On Computational Interpretations of the Modal Logic S4 II. The $\lambda\mathtt{ev}Q$-Calculus

Jean Goubault-Larrecq
Institut für Logik, Komplexität und Deduktionssysteme
Universität Karlsruhe, Am Fasanengarten 5, D-76128 Karlsruhe[*][†]
Jean.Goubault@pauillac.inria.fr, Jean.Goubault@ira.uka.de

August 29, 1996

Abstract

A language of constructions for minimal logic is the λ-calculus, where cut-elimination is encoded as β-reduction. We examine corresponding languages for the minimal version of the modal logic S4, with notions of reduction that encodes cut-elimination for the corresponding sequent system. It turns out that a natural interpretation of the latter constructions is a λ-calculus extended by an idealized version of Lisp's `eval` and `quote` constructs.

In this Part II, we repair the non-computational defect of the λ_{S4}-calculus of Part I by deriving an entirely different interpretation. Quotation closures are not provided *ex abrupto*, but are built from more primitive combinators. There is almost no freedom of choice here, and we are forced to use variants of the $\lambda\sigma$-calculus, i.e. descriptions of interpreters as formal languages. We end up defining an infinite tower of interpreters, which provides an interesting analogy with Lisp's reflexive tower. This is another argument backing the thesis that the meaning of modal constructions in S4 corresponds to `eval` and `quote`.

1 Plan

The box _ with _ for _ notation of part I installs a barrier that blocks substitutions from going down past ‘ signs, and we still have to implement this in our calculus. The main feature of this notation is that it represents some substitutions *explicitly* inside the terms. Explicit substitutions are the main feature of calculi like the $\lambda\sigma$-calculus, i.e. the λ-calculus with explicit substitutions of Abadi *et al.* [ACCL90], and in fact our calculus will be a variant of the $\lambda\sigma_{\Uparrow}$-calculus [HL89], augmented with pairs (to build stacks), terminal object (the empty stack), eval and quote.

We develop the $\lambda\mathtt{ev}Q$-calculus from these principles in Section 2, motivating each step. We then show in Section 3 that cut elimination, or equivalently reduction in the $\lambda_{S4}^{\approx}$-calculus, can be simulated in $\lambda\mathtt{ev}Q$, although the latter does not have any commutative conversion rule. We extend both λ_{S4} and $\lambda\mathtt{ev}Q$ to handle so-called modal η-like rules in Section 4; these rules do not eliminate cuts, but also simplify proofs. In Section 5, we examine other possible ways of eliminating cuts in minimal S4 proofs, and argue that they are unreasonable.

We don't prove any confluence or termination property of the new calculi. We shall examine them in Part III.

2 Eliminating Cuts by Explicit Substitutions

There are two ways of understanding how we shall build our new language, the $\lambda\mathtt{ev}Q$-calculus. The first is logical, and consists in decomposing, or precompiling the effect of the $(\Box R)$ rule. The second is computational,

[*]Research partially funded by the HCM grant 7532.7-06 from the European Union. This work started in July 1994 while I was at Bull, and was finished while I was at the university of Karlsruhe.

[†]On leave from Bull Corporate Research Center, rue Jean Jaurès, F-78340 Les Clayes sous Bois.

and comes from the idea stated above that box closures actually implement explicit substitutions that would be best represented in a form of λ-calculus with explicit substitutions.

Logically, what we are going to do is push up instances of the $(\Box R)$ rule in proofs, replacing it by simpler rules that (Cut) will be able to cross without problems. In the natural deduction system, this means pushing up instances of $(\Box\mathcal{I})$. For example, consider the case where the last rule above $(\Box\mathcal{I})$ is $(\Rightarrow E)$:

$$\begin{array}{l} \\ (\Rightarrow E) \\ (\Box\mathcal{I}) \end{array} \begin{array}{c} \Box,\ \vdash u : \Phi_1 \Rightarrow \Phi_2 \quad \Box,\ \vdash v : \Phi_1 \\ \hline \Box,\ \vdash uv : \Phi_2 \\ \hline \Box,\ ,\ ,' \vdash (uv)^{`} : \Box\Phi_2 \end{array}$$

Then we create a new operator $\star$ and a new typing rule:

$$(\Box \Rightarrow E)\ \frac{,\ \vdash u : \Box(\Phi_1 \Rightarrow \Phi_2) \quad ,\ \vdash v : \Box\Phi_1}{,\ \vdash u \star v : \Box\Phi_2}$$

which is a valid deduction rule (take $u\star v$ as an abbreviation of box xy with u, v for x, y). Now we can rewrite the former proof into:

$$\begin{array}{l} \\ (\Box\mathcal{I}) \\ (\Box \Rightarrow E) \end{array} \begin{array}{c} \begin{array}{cc} \begin{array}{c} \Box,\ \vdash u : \Phi_1 \Rightarrow \Phi_2 \\ \hline \Box,\ ,\ ,' \vdash u^{`} : \Box(\Phi_1 \Rightarrow \Phi_2) \end{array} & (\Box\mathcal{I})\ \begin{array}{c} \Box,\ \vdash v : \Phi_1 \\ \hline \Box,\ ,\ ,' \vdash v^{`} : \Box\Phi_1 \end{array} \end{array} \\ \hline \Box,\ ,\ ,' \vdash u^{`} \star v^{`} : \Box\Phi_2 \end{array}$$

One $(\Box\mathcal{I})$ (or $(\Box R)$) rule has been eliminated. The rule that corresponds to $(\Box \Rightarrow E)$ in sequent style would be:

$$(\Box \Rightarrow L)\ \frac{,\ , x : \Box\Phi_2 \vdash w : \Phi \qquad ,\ \vdash v : \Box\Phi_1}{,\ , y : \Box(\Phi_1 \Rightarrow \Phi_2) \vdash w[y \star v/x] : \Phi}$$

and this rule does not exhibit any pathological behaviour any longer with respect to (Cut).

There are some difficulties with this approach, however. Notably, it is not clear how to push instances of $(\Box\mathcal{I})$ above instances of $(\Rightarrow I)$, because in the premises of $(\Rightarrow I)$ there is an additional free variable in the assumption list, and it may not have a boxed typed: this prevents us from pushing $(\Box\mathcal{I})$ upwards. But one thing is clear: $\star$ is a combinator, and we can only succeed in pushing all instances of $(\Box\mathcal{I})$ upwards if we can translate all λ-terms into a combinatory system. These combinators should then be able to represent all reductions in the λ-calculus; and they should be typed by rules such as $(\Box \Rightarrow L)$ above, i.e. rules that are not pathological with respect to (Cut).

We shall be guided into finding this miracle combinatory system by our second, computational intuition: that we have to represent explicit substitutions in some way. The λ-calculi with explicit substitutions [ACCL90] are just combinatory systems that have the required properties. There are many variants of them, so we shall explain the basic constructions on some simple variant of Curien's system of categorical combinators first [Cur86]. Before we embark on building our language, we wish to make a few comments on the relation with Lisp.

