, let $a_1 := \mathbf{unit}$ and $a_2 := \mathbf{true}$.

Then, for some j, H_1 and b, we have, $[\emptyset]f\ {A_1}^0\ {a_1}^1 \Longrightarrow_{1,j}^* [H_1](\lambda^0 x.b)\ {A_1}^0\ {a_1}^1$ and $[\emptyset]f\ {A_2}^0\ {a_2}^1 \Longrightarrow_{1,j}^* [H_1](\lambda^0 x.b)\ {A_2}^0\ {a_2}^1$.

Again, $[H_1](\lambda^0 x.b){A_1}^0 a_1 \Longrightarrow_S^1 [H_1, x \overset{0}{\mapsto} A_1]b\,{a_1}^1$ and $[H_1](\lambda^0 x.b){A_2}^0 {a_2}^1 \Longrightarrow_S^1 [H_1, x \overset{0}{\mapsto} A_2]b\ {a_2}^1$ (say $x \notin S$).

Then, $[H_1, x \overset{0}{\mapsto} A_1]b\ {a_1}^1 \Longrightarrow_{1,k}^* [H_1', x \overset{0}{\mapsto} A_1, H_2](\lambda^1 y.c)\ {a_1}^1$ and $[H_1, x \overset{0}{\mapsto} A_2]b\ {a_2}^1 \Longrightarrow_{1,k}^* [H_1', x \overset{0}{\mapsto} A_2, H_2](\lambda^1 y.c)\ {a_2}^1$, for some k (By Lemma 28).

Again, $[H_1', x \overset{0}{\mapsto} A_1, H_2](\lambda^1 y.c)\ {a_1}^1 \Longrightarrow_S^1 [H_1', x \overset{0}{\mapsto} A_1, H_2, y \overset{1}{\mapsto} a_1]c$ and $[H_1', x \overset{0}{\mapsto} A_2, H_2](\lambda^1 y.c)\ {a_2}^1 \Longrightarrow_S^1 [H_1', x \overset{0}{\mapsto} A_2, H_2, y \overset{1}{\mapsto} a_2]c$ (say $y \notin S$).

Since the reductions terminate, the value of y gets looked up. Till that look-up, the two reductions are indistinguishable from one another.

Let $[H_1', x \overset{0}{\mapsto} A_1, H_2, y \overset{1}{\mapsto} a_1]c \Longrightarrow_{1,k'}^* [H_1'', x \overset{0}{\mapsto} A_1, H_2', y \overset{1}{\mapsto} a_1, H_3]c'$ be the point at which y is looked-up. Analysing the stepping rules, $c' = y$ or a proper path headed by y (where a path is a series of nested elimination forms headed by a variable). By soundness, c' is well-typed. But there is no well-typed

proper path that can be headed by both a_1 and a_2 because $\emptyset \vdash a_1 : \mathbf{Unit}$ and $\emptyset \vdash a_2 : \mathbf{Bool}$. Therefore, $c' = y$.

This means that $f\, {A_1}^0\, {a_1}^1$ reduces to a_1 and $f\, {A_2}^0\, {a_2}^1$ to a_2, and hence $f\, A^0\, a^1$ to a. Therefore, $f\ A^0\ a^1 =_\beta a$.

G Dependency Analysis in PTS Version of LDC

Lemma 41 (Multiplication). *If $\Gamma \vdash a :^\ell A$, then $r_0 \sqcup \Gamma \vdash a :^{r_0 \sqcup \ell} A$.*

Proof. By induction on $\Gamma \vdash a :^\ell A$. Follow the proof of lemma 21.

Lemma 42 (Splitting). *If $\Gamma \vdash a :^{\ell_1 \sqcap \ell_2} A$, then there exists Γ_1 and Γ_2 such that $\Gamma_1 \vdash a :^{\ell_1} A$ and $\Gamma_2 \vdash a :^{\ell_2} A$ and $\Gamma = \Gamma_1 \sqcap \Gamma_2$.*

Proof. Have: $\Gamma \vdash a :^{\ell_1 \sqcap \ell_2} A$. By rule PTS-SubRD, $\Gamma \vdash a :^{\ell_1} A$ and $\Gamma \vdash a :^{\ell_2} A$. The lemma follows by setting $\Gamma_1 := \Gamma$ and $\Gamma_2 := \Gamma$.

Lemma 43 (Weakening (Lemma 14)). *If $\Gamma_1, \Gamma_2 \vdash a :^\ell A$ and $\Delta_1 \vdash_0 C : s$ and $\lfloor \Gamma_1 \rfloor = \Delta_1$, then $\Gamma_1, z :^\top C, \Gamma_2 \vdash a :^\ell A$.*

Proof. By induction on $\Gamma_1, \Gamma_2 \vdash a :^\ell A$.

Lemma 44 (Substitution (Lemma 15)). *If $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a :^\ell A$ and $\Gamma \vdash c :^{m_0} C$ and $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$, then $\Gamma_1 \sqcap \Gamma, \Gamma_2\{c/z\} \vdash a\{c/z\} :^\ell A\{c/z\}$.*

Proof. By induction on $\Gamma_1, z :^{m_0} C, \Gamma_2 \vdash a :^\ell A$. Follow lemma 9.