The traditional solution to implement `eval` in Lisp is to include an interpreter for the language itself, wrapped inside the `eval` primitive. What can we understand by "interpreter" in a λ-calculus setting? In a broad sense, we need a general evaluation mechanism, whether it is a syntax-directed interpreter, or a bytecode interpreter, or even a system that compiles a piece of text and executes it at run-time: in the recursion-theoretic sense, what we mean by an interpreter is a *machine*. Abadi *et al.* [ACCL90] argue that the $\lambda\sigma$-calculus is precisely a way of understanding the λ-calculus from an machine-oriented point of view. In fact, they show how to derive, with minimal efforts, a machine — an interpreter — for implementing the λ-calculus from the rules of the $\lambda\sigma$-calculus. Notice in particular that a term of the form box u with $v_1, \ldots, v_n$ for $x_1, \ldots, x_n$ represents a program u, in the context of a *stack* with n entries containing v_1, ..., v_n respectively, and packaged as a syntactic object. (Abadi *et al.* separate the stack from the substitution in the machine, but we can see them as two parts of a global substitution or stack.)

A natural question we can ask is whether we could use some other system of combinators, for example Schönfinkel and Curry's S and K (whose types are the Hilbert axioms (s) and (k)). The answer is, unfortunately, no: SK-reduction, or *weak reduction*, fails to reduce some translations of λ-terms that would be reduced by the β-rule. Proof-theoretically, SK-reductions do not eliminate all cuts. It may be possible to correct this by finding a proof-theoretic equivalent of Curry's equations [Bar84], but it does not seem rewarding: Curry's equations are inscrutable and will hardly give us any insight into what happens during cut-elimination. We are therefore committed to using categorical combinators or, in general, $\lambda\sigma$-calculi.

2.1 Categorical Combinators

We first provide a quick tour through categorical combinators, and explore the difficulties that await us, then we vindicate and explain our solution.

Because Curien's (CCL_β) is simpler than other λ-calculi with explicit substitutions, we shall explain the intuitions behind our constructions by using a formalism of categorical combinators very much like (CCL_β). We shall then develop our constructions for real in Section 2.4.

Our language of categorical combinators will be based on the following constants and operators, at least so far as ’ (unbox) and ‘ (box) are not concerned. Do remember, when considering the typing rules we give, that we are trying to construct terms u‘ of type $\Box\Phi$ in a systematic way from the structure of u of type Φ.

Because the categorical combinators manipulate not only terms but also stacks of terms, we need additional logical connectives for typing stacks: first, we need the type $\top$ of the empty stack (alternatively, the value "true", or the terminal object in a cartesian closed category); then, we need to express how to extend a stack $s : S$ by pushing an element $a : A$ onto it: this will be a couple (a, s) of type $A \times S$, where $\times$ introduces product types (alternatively, conjunction; we did not denote it by $\wedge$, as this conjunction may be totally different from any conjunction we might wish to add to the original logic). Another adjustment we have to make is the following: we introduce a new binary connective $\overset{\Box}{\Rightarrow}$, and consider $\Box\Phi$ as being an abbreviation for $\top \overset{\Box}{\Rightarrow} \Phi$. ($\Box$ is no longer a primitive connective in the logic.) Conversely, $\Psi \overset{\Box}{\Rightarrow} \Phi$ will be logically equivalent to $\Box(\Psi \Rightarrow \Phi)$.

The combinators are as follows:

- If u is a term of type $\Phi_1 \times \Phi_2 \overset{\Box}{\Rightarrow} \Phi_3$, then $\lambda^\star u$ is a term of type $\Phi_2 \overset{\Box}{\Rightarrow} \Phi_1 \Rightarrow \Phi_3$. ($\lambda^\star$ is the currification, or abstraction operator; this is the λ operator of the $\lambda\sigma$-calculus.)
- If u is a term of type $\Phi_1 \overset{\Box}{\Rightarrow} \Phi_2 \Rightarrow \Phi_3$, and v is a term of type $\Phi_1 \overset{\Box}{\Rightarrow} \Phi_2$, then $u \star v$ is a term of type $\Phi_1 \overset{\Box}{\Rightarrow} \Phi_3$. ($\star$ is the application operator, also called the evaluation operator in categorical circles; don't confuse it with the ’ evaluation operator. This is also called application in $\lambda\sigma$.)
- If u is a term of type $\Phi_2 \overset{\Box}{\Rightarrow} \Phi_3$ and v is a term of type $\Phi_1 \overset{\Box}{\Rightarrow} \Phi_2$, then $u \circ v$ is a term of type $\Phi_1 \overset{\Box}{\Rightarrow} \Phi_3$. (The composition operator. In $\lambda\sigma$, this would also be the explicit substitution application operator.)
- If u is a term of type $\Phi_1 \overset{\Box}{\Rightarrow} \Phi_2$ and v is a term of type $\Phi_1 \overset{\Box}{\Rightarrow} \Phi_3$, then $u{\bullet}^\star v$ is a term of type $\Phi_1 \overset{\Box}{\Rightarrow} \Phi_2 \times \Phi_3$. (The pairing combinator, or cons in $\lambda\sigma$, which pushes u on top of stack v.)
- $\uparrow^\star$ is a constant of type $\Phi_1 \times \Phi_2 \overset{\Box}{\Rightarrow} \Phi_2$ (the shift operation, which pops the top of stack in $\lambda\sigma$), $1^\star$ is a constant of type $\Phi_1 \times \Phi_2 \overset{\Box}{\Rightarrow} \Phi_1$ (the operation $\mathbf{1}$ that gets the top of stack, in $\lambda\sigma$), and $id^\star$ is a constant of type $\Phi \overset{\Box}{\Rightarrow} \Phi$ (this plays the rôle of the "placeholder for stacks" id in $\lambda\sigma$, and also of the empty stack).

We also adopt the following handy abbreviations. First, we assume that $\times$ associates to the right, i.e. $\Phi_0 \times \Phi_1 \times \ldots \times \Phi_n \times \Psi = \Phi_0 \times (\Phi_1 \times \ldots \times (\Phi_n \times \Psi) \ldots)$. Then, we also write pop_n for $\underbrace{(\uparrow^\star \circ (\uparrow^\star \circ \ldots \circ \uparrow^\star) \ldots)}_{n \text{ times}}$ if $n \geq 1$, or for $id^\star$ if $n = 0$. We write get_n for $1^\star \circ \mathsf{pop}_n$ if $n \geq 1$, or for $1^\star$ if $n = 0$.

The reduction rules, adapted from Curien's (or from Abadi *et al.*'s), are the following:

$(\circ)$ $(u \circ v) \circ w \to u \circ (v \circ w)$

$(\circ id^\star)$ $u \circ id^\star \to u$

$(id^\star \circ)$ $id^\star \circ u \to u$

$(\uparrow^\star)$ $\uparrow^\star \circ (u{\bullet}^\star v) \to v$

$(1^\star)$ $1^\star \circ (u{\bullet}^\star v) \to u$

$(\beta^\star)$ $(\lambda^\star u) \star v \to u \circ (v \bullet^\star id^\star)$

$(\bullet^\star)$ $(u \bullet^\star v) \circ w \to (u \circ w) \bullet^\star (v \circ w)$

$(\star)$ $(u \star v) \circ w \to (u \circ w) \star (v \circ w)$

$(\lambda^\star)$ $(\lambda^\star u) \circ w \to \lambda^\star(u \circ (1^\star \bullet^\star (w \circ \uparrow^\star)))$

Operationally speaking, these rules can be seen as describing a stack machine for evaluating λ-terms, as already announced. Read $u \circ v$ as "u in the context of stack v" or as "do v then u", according to context. We can derive the rule $\mathsf{get}_i \circ (v_0 \bullet^\star (v_1 \bullet^\star \ldots \bullet^\star (v_i \bullet^\star u) \ldots)) \longrightarrow^* v_i$, so that we can read get_i as the instruction that reads stack location i (the top of stack is at index 0). We can also derive the rule $\mathsf{pop}_i \circ (v_0 \bullet^\star \ldots \bullet^\star (v_{i-1} \bullet^\star u) \ldots) \to u$, so pop_i is the instruction that pops i stack locations. Finally $u \bullet^\star w$ pushes u on stack w and returns the resulting stack. The $(\beta^\star)$ rule describes β-reduction: to evaluate a λ-expression applied to v, push v on the current stack and evaluate the body u of the λ-expression in this new stack. (In particular, if we evaluate $(\lambda^\star u) \star v$ in the stack w, then $((\lambda^\star u) \star v) \circ w \xrightarrow{(\beta^\star)} (u \circ (v \bullet^\star id^\star)) \circ w \xrightarrow{(\circ)} u \circ (v \bullet^\star id^\star) \circ w) \xrightarrow{(\bullet^\star)} u \circ ((v \circ w) \bullet^\star (id^\star \circ w)) \xrightarrow{(id^\star \circ)} u \circ (v \circ w \bullet^\star w)$, i.e. we evaluate u in the stack formed by pushing the value of v in stack w onto stack w.) The last three rules express the fact that to evaluate an expression in a stack, we have to evaluate its constituents in the same stack, or (last rule) in some extended stack.

Now, given a pure λ-term u, we build u‘ by translating u into a categorical combinator form using the constructions above. The translation is defined as follows. We use an environment ρ, which is a map from variable names to integer indices; we define u‘ as u‘$[]$, where $[]$ is the empty environment, and in general define u‘ρ by:

- $x\text{‘}\rho = \begin{cases} \mathsf{get}_{n-i-1} & \text{if } \rho(x) = i \\ Qx & \text{if } \rho(x) \text{ is undefined} \end{cases}$ where n is the cardinality of ρ;
- $(uv)\text{‘}\rho = (u\text{‘}\rho) \star (v\text{‘}\rho)$;
- $(\lambda x \cdot u)\text{‘}\rho = \lambda^\star(u\text{‘}\rho[x \mapsto n])$, where $\rho[x \mapsto n]$ is ρ with the binding $x \mapsto n$ added, and n is the cardinality of ρ.

The environment ρ is used to keep track of the positions allotted to each local variable in the run-time stack. The additional operator Q is such that:

- if x has type $\Phi_1 \overset{\Box}{\Rightarrow} \Phi_2$, then Qx has type $\Phi_3 \overset{\Box}{\Rightarrow} \Phi_1 \overset{\Box}{\Rightarrow} \Phi_2$.

Defining the Q operator of this type is legal, intuitively, because if we understand $\_ \overset{\Box}{\Rightarrow} \_$ as meaning $\Box(\_ \Rightarrow \_)$, the type $(\Phi_1 \overset{\Box}{\Rightarrow} \Phi_2) \Rightarrow (\Phi_3 \overset{\Box}{\Rightarrow} \Phi_1 \overset{\Box}{\Rightarrow} \Phi_2)$ is inhabited. Indeed, using the functional interpretation for the Hilbert-style system, we may understand Qx as the analogue of ‘$K \star kwote\ x$. This Q operator is mostly the analogue of `kwote` in the categorical combinator algebra.

It then happens that the following reduction rule:

$$Qu \circ v \to Qu$$

is legal from the type perspective, and we add it.

2.2 Adding Eval

This translation is fine for forming quotations of pure λ-terms, but what about λ-terms that already contain some quotations or evaluations? And what about quoting or evaluating categorical combinator terms (i.e., quoted terms)?

We first deal with ‚. This incurs two separate problems: we have to define how ‚ applies to quoted terms, and we have to provide a translation of terms of the form ‚u to quote them.

First, define the reduction rules of ‚ applied to quoted objects. Reducing ‚$(\lambda^\star u)$ poses a problem, in that we would like it to reduce to $\lambda x \cdot e(u, x)$, where e is some evaluation function for u in the context x. We are therefore forced to trade the unary operator ‘ for a binary operator $\mathtt{ev}^\star$ that takes a quoted expression to evaluate and a context in which to evaluate it. (This is also a classical way of defining an interpreter in Lisp.) Therefore, from now on, we shall consider ‘u as an abbreviation for $\mathtt{ev}^\star u()$, where $() : \top$ denotes the empty stack (and $\top$, the type of the empty stack, is the logical constant true).

We also have the choice of defining $\mathtt{ev}^\star$ as a binary function, taking a term to evaluate and a stack; or as a unary function, taking a term and returning a function from stacks to terms; or as a constant, or ... We shall consider $\mathtt{ev}^\star$ as a binary function: we shall see in Section 2.4 that otherwise confluence would not hold. (See the discussion on rule ($\mathtt{ev}\Uparrow^\ell$).)

So, the typing rule for $\mathtt{ev}^\star$ is:

- if $u : \Psi \overset{\Box}{\Rightarrow} \Phi$ and $v : \Psi$, then $\mathtt{ev}^\star uv : \Phi$,

And the reduction rules come along:

$$
\begin{array}{rl}
(\mathtt{ev}\lambda^\star) & \mathtt{ev}^\star(\lambda^\star u)w \to \lambda x \cdot \mathtt{ev}^\star u(x, w) \\
(\mathtt{ev}\star) & \mathtt{ev}^\star(u \star v)w \to (\mathtt{ev}^\star uw)(\mathtt{ev}^\star vw) \\
(\mathtt{ev}\circ) & \mathtt{ev}^\star(u \circ v)w \to \mathtt{ev}^\star u(\mathtt{ev}^\star vw) \\
(\mathtt{ev}\bullet^\star) & \mathtt{ev}^\star(u \bullet^\star v)w \to \mathtt{ev}^\star uw \bullet \mathtt{ev}^\star vw \\
(\mathtt{ev}\uparrow^\star) & \mathtt{ev}^\star \uparrow^\star w \to \uparrow w \\
(\mathtt{ev}1^\star) & \mathtt{ev}^\star 1^\star w \to 1w \\
(\mathtt{ev}id^\star) & \mathtt{ev}^\star id^\star w \to w \\
(\mathtt{ev}Q) & \mathtt{ev}^\star(Qu)w \to u
\end{array}
$$

where we have had to add new couple $\bullet$ and projection operators $\uparrow$, 1 to account for evaluations of $\bullet^\star$ and $\uparrow^\star$, $1^\star$ respectively. Their types are: $u \bullet v$ has type $\Phi_1 \times \Phi_2$ whenever u has type Φ_1 and v has type Φ_2; whenever u has type $\Phi_1 \times \Phi_2$, $\uparrow u$ has type Φ_2 and $1u$ has type Φ_1. (That is, we add a conjunction $\times$ to the original λ-calculus.)