Theorem 26 (Preservation (Theorem 6)). *If $\Gamma \vdash a :^\ell A$ and $\vdash a \rightsquigarrow a'$, then $\Gamma \vdash a' :^\ell A$.*

Proof. By induction on $\Gamma \vdash a :^\ell A$ and inversion on $\vdash a \rightsquigarrow a'$. Follow theorem 21.

Theorem 27 (Progress (Theorem 7)). *If $\emptyset \vdash a :^\ell A$, then either a is a value or there exists a' such that $\vdash a \rightsquigarrow a'$.*

Proof. By induction on $\emptyset \vdash a :^\ell A$. Follow theorem 22.

Lemma 45. *If $H \models \Gamma_1 \sqcap \Gamma_2$ and $\Gamma_2 \vdash a :^\ell A$ and $l \neq \top$, then either a is a value or there exists H', Γ', a' such that:*

- $[H]a \Longrightarrow_S^\ell [H']a'$
- $H' \models \Gamma_1 \sqcap \Gamma_2'$
- $\Gamma_2' \vdash a' :^\ell A$

Proof. By induction on $\Gamma_2 \vdash a :^\ell A$. Follow lemma 40.

Theorem 28 (Soundness (Theorem 8)). *If $H \models \Gamma$ and $\Gamma \vdash a :^\ell A$ and $l \neq \top$, then either a is a value or there exists H', Γ', a' such that $[H]a \Longrightarrow_S^\ell [H']a'$ and $H' \models \Gamma'$ and $\Gamma' \vdash a' :^\ell A$.*

Proof. Use lemma 45 with $\Gamma_1 := 0 \sqcup \Gamma$ and $\Gamma_2 := \Gamma$.

H Unrestricted Use

Lemma 46 (Multiplication). *If $\Gamma \vdash a :^q A$, then $r_0 \cdot \Gamma \vdash a :^{r_0 \cdot q} A$.*

Proof. By induction on $\Gamma \vdash a :^q A$. All the cases other than rule PTS-LamOmega are similar to those of lemma 33.

- Rule PTS-LamOmega. Have: $q_0 \cdot \Gamma \vdash \lambda^r x : A.b :^{q_0 \cdot q} \Pi x :^r A.B$ where $\Gamma, x :^{q \cdot r} A \vdash b : B$ and $q = \omega \Rightarrow r = \omega$ and $q_0 \neq 0$.
  There are two cases to consider:
  - $r_0 = 0$. By IH, $0 \cdot \Gamma, x :^0 A \vdash b :^0 B$.
    This case, then, follows by rule PTS-LamOmega.
  - $r_0 \neq 0$. By rule PTS-LamOmega, then, $r_0 \cdot q_0 \cdot \Gamma \vdash \lambda^r x : A.b :^{r_0 \cdot q_0 \cdot q} \Pi x :^r A.B$.

Lemma 47 (Independence). *If $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$ and $\neg(r_0 \ll: q)$, then $\Gamma_1, z :^0 C, \Gamma_2 \vdash a :^q A$. Here, $r_0 \ll: q \triangleq \exists q_0, r_0 = q + q_0$.*

Proof. By induction on $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$.

- Rule PTS-Pi. Have: $\Gamma_{11} + \Gamma_{21}, z :^{r_{01}+r_{02}} C, \Gamma_{12} + \Gamma_{22} \vdash \Pi x :^r A.B :^q s_3$ where $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{12} \vdash A :^q s_1$ and $\Gamma_{21}, z :^{r_{02}} C, \Gamma_{22}, x :^{r'} A \vdash B :^q s_2$ and $\mathcal{R}(s_1, s_2, s_3)$ and $\neg(r_{01} + r_{02} \ll: q)$.
  Now, if $r_{01} \ll: q$ or $r_{02} \ll: q$, then $r_{01} + r_{02} \ll: q$, a contradiction.
  Therefore, $\neg(r_{01} \ll: q)$ and $\neg(r_{02} \ll: q)$.
  This case, then, follows by IH and rule PTS-Pi.
- Rule PTS-LamOmega. Have: $q_0 \cdot \Gamma_1, z :^{q_0 \cdot r_0} C, q_0 \cdot \Gamma_2 \vdash \lambda^r x : A.b :^{q_0 \cdot q} \Pi x :^r A.B$ where $\Gamma_1, z :^{r_0} C, \Gamma_2, x :^{q \cdot r} A \vdash b :^q B$ and $q = \omega \Rightarrow r = \omega$ and $q_0 \neq 0$ and $\neg(q_0 \cdot r_0 \ll: q_0 \cdot q)$.
  If $r_0 \ll: q$, then $q_0 \cdot r_0 \ll: q_0 \cdot q$, a contradiction. Therefore, $\neg(r_0 \ll: q)$.
  This case, then, follows by IH and rule PTS-LamOmega.
- Rule PTS-App. Have: $\Gamma_{11} + \Gamma_{21}, z :^{r_{01}+r_{02}} C, \Gamma_{12} + \Gamma_{22} \vdash b\ a^r :^q B\{a/x\}$ where $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{12} \vdash b :^q \Pi x :^r A.B$ and $\Gamma_{21}, z :^{r_{02}} C, \Gamma_{22} \vdash a :^{q \cdot r} A$ and $\neg(r_{01} + r_{02} \ll: q)$.
  There are two cases to consider.
  - $r = 0$. Then, $\Gamma_{21}, z :^0 C, \Gamma_{22} \vdash a :^0 A$.
    Now, if $r_{01} \ll: q$, then $r_{01} + r_{02} \ll: q$, a contradiction. Therefore, $\neg(r_{01} \ll: q)$.
    This case, then, follows by IH and rule PTS-App.
  - $r \neq 0$. Then, $\neg(r_{01} \ll: q)$ and $\neg(r_{02} \ll: q \cdot r)$.
    This case, then, follows by IH and rule PTS-App.
- Rule PTS-WPair. Have: $\Gamma_{11} + \Gamma_{21}, z :^{r_{01}+r_{02}} C, \Gamma_{12} + \Gamma_{22} \vdash (a_1{}^r, a_2) :^q \Sigma x :^r A_1.A_2$ where $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{12} \vdash a_1 :^{q \cdot r} A_1$ and $\Gamma_{21}, z :^{r_{02}} C, \Gamma_{22} \vdash a_2 :^q A_2\{a_1/x\}$ and $\neg(r_{01} + r_{02} \ll: q)$.
  Here too, there are two cases to consider.