It is clear that we should have $(\uparrow u)‘\rho = \uparrow^\star \circ u‘\rho$, $(1u)‘\rho = 1^\star \circ u‘\rho$, and $(u \bullet v)‘\rho = (u‘\rho)\bullet^\star(v‘\rho)$. Quoting $()$ is a bit trickier. Looking at types, $()‘\rho$ should be of type $\Phi \overset{\Box}{\Rightarrow} \top$, that is, it should throw away the whole stack, of type Φ, and produce the empty stack, of type $\top$. But a stack of height n will have a type Φ of the form $\Phi_1 \times \ldots \times \Phi_n \times \top$, so pop_n has the desired type. We therefore define:

- $()‘\rho = \mathsf{pop}_n$ where n is the cardinality of ρ;

Because the stack is of height n, $()‘\rho$ produces the empty stack by just popping n elements off the stack. This way, we avoid introducing new constants, and simplify later calculi. (This works not so much because we are in a typed universe, but because scoping is lexical: new constants would have to be introduced in non-lexical languages.)

Then, we have to define how we quote terms of the form $\mathtt{ev}^\star uv$. We could use the following trick: $\mathtt{ev}^\star$ alone is a perfectly valid function, of type $(\Phi_1 \overset{\Box}{\Rightarrow} \Phi_2) \Rightarrow \Phi_1 \Rightarrow \Phi_2$. It is valid because its type is indeed provable in S4, provided we understand $\Phi_1 \overset{\Box}{\Rightarrow} \Phi_2$ as $\Box(\Phi_1 \Rightarrow \Phi_2)$. Now, $\Psi \overset{\Box}{\Rightarrow} ((\Phi_1 \overset{\Box}{\Rightarrow} \Phi_2) \Rightarrow \Phi_1 \Rightarrow \Phi_2)$ is also provable, which means that we can posit the existence of a special constant $\mathtt{ev}^{\star\star}$, with the following typing rule:

- $\mathtt{ev}^{\star\star} : \Psi \overset{\Box}{\Rightarrow} (\Phi_1 \overset{\Box}{\Rightarrow} \Phi_2) \Rightarrow \Phi_1 \Rightarrow \Phi_2$

If $\mathtt{ev}^\star$ was just a constant, we might then quote $\mathtt{ev}^\star uv$ by reading this term as the application of $\mathtt{ev}^\star$ to u and v, introducing a new constant $\mathtt{ev}^{\star\star}$ and using the translation:

- $\mathtt{ev}^\star‘\rho = \mathtt{ev}^{\star\star}$

We shall see in Section 2.3 that using this kind of trick is not really enough, and we shall do otherwise.

2.3 Adding Quote

Adding quote is again the hardest task at hand. To see what we have to do, notice that the typing rules for $\lambda^\star$, $\star$, $\bullet^\star$, $\uparrow^\star$, $1^\star$, $id^\star$ are some kind of an internal natural deduction system. By "internal" we mean that this system handles sequents by representing them inside the very language of formulas: consider the symbol $\overset{\Box}{\Rightarrow}$ as the analogue of the $\vdash$ sequent formation arrow, and read $\top \times \Phi_n \times \ldots \times \Phi_0 \overset{\Box}{\Rightarrow} \Phi$ as the internalized version of the sequent $\Phi_n, \ldots, \Phi_0 \vdash \Phi$. We then have the following correspondence:

$\lambda^\star$ typing rule	$\sim$	Internal $(\Rightarrow I)$
$\star$ typing rule	$\sim$	Internal $(\Rightarrow E)$
$\circ$ typing rule	$\sim$	Internal cut
$\bullet^\star$ typing rule	$\sim$	Internal $(\times I)$
$\uparrow^\star$ typing rule	$\sim$	Internal weakening and $(\times E)$
$1^\star$ typing rule	$\sim$	Internal (Ax) and $(\times E)$
$id^\star$	$\sim$	Internal (Ax) and $(\top I)$
$\mathtt{ev}^{\star\star}$ typing rule	$\sim$	Internal $(\Box E)$

The last one is less clear than the others: notice that if $u : \Phi_0 \times \ldots \times \Phi_n \times \top \overset{\Box}{\Rightarrow} \Box\Phi$, where $\Box\Phi$ is an abbreviation for $\top \overset{\Box}{\Rightarrow} \Phi$, then $\mathtt{ev}^{\star\star} \star u \star Q() : \Phi_0 \times \ldots \times \Phi_n \times \top \overset{\Box}{\Rightarrow} \Phi$, which is indeed the internal version of $(\Box E)$. Also, most other rules need to be weakened to find back the full version of the original rules.

A main difference between (external) sequents and formulas are that the left-hand sides of sequents are sets, whereas left-hand sides of $\overset{\Box}{\Rightarrow}$-formulas are lists of formulas. The gap between sets and lists is filled in by the conjunction rules, which handle all chores of permuting, contracting or dropping assumptions (the so-called *structural rules* of logic).

Given this view of $\overset{\Box}{\Rightarrow}$-formulas as internal sequents, it is now easy to see what kind of rule we have to find, namely an internal version of $(\Box I)$:

- if u is a term of type $\Box\Phi_0 \times \ldots \times \Box\Phi_n \times \top \overset{\Box}{\Rightarrow} \Phi$, then $‘^\star u$ is a term of type $\Box\Phi_0 \times \ldots \times \Box\Phi_n \times \top \overset{\Box}{\Rightarrow} \Box\Phi$ ($\sim$ $(\Box I)$ without weakening, internally; and $‘^\star$ is then the internal quote operator).

The $‘^\star$ operator poses several problems. First, its computational content is not trivial, since $‘^\star$ does not operate on a mere part of the formula, but has to check that the left-hand side of $\overset{\Box}{\Rightarrow}$ really is a conjunction of $\top$ and of boxed formulas. But there is another problem: compositions $\circ$ are the internalized version of the cut rule, and to eliminate such internalized cuts in the presence of an internal quote operator $‘^\star$ is the same task as eliminating ordinary cuts in the presence of the quote operator $‘$. (Apart from the fact that cuts with an axiom, namely weakenings, or from the computational viewpoint, the $\mathtt{get}_n$ variable-fetching instructions, cannot be eliminated).

Since the problems posed by the $‘^\star$ operator are exactly the same as those posed by the $‘$ operator, we shall adopt the same solution: consider the operators $\lambda^\star$, $\star$, $\circ$, $\bullet^\star$, $\uparrow^\star$, $1^\star$, $id^\star$, Q and $\mathtt{ev}^\star$ as being operators at level 1, and write them λ^1, $\star^1$, $\circ^1$, $\bullet^1$, $\uparrow^1$, 1^1, id^1, Q^1 and $\mathtt{ev}^1$; then add similar operators λ^2, $\star^2$, $\circ^2$, $\bullet^2$, $\uparrow^2$, 1^2, id^2, Q^2 and $\mathtt{ev}^2$ at level 2 with similar reduction rules. We shall again have the same problem encoding quote in the new embedded calculus, so we create new operators with a 3 superscript, then a 4, a 5, and so on. This creates infinitely many operators, corresponding to infinitely many levels of computation (operators at level ℓ compute under ℓ boxes).

This infinite regression of embeddings of combinatory systems — of interpreters — is also visible in Lisp: this is the so-called *reflexive* property of Lisp systems. In Lisp, we can produce a syntactic representation — a piece of data — of the code for an interpreter; if we evaluate this code (say, by compiling it and executing it), we get the Lisp interpreter. If we submit this same code to the thus obtained Lisp interpreter, we get a level 2 interpreter. Then we can interpret this code again by the level 2 interpreter to get a level 3 interpreter, and so on. (This leads to drastic losses in performance, but this is not the point of this paper.) There are still differences between Lisp and our language. One of them is that, in Lisp, we only need one interpreter to implement the infinite tower of interpreters. On the other hand, in Lisp, we never evaluate under a `quote`, although we are allowed to do this in our language. If we only used weak notions of reductions, where we don't reduce under λ or under $‘$, we would only need one evaluator, namely rules $(\mathtt{ev}\lambda^\star)$ through $(\mathtt{evev}^\star)$. At least, we could dispense with having infinitely many operators, as Lisp does, by encoding operators at level

2 and up with finitely many constants, but this brings little benefit for the study of both the calculus and the deduction system.

2.4 The $\lambda\mathtt{ev}Q$-Calculus

To sum up, we have arrived at our final language of constructions, in its (yet) untyped form. This is an extension of the $\lambda\sigma_{\Uparrow}$-calculus: a special operator $\Uparrow^\ell$ is used to wrap stacks that have been pushed under a λ, and there are two sorts of terms, the elementary terms and the stacks. This forces us to split $\circ$, $\mathtt{ev}$ and Q in two versions, one when their first argument is an elementary term, and one where it is a stack.

Definition 2.1 **($\lambda\mathtt{ev}Q$)** *The set of* $\lambda\mathtt{ev}Q$-terms *is defined by the following syntax, where x, y, z, … denote (elementary) term variables taken from an infinite set $\mathcal{V}$. Terms s, t, u, v, w, … are elements of the language $T \cup S$, where T is the language of* elementary terms *and S is that of* explicit substitutions *or* stacks*:*

$$\begin{array}{lcl} T & ::= & \mathcal{V} \mid \lambda\mathcal{V}\cdot T \mid TT \mid 1S \mid \lambda^\ell T \mid T \star^\ell T \mid T \circ^\ell_T S \mid 1^\ell \mid \mathtt{ev}^\ell_T TS \mid Q^\ell_T T \\ S & ::= & () \mid T \bullet S \mid \uparrow S \mid S \circ^\ell_S S \mid id^\ell \mid T \bullet^\ell S \mid \uparrow^\ell \mid \Uparrow^\ell S \mid \mathtt{ev}^\ell_S SS \mid Q^\ell_S S \end{array}$$

modulo α-renaming, and where ℓ varies among all integers ≥ 1.

By extension, we also write $u \bullet^0 v$ for $u \bullet v$, and $u \star^0 v$ for uv. Moreover, we shall write $\circ^\ell$ for either $\circ^\ell_S$ or $\circ^\ell_T$, $\mathtt{ev}^\ell$ for either $\mathtt{ev}^\ell_S$ or $\mathtt{ev}^\ell_T$, and Q^ℓ for either Q^ℓ_S or Q^ℓ_T ambiguously: the sorts will tell which one is intended.

Although this language contains sorts, we shall also sometimes refer to it as the *untyped* $\lambda\mathtt{ev}Q$-calculus. This is in contrast with the typed calculus, which we introduce in Definition 2.2.

Notice that if we forget about the ℓ subscript, $u \circ^\ell_T v$ is $u[v]$ of the $\lambda\sigma$-calculus, and $u \circ^\ell_S v$ is $u \circ v$. Notice also that, if $\mathtt{ev}^\ell_T$ is similar to Lisp's `eval`, then $\mathtt{ev}^\ell_S$ is Lisp's `evlis`. In the sequel, we shall happily leave the sorts implicit whenever possible, writing $\circ^\ell$, $\mathtt{ev}^\ell$, Q^ℓ.

There is no apparent reason why we split terms into elementary terms and stacks, and we might have reunited them for good. However, we shall be interested in an extension of the calculus with so-called η-like rules (see Section 4). This will be an extension of the $\lambda\sigma$-calculus instead of the $\lambda\sigma_{\Uparrow}$-calculus. The latter calculus is only confluent when only elementary term variables, and no stack varibles, are allowed. Since we need elementary term variables (as bound variables in λ)-abstractions, we need to separate elementary terms from stacks.

In fact, just having two sorts, T and S, will not be enough to get a well-behaved calculus. Indeed, the part that propagates substitutions down in this calculus does not terminate (see Part III). In fact, the calculus with the η-like rules won't be confluent either. Our efforts to refine the sorts T and S have led to a calculus that is almost the following typed system, so we consider the typed system directly:

Definition 2.2 (Types) *The set $\mathcal{T}$ of* term types*, the set $\mathcal{S}$ of* stack types *and the set $\mathcal{M}$ of* metastack types *are defined as follows:*

$$\begin{array}{lcl} \mathcal{T} & ::= & B \mid \mathcal{T} \Rightarrow \mathcal{T} \mid \mathcal{S} \overset{\Box}{\Rightarrow} \mathcal{T} \\ \mathcal{S} & ::= & \top \mid \mathcal{T} \times \mathcal{S} \\ \mathcal{M} & ::= & \mathcal{S} \mid \mathcal{S} \overset{\Box}{\Rightarrow} \mathcal{M} \end{array}$$

We call types *or* formulas *any term, stack or metastack types.*

We denote types by Φ, Ψ, …, (possibly primed or subscripted), term types by τ, τ', …, stack types by ς, ς', …, and metastack types by μ, μ', … Observe that every type is either a term type or a metastack type, and not both at the same time.

We use the following convention. For every ℓ-tuple $\overline{\varsigma^\ell}$ of stack types $(\varsigma_1, \ldots, \varsigma_\ell)$, we write $\overline{\varsigma^\ell \overset{\Box}{\Rightarrow}}\Psi$ for $\varsigma_1 \overset{\Box}{\Rightarrow} \varsigma_2 \overset{\Box}{\Rightarrow} \ldots \varsigma_\ell \overset{\Box}{\Rightarrow} \Psi$ (or simply Ψ if $\ell = 0$).