- $r = 0$. Then, $\Gamma_{11}, z :^0 C, \Gamma_{12} \vdash a_1 :^0 A_1$.
  Now, if $r_{02} \ll: q$, then $r_{01} + r_{02} \ll: q$, a contradiction. Therefore, $\neg(r_{02} \ll: q)$.
  This case, then, follows by IH and rule PTS-WPair.
- $r \neq 0$. Then, $\neg(r_{01} \ll: q \cdot r)$ and $\neg(r_{02} \ll: q)$.
  This case, then, follows by IH and rule PTS-WPair.

– Rule PTS-LetPair. Have: $\Gamma_{11} + \Gamma_{21}, z :^{r_{01}+r_{02}} C, \Gamma_{12} + \Gamma_{22} \vdash \mathbf{let}_{q_0}(x^r, y) = a \mathbf{\ in\ } b :^q B\{a/z\}$ where $\Delta, z : \Sigma x :^r A_1.A_2 \vdash_0 B : s$ and $\Gamma_{11}, z :^{r_{01}} C, \Gamma_{12} \vdash a :^{q \cdot q_0} \Sigma x :^r A_1.A_2$ and $\Gamma_{21}, z :^{r_{02}} C, \Gamma_{22}, x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2 \vdash b :^q B\{(x^r, y)/z\}$ and $\neg(r_{01} + r_{02} \ll: q)$.
  Now, $\neg(r_{01} \ll: q \cdot q_0)$ and $\neg(r_{02} \ll: q)$.
  This case, then, follows by IH and rule PTS-LetPair.
– The other cases follow similarly.

Lemma 48 (Factorization). *If $\Gamma \vdash a :^q A$ and $q \neq 0$, then there exists Γ' such that $\Gamma' \vdash a :^1 A$ and $\Gamma <: q \cdot \Gamma'$.*

Proof. By induction on $\Gamma \vdash a :^q A$.

– Rule PTS-Pi. Have: $\Gamma_1 + \Gamma_2 \vdash \Pi x :^r A.B :^q s_3$ where $\Gamma_1 \vdash A :^q s_1$ and $\Gamma_2, x :^{r_0} A \vdash B :^q s_2$.
  By IH, $\exists \Gamma_1', \Gamma_2'$ such that $\Gamma_1' \vdash A :^1 s_1$ and $\Gamma_2', x :^{r'} A \vdash B :^1 s_2$ where $\Gamma_1 <: q \cdot \Gamma_1'$ and $\Gamma_2 <: q \cdot \Gamma_2'$.
  This case, then, follows by rule PTS-Pi.
– Rule PTS-LamOmega. Have: $q_0 \cdot \Gamma \vdash \lambda^r x : A.b :^{q_0 \cdot q} \Pi x :^r A.B$ where $\Gamma, x :^{q \cdot r} A \vdash b :^q B$ and $q = \omega \Rightarrow r = \omega$ and $q_0 \neq 0$.
  There are two cases to consider:
  - $q = \omega$. Then $q_0 \cdot q = \omega$.
    Here, we need to define an operation on contexts: for a context Γ, define $\Gamma^{\{0,\omega\}}$ as:

$$\emptyset^{\{0,\omega\}} = \emptyset$$

$$(\Gamma, x :^q A)^{\{0,\omega\}} = \begin{cases} \Gamma^{\{0,\omega\}}, x :^q A \text{ if } q \in \{0, \omega\} \\ \Gamma^{\{0,\omega\}}, x :^0 A \text{ otherwise} \end{cases}$$

    Now, since $q_0 \cdot \Gamma \vdash \lambda^r x : A.b :^\omega \Pi x :^r A.B$, by lemma 47 $(q_0 \cdot \Gamma)^{\{0,\omega\}} \vdash \lambda^r x : A.b :^\omega \Pi x :^r A.B$.
    By rule PTS-SubR, $(q_0 \cdot \Gamma)^{\{0,\omega\}} \vdash \lambda^r x : A.b :^1 \Pi x :^r A.B$.
    Note that $q_0 \cdot \Gamma <: (q_0 \cdot \Gamma)^{\{0,\omega\}} = \omega \cdot (q_0 \cdot \Gamma)^{\{0,\omega\}} = q_0 \cdot q \cdot (q_0 \cdot \Gamma)^{\{0,\omega\}}$.
  - $q \neq \omega$. By IH, $\Gamma', x :^{r'} A \vdash b :^1 B$ where $\Gamma <: q \cdot \Gamma'$ and $q \cdot r <: q \cdot r'$.
    Since $q \cdot r <: q \cdot r'$ and $q \notin \{0, \omega\}$, so $r <: r'$.
    By rule PTS-SubL, $\Gamma', x :^r A \vdash b :^1 B$.
    This case, then, follows by rule PTS-LamOmega.
– Rule PTS-App. Have: $\Gamma_1 + \Gamma_2 \vdash b\ a^r :^q B\{a/x\}$ where $\Gamma_1 \vdash b :^q \Pi x :^r A.B$ and $\Gamma_2 \vdash a :^{q \cdot r} A$.
  There are two cases to consider:

- $r = 0$. By IH, $\exists \Gamma_1'$ such that $\Gamma_1' \vdash b :^1 \Pi x :^r A.B$ and $\Gamma_1 <: q \cdot \Gamma_1'$.
  Next, by the multiplication lemma, $0 \cdot \Gamma_2 \vdash a :^0 A$.
  This case, then, follows by rule PTS-App.
- $r \neq 0$. By IH, $\exists \Gamma_1', \Gamma_2'$ such that $\Gamma_1' \vdash b :^1 \Pi x :^r A.B$ and $\Gamma_2' \vdash a :^1 A$ where $\Gamma_1 <: q \cdot \Gamma_1'$ and $\Gamma_2 <: q \cdot r \cdot \Gamma_2'$.
  Then, by the multiplication lemma, $r \cdot \Gamma_2' \vdash a :^r A$.
  This case, then, follows by rule PTS-App.

– Rule PTS-WPair. Have: $\Gamma_1 + \Gamma_2 \vdash (a_1{}^r, a_2) :^q \Sigma x :^r A_1.A_2$ where $\Gamma_1 \vdash a_1 :^{q \cdot r} A_1$ and $\Gamma_2 \vdash a_2 :^q A_2\{a_1/x\}$.
  There are two cases to consider.
  - $r = 0$. Then, by the multiplication lemma, $0 \cdot \Gamma_1 \vdash a_1 :^0 A_1$.
    By IH, $\exists \Gamma_2'$ such that $\Gamma_2' \vdash a_2 :^1 A_2\{a_1/x\}$ and $\Gamma_2 <: q \cdot \Gamma_2'$.
    This case, then, follows by rule PTS-WPair.
  - $r \neq 0$. By IH, $\exists \Gamma_1', \Gamma_2'$ such that $\Gamma_1' \vdash a_1 :^1 A_1$ and $\Gamma_2' \vdash a_2 :^1 A_2\{a_1/x\}$ and $\Gamma_1 <: (q \cdot r) \cdot \Gamma_1'$ and $\Gamma_2 <: q \cdot \Gamma_2'$.
    Then, by the multiplication lemma, $r \cdot \Gamma_1' \vdash a_1 :^r A_1$.
    This case, then, follows by rule PTS-WPair.

– Rule PTS-LetPair. Have: $\Gamma_1 + \Gamma_2 \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^q B\{a/z\}$ where $\Delta, z : \Sigma x :^r A_1.A_2 \vdash_0 B : s$ and $\Gamma_1 \vdash a :^{q \cdot q_0} \Sigma x :^r A_1.A_2$ and $\Gamma_2, x :^{q \cdot q_0 \cdot r} A_1, y :^{q \cdot q_0} A_2 \vdash b :^q B\{(x^r, y)/z\}$.
  There are two cases to consider:
  - $q = \omega$. Now, by lemma 47, $(\Gamma_1 + \Gamma_2)^{\{0,\omega\}} \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^\omega B\{a/z\}$.
    By rule PTS-SubR, $(\Gamma_1 + \Gamma_2)^{\{0,\omega\}} \vdash \mathbf{let}_{q_0}\ (x^r, y) = a\ \mathbf{in}\ b :^1 B\{a/z\}$.
    Note that $\Gamma_1 + \Gamma_2 <: (\Gamma_1 + \Gamma_2)^{\{0,\omega\}} = \omega \cdot (\Gamma_1 + \Gamma_2)^{\{0,\omega\}}$.
  - $q \neq \omega$. By IH, $\Gamma_1' \vdash a :^1 \Sigma x :^r A_1.A_2$ and $\Gamma_2', x :^{r_1'} A_1, y :^{r_2'} A_2 \vdash b :^1 B\{(x^r, y)/z\}$ where $\Gamma_1 <: q \cdot q_0 \cdot \Gamma_1'$ and $\Gamma_2 <: q \cdot \Gamma_2'$ and $q \cdot q_0 \cdot r <: q \cdot r_1'$ and $q \cdot q_0 <: q \cdot r_2'$.
    Now, since $q \notin \{0, \omega\}$, so $q_0 \cdot r <: r_1'$ and $q_0 <: r_2'$.
    By rule PTS-SubL, $\Gamma_2', x :^{q_0 \cdot r} A_1, y :^{q_0} A_2 \vdash b :^1 B\{(x^r, y)/z\}$.
    Next, by the multiplication lemma, $q_0 \cdot \Gamma_1' \vdash a :^{q_0} \Sigma x :^r A_1.A_2$.
    This case, then, follows by rule PTS-LetPair.

– The other cases follow similarly.