The typing rules are then given in Figure 1, where contexts , are finite sets of assumptions of the form $x : \tau$, where x is a term variable and τ is a term type; furthermore, in rules $(\Box^\ell\Box E)$, $(\Box^\ell\Box I)$ and $(\Box^\ell Cut)$, the operators $\mathtt{ev}^\ell$, Q^ℓ and $\circ^\ell$ denote $\mathtt{ev}^\ell_T$, Q^ℓ_T and $\circ^\ell_T$ respectively if and only if Φ is a term type τ, and denote $\mathtt{ev}^\ell_S$, Q^ℓ_S and $\circ^\ell_S$ respectively if and only if Φ is a metastack type μ.

Alternatively, we consider this as a natural deduction system for S4 (with nested sequents), extended by using $\overset{\Box}{\Rightarrow}$ in place of $\Box$ (and of $\vdash$ in nested sequents). The idea is that the type of elementary terms will be a term type, and the type of stacks will be a metastack type.

Notice that all the rules at levels 1 and up are actually almost Hilbert-style rules. From the deductive point of view, the $(\Box^\ell \Uparrow)$ rule is superfluous. On the other hand, useful rules from the deductive point of view and which are derivable from the rules of Figure 1 are:

$$\frac{,\ \vdash u : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma \overset{\Box}{\Rightarrow} \Phi}{,\ \vdash u \circ^\ell \uparrow^\ell : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}}\tau \times \varsigma \overset{\Box}{\Rightarrow} \Phi}$$

(a weakening rule) where $\circ^\ell$ is $\circ^\ell_T$ if Φ is a term type, and $\circ^\ell_S$ if Φ is a metastack type; and:

$$(\Box^\ell \times E) \qquad \frac{,\ \vdash u : \overline{\varsigma^{\ell} \overset{\Box}{\Rightarrow}}\tau \times \varsigma}{\begin{array}{c},\ \vdash 1^\ell \circ^\ell_T u : \overline{\varsigma^{\ell} \overset{\Box}{\Rightarrow}}\tau \\ (\text{resp. } ,\ \vdash \uparrow^\ell \circ^\ell_S u : \overline{\varsigma^{\ell} \overset{\Box}{\Rightarrow}}\varsigma)\end{array}}$$

which are boxed conjonction elimination rules. Notice that the typing rules use explicit boxed cut rules; The idea of adding an explicit cut-rule to a natural deduction system to represent a calculus with explicit substitutions is originally due to Hugo Herbelin and Bruno Pagano.

The translation rules from terms of the λ_{S4}-calculus to the new language are given in Figure 2. The G-translation uses an auxiliary *quotation function* $u \mapsto u`\rho$, where ρ is an *environment* mapping variables to integer indices. The cardinality of ρ is the number of variables that it binds. Because of our conventions that $\star^0 = \star$ and $\bullet^0 {=} \bullet$, we could have dispensed with the statement of the translations of application uv and pairing $u \bullet v$. We have left them for clarity.

The G-translation is well-defined:

Lemma 2.1 *For every λ_{S4}-term u, $G(u)$ is a term of sort T. Moreover, for every environment ρ, the quotation function $u \mapsto u`\rho$ maps terms of sort T to terms of sort T, and terms of sort S to terms of sort S.*

Proof: We first show that $u \mapsto u`\rho$ maps terms of sort T to terms of sort T, and terms of sort S to terms of sort S. This is a routine check.

We now prove that $G(u)$ is a term of sort T for every u, by structural induction on u. The only non-trivial case is when u is of the form $\mathtt{box}\ w\ \mathtt{with}\ v_1, \ldots, v_n\ \mathtt{for}\ x_1, \ldots, x_n$. By induction hypothesis, $G(w)$ is of sort T, hence $(G(w))`[]$ is of sort $T$. By induction hypothesis again, $G(v_1)$, $\ldots$, $G(v_n)$ are of sort $T$, hence $((G(w))`[])[G(v_1)/x_1, \ldots, G(v_n)/x_n]$ is well-sorted again, and of sort T. $\Box$

The reduction rules that we need are given in Figure 3, and Figure 4. Intuitively, the rules of Figure 4 are sorting rules, and allow terms of lower levels ℓ to go down through terms of higher levels $\mathcal{L}$ until they come to their proper place, in a context of level ℓ. These rules are needed to simulate cut elimination in S4 (see Section 3.2).

The reduction rules are to be read by replacing every $\mathtt{ev}^\ell$, $\circ^\ell$ and Q^ℓ by their respective versions with an S or T index, so that the rules are well-sorted, i.e. so that both sides are valid untyped $\lambda\mathtt{ev}Q$-terms of the same sort (S or T). For example, rule $(\circ^\ell)$ actually denotes two rules: $(u \circ^\ell_T v) \circ^\ell_S w \to u \circ^\ell_T (v \circ^\ell_S w)$ and $(u \circ^\ell_S v) \circ^\ell_S w \to u \circ^\ell_S (v \circ^\ell_S w)$. We have done this so as to be able to state the rules in as little space as possible: doing otherwise would not contribute any significant information anyway. This won't be too much of a nuisance, as the S and T subscripts have little influence on how rewriting proceeds.

Some rules could have been stated as one schema instead of two: if we overlook the constraints of the form $1 \leq \ell < \mathcal{L}$, then $(\mathtt{ev}\star^\ell)$ and $(\mathtt{ev}\ \bullet^\ell)$ are $(\mathtt{ev}^\ell\star^{\mathcal{L}})$ and $(\mathtt{ev}^\ell\ \bullet^{\mathcal{L}})$ respectively, with $\mathcal{L} = \ell$. As announced, group (B) is more complicated than what we presented in Section 2.1, because it is now a variant of the $\lambda\sigma_\Uparrow$-calculus.

Moreover, notice how we have stated rule $(\mathtt{ev}\ \Uparrow^\ell)$. For simplicity, consider the case when $\ell = 1$. Had we stated it as $\mathtt{ev}^1(\Uparrow^1 u)w \to (1w) \bullet (\mathtt{ev}^1 u(\uparrow w))$, we would lose confluence. Consider indeed $\mathtt{ev}^1(\Uparrow^1 id^1)w$: by rules $(\Uparrow id^1)$ and $(\mathtt{ev}id^1)$, this reduces to w, while it reduces to $(1w) \bullet (\uparrow w)$ by using the latter formulation of $(\mathtt{ev}\ \Uparrow^1)$ and rule $(\mathtt{ev}id^1)$. This would then suggest the rule $(1w) \bullet (\uparrow w) \to w$, i.e. surjective pairing, which destroys confluence [Klo80]. This is also the main reason why $\mathtt{ev}^\ell$ is a binary operator, and not a unary one:

Level 0:

$$(Ax) \qquad \frac{}{,\ , x : \tau \vdash x : \tau}$$

$$(\Rightarrow E) \quad \frac{\begin{array}{c} , \vdash u : \tau_1 \Rightarrow \tau_2 \\ , \vdash v : \tau_1 \end{array}}{, \vdash uv : \tau_2} \qquad (\Rightarrow I) \quad \frac{,\ , x : \tau_1 \vdash u : \tau_2}{, \vdash \lambda x \cdot u : \tau_1 \Rightarrow \tau_2}$$

$$(\times E) \quad \frac{, \vdash u : \tau \times \varsigma}{\begin{array}{c} , \vdash 1u : \tau \\ (\text{resp. } , \vdash \uparrow u : \varsigma) \end{array}} \qquad (\times I) \quad \frac{\begin{array}{c} , \vdash u : \tau \\ , \vdash v : \varsigma \end{array}}{, \vdash u \bullet v : \tau \times \varsigma}$$

$$(\top I) \qquad \frac{}{, \vdash () : \top}$$

Level $\ell \geq 1$:

$$(\Box^\ell Ax_1) \quad \frac{}{, \vdash 1^\ell : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \tau \times \varsigma \overset{\Box}{\Rightarrow} \tau} \qquad (\Box^\ell Ax_2) \quad \frac{}{, \vdash id^\ell : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \varsigma \overset{\Box}{\Rightarrow} \varsigma}$$

$$(\Box^\ell W_1) \quad \frac{}{, \vdash \uparrow^\ell : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \tau \times \varsigma \overset{\Box}{\Rightarrow} \varsigma}$$

$$(\Box^\ell \Rightarrow E) \quad \frac{\begin{array}{c} , \vdash u : \overline{\varsigma^{\ell} \overset{\Box}{\Rightarrow}} \tau_1 \Rightarrow \tau_2 \\ , \vdash v : \overline{\varsigma^{\ell} \overset{\Box}{\Rightarrow}} \tau_1 \end{array}}{, \vdash u \star^\ell v : \overline{\varsigma^{\ell} \overset{\Box}{\Rightarrow}} \tau_2} \qquad (\Box^\ell \Rightarrow I) \quad \frac{, \vdash u : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \tau_1 \times \varsigma \overset{\Box}{\Rightarrow} \tau_2}{, \vdash \lambda^\ell u : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \varsigma \overset{\Box}{\Rightarrow} \tau_1 \Rightarrow \tau_2}$$

$$(\Box^\ell \Uparrow) \quad \frac{, \vdash u : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \varsigma_1' \overset{\Box}{\Rightarrow} \varsigma_2'}{, \vdash \Uparrow^\ell u : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \tau \times \varsigma_1' \overset{\Box}{\Rightarrow} \tau \times \varsigma_2'} \qquad (\Box^\ell \times I) \quad \frac{\begin{array}{c} , \vdash u : \overline{\varsigma^{\ell} \overset{\Box}{\Rightarrow}} \tau \\ , \vdash v : \overline{\varsigma^{\ell} \overset{\Box}{\Rightarrow}} \varsigma \end{array}}{, \vdash u \bullet^\ell v : \overline{\varsigma^{\ell} \overset{\Box}{\Rightarrow}} \tau \times \varsigma}$$

$$(\Box^\ell \Box E) \quad \frac{\begin{array}{c} , \vdash u : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \varsigma \overset{\Box}{\Rightarrow} \Phi \\ , \vdash w : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \varsigma \end{array}}{, \vdash \mathtt{ev}^\ell uw : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \Phi} \qquad (\Box^\ell \Box I) \quad \frac{, \vdash u : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \varsigma \overset{\Box}{\Rightarrow} \Phi}{\begin{array}{c} , \vdash Q^\ell u : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \\ \varsigma' \overset{\Box}{\Rightarrow} \varsigma \overset{\Box}{\Rightarrow} \Phi \end{array}}$$

$$(\Box^\ell Cut) \quad \frac{, \vdash u : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \varsigma_1' \overset{\Box}{\Rightarrow} \Phi \quad , \vdash v : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \varsigma_2' \overset{\Box}{\Rightarrow} \varsigma_1'}{, \vdash u \circ^\ell v : \overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}} \varsigma_2' \overset{\Box}{\Rightarrow} \Phi}$$

Figure 1: Typing rules

G-translation, from λ_{S4}-terms to $\lambda\mathsf{ev}Q$-terms:

$G(x) = x$	$G(uv) = G(u)G(v)$	$G(\lambda x \cdot u) = \lambda x \cdot G(u)$
$G(\mathsf{unbox}\ u) = \mathsf{ev}^1_T G(u)()$		
$G(\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n) = (G(u)\text{‘}[])[G(v_1)/x_1, \ldots, G(v_n)/x_n]$		

Quotation $\_\text{‘}\rho$ in an environment ρ (of cardinality n):

$x\text{‘}\rho$	$= Q^1_T x$	if $\rho(x)$ is undefined
$x\text{‘}\rho$	$= 1^1 \underbrace{\circ^1_T (\uparrow^1 \circ^1_S (\uparrow^1 \circ^1_S \ldots \circ^1_S \uparrow^1) \ldots)}_{n-1-\rho(x)\ \text{times}}$	otherwise
$()\text{‘}\rho$	$= id^1$	if $n = 0$
$()\text{‘}\rho$	$= \underbrace{\uparrow^1 \circ^1_S (\uparrow^1 \circ^1_S \ldots \circ^1_S \uparrow^1)}_{n\ \text{times}}$	if $n \neq 0$
$(1u)\text{‘}\rho$	$= 1^1 \circ^1_T u\text{‘}\rho$	
$(\uparrow u)\text{‘}\rho$	$= \uparrow^1 \circ^1_S u\text{‘}\rho$	
$(u \bullet v)\text{‘}\rho$	$= u\text{‘}\rho \bullet^1 v\text{‘}\rho$	
$(uv)\text{‘}\rho$	$= (u\text{‘}\rho) \star^1 (v\text{‘}\rho)$	
$(\lambda x \cdot u)\text{‘}\rho$	$= \lambda^1(u\text{‘}\rho[x \mapsto n])$	
$(id^\ell)\text{‘}\rho$	$= id^{\ell+1}$	
$(1^\ell)\text{‘}\rho$	$= 1^{\ell+1}$	
$(\uparrow^\ell)\text{‘}\rho$	$= \uparrow^{\ell+1}$	
$(u \bullet^\ell v)\text{‘}\rho$	$= (u\text{‘}\rho) \bullet^{\ell+1} (v\text{‘}\rho)$	
$(u \star^\ell v)\text{‘}\rho$	$= (u\text{‘}\rho) \star^{\ell+1} (v\text{‘}\rho)$	
$(\lambda^\ell u)\text{‘}\rho$	$= \lambda^{\ell+1}(u\text{‘}\rho)$	
$(\Uparrow^\ell u)\text{‘}\rho$	$= \Uparrow^{\ell+1} (u\text{‘}\rho)$	
$(\mathsf{ev}^\ell_T uw)\text{‘}\rho$	$= \mathsf{ev}^{\ell+1}_T(u\text{‘}\rho)(w\text{‘}\rho)$	
$(\mathsf{ev}^\ell_S uw)\text{‘}\rho$	$= \mathsf{ev}^{\ell+1}_S(u\text{‘}\rho)(w\text{‘}\rho)$	
$(u \circ^\ell_T v)\text{‘}\rho$	$= (u\text{‘}\rho) \circ^{\ell+1}_T (v\text{‘}\rho)$	
$(u \circ^\ell_S v)\text{‘}\rho$	$= (u\text{‘}\rho) \circ^{\ell+1}_S (v\text{‘}\rho)$	
$(Q^\ell_T u)\text{‘}\rho$	$= Q^{\ell+1}_T(u\text{‘}\rho)$	
$(Q^\ell_S u)\text{‘}\rho$	$= Q^{\ell+1}_S(u\text{‘}\rho)$	

Figure 2: Translation rules

were it unary, we would be forced to say $\mathtt{ev}^1(\Uparrow^1 u) \to \lambda w \cdot (1w) \bullet (\mathtt{ev}^1 u(\uparrow w))$, which would lead to the same problem.