Lemma 49 (Splitting). *If $\Gamma \vdash a :^{q_1+q_2} A$, then there exists Γ_1 and Γ_2 such that $\Gamma_1 \vdash a :^{q_1} A$ and $\Gamma_2 \vdash a :^{q_2} A$ and $\Gamma = \Gamma_1 + \Gamma_2$.*

Proof. If $q_1 + q_2 = 0$, then $\Gamma_1 := 0 \cdot \Gamma$ and $\Gamma_2 := \Gamma$.
Otherwise, by Lemma 48, $\exists \Gamma'$ such that $\Gamma' \vdash a :^1 A$ and $\Gamma <: (q_1 + q_2) \cdot \Gamma'$.
Then, $\Gamma = (q_1 + q_2) \cdot \Gamma' + \Gamma_0$ for some Γ_0.
Now, by Lemma 46, $q_1 \cdot \Gamma' \vdash a :^{q_1} A$ and $q_2 \cdot \Gamma' \vdash a :^{q_2} A$.
By rule PTS-SubL, $q_2 \cdot \Gamma' + \Gamma_0 \vdash a :^{q_2} A$.
The lemma follows by setting $\Gamma_1 := q_1 \cdot \Gamma'$ and $\Gamma_2 := q_2 \cdot \Gamma' + \Gamma_0$.

Lemma 50 (Weakening). *If $\Gamma_1, \Gamma_2 \vdash a :^q A$ and $\Delta_1 \vdash_0 C : s$ and $\lfloor \Gamma_1 \rfloor = \Delta_1$, then $\Gamma_1, z :^0 C, \Gamma_2 \vdash a :^q A$.*

Proof. By induction on $\Gamma_1, \Gamma_2 \vdash a :^q A$.

Lemma 51 (Substitution). *If* $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$ *and* $\Gamma \vdash c :^{r_0} C$ *and* $\lfloor \Gamma_1 \rfloor = \lfloor \Gamma \rfloor$*, then* $\Gamma_1 + \Gamma, \Gamma_2\{c/z\} \vdash a\{c/z\} :^q A\{c/z\}$.

Proof. By induction on $\Gamma_1, z :^{r_0} C, \Gamma_2 \vdash a :^q A$. All the cases other than rule PTS-LamOmega are similar to those of lemma 37.

- Rule PTS-LamOmega. Have: $q_0 \cdot \Gamma_1, z :^{q_0 \cdot r_0} C, q_0 \cdot \Gamma_2 \vdash \lambda^r x : A.b :^{q_0 \cdot q} \Pi x :^r A.B$ where $\Gamma_1, z :^{r_0} C, \Gamma_2, x :^{q \cdot r} A \vdash b :^q B$ and $q = \omega \Rightarrow r = \omega$ and $q_0 \neq 0$.
  Further, $\Gamma \vdash c :^{q_0 \cdot r_0} C$ where $\lfloor \Gamma \rfloor = \lfloor \Gamma_1 \rfloor$.
  There are two cases to consider:
  - $r_0 = 0$. Then, $0 \cdot \Gamma \vdash c :^0 C$.
    By IH, $\Gamma_1, \Gamma_2\{c/z\}, x :^{q \cdot r} A\{c/z\} \vdash b\{c/z\} :^q B\{c/z\}$.
    This case, then, follows by rules PTS-LamOmega and PTS-SubL.
  - $r_0 \neq 0$. Then $q_0 \cdot r_0 \neq 0$.
    By the factorization lemma, $\exists \Gamma'$ such that $\Gamma' \vdash c :^1 C$ where $\Gamma <: q_0 \cdot r_0 \cdot \Gamma'$.
    By the multiplication lemma, $r_0 \cdot \Gamma' \vdash c :^{r_0} C$.
    By IH, $\Gamma_1 + r_0 \cdot \Gamma', \Gamma_2\{c/z\}, x :^{q \cdot r} A\{c/z\} \vdash b\{c/z\} :^q B\{c/z\}$.
    By rule PTS-LamOmega, $q_0 \cdot \Gamma_1 + q_0 \cdot r_0 \cdot \Gamma', q_0 \cdot \Gamma_2\{c/z\} \vdash \lambda^r x : A\{c/z\}.b\{c/z\} :^{q_0 \cdot q} \Pi x :^r A\{c/z\}.B\{c/z\}$.
    This case, then, follows by rule PTS-SubL.

Lemma 52 (Lambda Inversion). *If* $\Gamma \vdash \lambda^r x : A.b :^q C$*, then* $\exists A', B', \Gamma_0, q_0, q_1$ *such that:*

- $\Gamma_0, x :^{q_0 \cdot r} A' \vdash b :^{q_0} B'$
- $\Gamma <: q_1 \cdot \Gamma_0$ *and* $q_1 \cdot q_0 <: q$
- $q_0 = \omega \Rightarrow r = \omega$ *and* $q_1 \neq 0$
- $A =_\beta A'$ *and* $C =_\beta \Pi x :^r A'.B'$ *and* $\Delta \vdash_0 C : s$ *where* $\Delta = \lfloor \Gamma \rfloor$.

Proof. By induction on $\Gamma \vdash \lambda^r x : A.b :^q C$.

Theorem 29 (Preservation). *If* $\Gamma \vdash a :^q A$ *and* $\vdash a \rightsquigarrow a'$*, then* $\Gamma \vdash a' :^q A$.

Proof. By induction on $\Gamma \vdash a :^q A$ and inversion on $\vdash a \rightsquigarrow a'$. All the cases other than rule PTS-App are similar to those of lemma 21.

- Rule PTS-App. Have: $\Gamma_1 + \Gamma_2 \vdash b\ a^r :^q B\{a/x\}$ where $\Gamma_1 \vdash b :^q \Pi x :^r A.B$ and $\Gamma_2 \vdash a :^{q \cdot r} A$.
  Let $\vdash b\ a^r \rightsquigarrow c$. By inversion:
  - $\vdash b\ a^r \rightsquigarrow b'\ a^r$ when $\vdash b \rightsquigarrow b'$.
    Need to show: $\Gamma_1 + \Gamma_2 \vdash b'\ a^r :^q B\{a/x\}$.
    Follows by IH and rule PTS-App.
  - $b = \lambda^r x : A'.b'$ and $\vdash b\ a^r \rightsquigarrow b'\{a/x\}$.
    Need to show: $\Gamma_1 + \Gamma_2 \vdash b'\{a/x\} :^q B\{a/x\}$.
    Applying lemma 52 on $\Gamma_1 \vdash \lambda^r x : A'.b' :^q \Pi x :^r A.B$, we get:

* $\Gamma_0, x :^{q_0 \cdot r} A'' \vdash b' :^{q_0} B'$
* $\Gamma_1 <: q_1 \cdot \Gamma_0$ and $q_1 \cdot q_0 <: q$
* $q_0 = \omega \Rightarrow r = \omega$ and $q_1 \neq 0$
* $A'' =_\beta A'$ and $\Pi x :^r A.B =_\beta \Pi x :^r A''.B'$.

Now, there are three cases to consider.

* $q_0 = 0$. Since $q_1 \cdot q_0 <: q$, so $q = 0$.
  Then, by the substitution lemma, $\Gamma_1 + \Gamma_2 \vdash b'\{a/x\} :^0 B\{a/x\}$.
* $q_0 = \omega$. So $r = \omega$.
  Now if $q = 0$, then by the substitution lemma, $\Gamma_1 + \Gamma_2 \vdash b'\{a/x\} :^0 B\{a/x\}$.
  If $q \neq 0$, then $q \cdot r = \omega$. So $\Gamma_2 \vdash a :^\omega A$. By rule PTS-Conv, $\Gamma_2 \vdash a :^\omega A''$.
  Next, $\Gamma_0, x :^\omega A'' \vdash b' :^\omega B'$.
  By the multiplication lemma and rule PTS-SubL, $\Gamma_1, x :^\omega A'' \vdash b' :^\omega B'$.
  By the substitution lemma, $\Gamma_1 + \Gamma_2 \vdash b'\{a/x\} :^\omega B'\{a/x\}$.
  This case, then, follows by rules PTS-Conv and PTS-SubR.
* $q_0 \notin \{0, \omega\}$. By lemma 48, $\exists \Gamma_0'$ and r' such that $\Gamma_0', x :^{r'} A'' \vdash b' :^1 B'$ and $\Gamma_0 <: q_0 \cdot \Gamma_0'$ and $q_0 \cdot r <: q_0 \cdot r'$.
  Since $q_0 \notin \{0, \omega\}$, so $r <: r'$. Hence, by rule PTS-SubL, $\Gamma_0', x :^r A'' \vdash b' :^1 B'$.
  Now, by lemma 46, $q \cdot \Gamma_0', x :^{q \cdot r} A'' \vdash b' :^q B'$. By rule PTS-SubL, $\Gamma_1, x :^{q \cdot r} A'' \vdash b' :^q B'$.
  This case, then, follows by rule PTS-Conv and the substitution lemma.

Theorem 30 (Progress). *If $\emptyset \vdash a :^q A$, then either a is a value or there exists a' such that $\vdash a \rightsquigarrow a'$.*

Proof. By induction on $\emptyset \vdash a :^q A$. Follow the proof of theorem 22.

Note: The multi-substitution and elaboration lemmas (Lemmas 38 and 39) are true of LDC($\mathcal{Q}^\omega_{\mathbb{N}}$).

Lemma 53. *If $H \models \Gamma_1 + \Gamma_2$ and $\Gamma_2 \vdash a :^q A$ and $q \neq 0$, then either a is a value or there exists H', Γ', a' such that:*

- $[H]a \Longrightarrow^q_S [H']a'$
- $H' \models \Gamma_1 + \Gamma_2'$
- $\Gamma_2' \vdash a' :^q A$

Proof. By induction on $\Gamma_2 \vdash a :^q A$. All the cases other than rule PTS-App are similar to those of lemma 40.

- Rule PTS-App. Have: $\Gamma_{21} + \Gamma_{22} \vdash b\ a^r :^q B\{a/x\}$ where $\Gamma_{21} \vdash b :^q \Pi x :^r A.B$ and $\Gamma_{22} \vdash a :^{q \cdot r} A$.
  Further, $H \models \Gamma_1 + (\Gamma_{21} + \Gamma_{22})$.

If b steps, then this case follows by IH.

Otherwise, b is a value. By inversion, $b = \lambda^r x : A'.b'$.
Using lemma 52 on $\Gamma_{21} \vdash \lambda^r x : A'.b' :^q \Pi x :^r A.B$, we get:

- $\Gamma_{20}, x :^{q_0 \cdot r} A'' \vdash b' :^{q_0} B'$
- $\Gamma_{21} <: q_1 \cdot \Gamma_{20}$ and $q_1 \cdot q_0 <: q$
- $q_0 = \omega \Rightarrow r = \omega$ and $q_1 \neq 0$
- $A''\{\Delta_2\} =_\beta A'\{\Delta_2\}$ and $\Pi x :^r A.B\{\Delta_2\} =_\beta \Pi x :^r A''.B'\{\Delta_2\}$ where $\Delta_2 = \lfloor \Gamma_{21} \rfloor$.

Now, there are two cases to consider.

- $q_0 = \omega$. So $r = \omega$. Further, since $q \neq 0$, so $q \cdot r = \omega$.
  So, $\Gamma_{20}, x :^\omega A'' \vdash b' :^\omega B'$.
  By the multiplication lemma and rule PTS-SubL, $\Gamma_{21}, x :^\omega A'' \vdash b' :^\omega B'$.
  Then, by rule PTS-DefConv, $\Gamma_{22} \vdash a :^\omega A''$.
  Further, $\Gamma_{21}, x = a :^\omega A'' \vdash b' :^\omega B'$. And by rule PTS-SubR, $\Gamma_{21}, x = a :^\omega A'' \vdash b' :^q B'$.
  Now,
  * $[H](\lambda^r x : A'.b')\ a^r \Longrightarrow_S^q [H, x \overset{\omega}{\mapsto} a]b'$
  * $H, x \overset{\omega}{\mapsto} a \models (\Gamma_1 + \Gamma_{21}), x = a :^\omega A''$
  * $\Gamma_{21}, x = a :^\omega A'' \vdash b' :^q B\{a/x\}$ by rule PTS-DefConv.
- $q_0 \neq \omega$. Note that if $q_0 = 0$, then $q = 0$, a contradiction. So $q_0 \notin \{0, \omega\}$.
  By lemma 48, $\exists \Gamma'_{20}$ and r' such that $\Gamma'_{20}, x :^{r'} A'' \vdash b' :^1 B'$ and $\Gamma_{20} <: q_0 \cdot \Gamma'_{20}$ and $q_0 \cdot r <: q_0 \cdot r'$.
  Since $q_0 \notin \{0, \omega\}$, therefore $r <: r'$. Hence, by rule PTS-SubL, $\Gamma'_{20}, x :^r A'' \vdash b' :^1 B'$.
  Now, by lemma 46, $q \cdot \Gamma'_{20}, x :^{q \cdot r} A'' \vdash b' :^q B'$. By rule PTS-SubL, $\Gamma_{21}, x :^{q \cdot r} A'' \vdash b' :^q B'$.
  Next, by rule PTS-DefConv, $\Gamma_{22} \vdash a :^{q \cdot r} A''$. Further, $\Gamma_{21}, x = a :^{q \cdot r} A'' \vdash b' :^q B'$.
  Now,
  * $[H](\lambda^r x : A'.b')\ a^r \Longrightarrow_S^q [H, x \overset{q \cdot r}{\mapsto} a]b'$
  * $H, x \overset{q \cdot r}{\mapsto} a \models (\Gamma_1 + \Gamma_{21}), x = a :^{q \cdot r} A''$
  * $\Gamma_{21}, x = a :^{q \cdot r} A'' \vdash b' :^q B\{a/x\}$ by rule PTS-DefConv.

Theorem 31 (Soundness). *If $H \models \Gamma$ and $\Gamma \vdash a :^q A$ and $q \neq 0$, then either a is a value or there exists H', Γ', a' such that $[H]a \Longrightarrow_S^q [H']a'$ and $H' \models \Gamma'$ and $\Gamma' \vdash a' :^q A$.*

Proof. Use lemma 53 with $\Gamma_1 := 0 \cdot \Gamma$ and $\Gamma_2 := \Gamma$.

Theorem 32 (Theorem 11). *LDC($\mathcal{Q}_{\mathbb{N}}^\omega$) satisfies type soundness and heap soundness.*

Proof. Follows by theorems 29, 30 and 31.

I Comparison of LDC with Other Calculi

Theorem 33 (Theorem 12). *If $\Theta; \Gamma \vdash_{\mathcal{L}} e : A$, then $\overline{\Theta}^{\omega}, \overline{\Gamma}^{1} \vdash \overline{e} :^{1} \overline{A}$.*
If $\Theta \vdash_{\mathcal{C}} t : X$, then $\overline{\Theta}^{\omega} \vdash \overline{t} :^{\omega} \overline{X}$.
If $e =_{\beta} f$, then $\overline{e} =_{\beta} \overline{f}$. If $s =_{\beta} t$ then $\overline{s} =_{\beta} \overline{t}$.

Proof. By mutual induction on $\Theta; \Gamma \vdash_{\mathcal{L}} e : A$ and $\Theta \vdash_{\mathcal{C}} t : X$ for typing.
By case analysis on $e =_{\beta} f$ and $s =_{\beta} t$ for β-equality.

Theorem 34 (Theorem 13). *With $\mathcal{Q}^{\omega}_{\mathbb{N}}$ as the parametrizing structure, if $\Gamma \vdash a : A$ in* GrAD*, then $\Gamma \vdash a :^{1} A$ in LDC. Further, if $\vdash a \rightsquigarrow a'$ in* GrAD*, then $\vdash a \rightsquigarrow a'$ in LDC.*

Proof. By induction on GrAD typing judgement.

Theorem 35 (Theorem 14). *With $\mathcal{L}$ as the parametrizing structure, if $\Gamma \vdash a :^{\ell} A$ in $DDC^{\top}$, then $\Gamma \vdash a :^{\ell} A$ in LDC. Further, if $\vdash a \rightsquigarrow a'$ in $DDC^{\top}$, then $\vdash a \rightsquigarrow a'$ in LDC.*

Proof. By induction on $\mathrm{DDC}^{\top}$ typing judgement.

J Derivations for Join and Fork in LDC($\mathcal{L}$)

Proposition 2. $\emptyset \vdash c_1 : T_{\ell_1}\ T_{\ell_2}\ A \to T_{\ell_1 \sqcup \ell_2}\ A$ *where*

$c_1 := \lambda x.\eta_{\ell_1 \sqcup \ell_2}\,(\mathbf{let}\,(y^{\ell_2}, \_) = (\mathbf{let}\,(z^{\ell_1}, \_) = x\,\mathbf{in}\,z)\,\mathbf{in}\,y)$, *where* $\eta_{\ell}\ a \triangleq (a^{\ell}, \mathbf{unit})$

.

Proof.
1. $x :^{\bot} T_{\ell_1}\ T_{\ell_2}\ A \vdash x :^{\ell_1} T_{\ell_1}\ T_{\ell_2}\ A$
2. $x :^{\bot} T_{\ell_1}\ T_{\ell_2}\ A, z :^{\ell_1} T_{\ell_2}\ A \vdash z :^{\ell_1} T_{\ell_2}\ A$
3. $x :^{\bot} T_{\ell_1}\ T_{\ell_2}\ A \vdash \mathbf{let}\,(z^{\ell_1}, \_) = x\,\mathbf{in}\,z :^{\ell_1} T_{\ell_2}\ A$ [By (1) and (2)]
4. $x :^{\bot} T_{\ell_1}\ T_{\ell_2}\ A \vdash \mathbf{let}\,(z^{\ell_1}, \_) = x\,\mathbf{in}\,z :^{\ell_1 \sqcup \ell_2} T_{\ell_2}\ A$ [By (3)]
5. $x :^{\bot} T_{\ell_1}\ T_{\ell_2}\ A, y :^{\ell_1 \sqcup \ell_2} A \vdash y :^{\ell_1 \sqcup \ell_2} A$
6. $x :^{\bot} T_{\ell_1}\ T_{\ell_2}\ A \vdash \mathbf{let}\,(y^{\ell_2}, \_) = (\mathbf{let}\,(z^{\ell_1}, \_) = x\,\mathbf{in}\,z)\,\mathbf{in}\,y :^{\ell_1 \sqcup \ell_2} A$ [By (4) and (5)]
7. $x :^{\bot} T_{\ell_1}\ T_{\ell_2}\ A \vdash \eta_{\ell_1 \sqcup \ell_2}\,(\mathbf{let}\,(y^{\ell_2}, \_) = (\mathbf{let}\,(z^{\ell_1}, \_) = x\,\mathbf{in}\,z)\,\mathbf{in}\,y) :^{\bot} T_{\ell_1 \sqcup \ell_2}\ A$

Proposition 3. $\emptyset \vdash c_2 : T_{\ell_1 \sqcup \ell_2}\ A \to T_{\ell_1}\ T_{\ell_2}\ A$ *where*

$$c_2 := c_2 := \lambda x.\eta_{\ell_1}\ \eta_{\ell_2}\,(\mathbf{let}\,(y^{\ell_1 \sqcup \ell_2}, \_) = x\,\mathbf{in}\,y), \text{ where } \eta_{\ell}\ a \triangleq (a^{\ell}, \mathbf{unit})$$

.

Proof.
1. $x :^{\bot} T_{\ell_1 \sqcup \ell_2}\ A \vdash x :^{\ell_1 \sqcup \ell_2} T_{\ell_1 \sqcup \ell_2}\ A$
2. $x :^{\bot} T_{\ell_1 \sqcup \ell_2}\ A, y :^{\ell_1 \sqcup \ell_2} A \vdash y :^{\ell_1 \sqcup \ell_2} A$
3. $x :^{\bot} T_{\ell_1 \sqcup \ell_2}\ A \vdash \mathbf{let}\,(y^{\ell_1 \sqcup \ell_2}, \_) = x\,\mathbf{in}\,y :^{\ell_1 \sqcup \ell_2} A$ [By (1) and (2)]
4. $x :^{\bot} T_{\ell_1 \sqcup \ell_2}\ A \vdash \eta_{\ell_2}\,(\mathbf{let}\,(y^{\ell_1 \sqcup \ell_2}, \_) = x\,\mathbf{in}\,y) :^{\ell_1} T_{\ell_2}\ A$
5. $x :^{\bot} T_{\ell_1 \sqcup \ell_2}\ A \vdash \eta_{\ell_1}\ \eta_{\ell_2}\,(\mathbf{let}\,(y^{\ell_1 \sqcup \ell_2}, \_) = x\,\mathbf{in}\,y) :^{\bot} T_{\ell_1}\ T_{\ell_2}\ A$