Notice also how close the reduction rules for $\mathtt{ev}^\ell$ are to those of $\circ^{\ell-1}$ (in particular, observe the interaction with $\Uparrow^\ell$ in rules in groups (B) and (C)). This follows our intuition that $\circ^{\ell-1}$ is the application of a substitution, while $\mathtt{ev}^\ell$ applies a substitution to an elementary term, which it also evaluates to some term at the lower level $\ell - 1$.

To conclude on the proof-theoretical level, the $(\Box R)$ rule and the quote operator are convenient shorthands for producing quoted objects and boxed types, but this deceptively simple rule hides a full tower of interpreters for the language at hand. To put it another way, whereas execution is cut-elimination, the $\lambda\mathtt{ev}Q$ perspective is that preprocessing the language for its implementation — or compilation — is the decomposition of the problematic $(\Box R)$ rule in atomic steps.

3 Interpreting Cut Elimination in $\lambda\mathtt{ev}Q$

We first prove some properties we can expect from a well-behaved calculus: in Section 3.1, we show that we can check whether an untyped $\lambda\mathtt{ev}Q$-term is typable in polynomial time, that every typable $\lambda\mathtt{ev}Q$-term has a principal type, and that the types are preserved by rewriting. Section 3.2 is a long and rather tedious series of results that we use in Section 3.3 to prove that every valid cut-elimination step in **LS4** can be simulated by a sequence of reduction steps in the typed $\lambda\mathtt{ev}Q$-calculus.

3.1 Typing

Theorem 3.1 (Typability) *Deciding whether a given $\lambda\mathtt{ev}Q$-term is typable in the system of Figure 1 is decidable in polynomial time. Moreover, if u is typable, then it has a most general type.*

Proof: The main difficulty is the fact that we have both term types and metastack types. We recast this by saying that the algebra of formulas is order-sorted, using two disjoint sorts $\mathcal{T}$ and $\mathcal{M}$, and a subsort $\mathcal{S} \subseteq \mathcal{M}$.

Write $\Phi :: s$ to say that formula Φ has sort s, and for any n-ary type constructor f, write $f :: s_1, \ldots, s_n \rightsquigarrow s$ to say that whenever $\Phi_1 :: s_1, \ldots, \Phi_n :: s_n$, then $f(\Phi_1, \ldots, \Phi_n) :: s$. The signature of the algebra of formulas and types is given by the following declarations:

- $\Phi :: \rightsquigarrow \mathcal{T}$, for any base type Φ;
- $\Rightarrow :: \mathcal{T}, \mathcal{T} \rightsquigarrow \mathcal{T}$;
- $\overset{\Box}{\Rightarrow} :: \mathcal{S}, \mathcal{T} \rightsquigarrow \mathcal{T}$ and $\overset{\Box}{\Rightarrow} :: \mathcal{S}, \mathcal{M} \rightsquigarrow \mathcal{M}$;
- $\top :: \mathcal{S}$;
- $\times :: \mathcal{T}, \mathcal{S} \rightsquigarrow \mathcal{S}$.

We also add meta-variables α of sort either $\mathcal{T}$ or $\mathcal{M}$.

We claim that this signature is *regular*, i.e. any formula Φ has a least sort $s(\Phi)$ (for the $\subseteq$ ordering). This is clear, since every formula is either a term type or a metastack type, but not both.

We then claim that this signature is *coregular*, i.e. for every n-ary type constructor f, and for any sort s, the set of n-tuples $(s_1, \ldots, s_n)$ of sorts such that $f :: s_1, \ldots, s_n \rightsquigarrow s'$ for some $s' \subseteq s$ has at most one greatest element (for the componentwise extension of $\subseteq$ to n-tuples). This is clear for base types and for $\top$. In the $\times$ case, if s is $\mathcal{M}$ or $\mathcal{S}$, then the greatest couple of argument sorts is $(\mathcal{T}, \mathcal{S})$, otherwise the set of argument sorts is empty. In the implication case, then the greatest couple is $(\mathcal{T}, \mathcal{T})$ if s is $\mathcal{T}$, otherwise there is no possible argument sort. In the $\overset{\Box}{\Rightarrow}$ case, then the greatest couple is $(\mathcal{S}, \mathcal{T})$ if $s = \mathcal{T}$ and $(\mathcal{S}, \mathcal{M})$ if $s = \mathcal{M}$.

And the sort signature is *downward-complete*, i.e. the set of sorts smaller than two given sorts is either empty or has a greatest element: this is also clear.

Since the signature is regular, coregular and downward-complete, unification in the order-sorted algebra of formulas is *unitary*, i.e. any unification problem has at most one more general solution, and unification can be done in polynomial time [JK90].

(A) Computation rules (level 0):

(β)	$(\lambda x \cdot u)v \to u[v/x]$
$(\uparrow)$	$\uparrow (u \bullet v) \to v$
(1)	$1(u \bullet v) \to u$

(B) Computation rules (level ℓ, $\ell \geq 1$):

(β^ℓ)	$(\lambda^\ell u) \star^\ell v \to u \circ^\ell (v \bullet^\ell id^\ell)$		
$(\circ id^\ell)$	$u \circ^\ell id^\ell \to u$	$(1 \Uparrow^\ell)$	$1^\ell \circ^\ell \Uparrow^\ell u \to 1^\ell$
$(id \circ^\ell)$	$id^\ell \circ^\ell u \to u$	$(1 \Uparrow \circ^\ell)$	$1^\ell \circ^\ell (\Uparrow^\ell u \circ^\ell v) \to 1^\ell \circ^\ell v$
$(\circ^\ell)$	$(u \circ^\ell v) \circ^\ell w \to u \circ^\ell (v \circ^\ell w)$	$(\uparrow\Uparrow^\ell)$	$\uparrow^\ell \circ^\ell \Uparrow^\ell u \to u \circ^\ell \uparrow^\ell$
$(\uparrow^\ell)$	$\uparrow^\ell \circ^\ell (u \bullet^\ell v) \to v$	$(\uparrow\Uparrow \circ^\ell)$	$\uparrow^\ell \circ^\ell (\Uparrow^\ell u \circ^\ell v) \to u \circ^\ell (\uparrow^\ell \circ^\ell v)$
(1^ℓ)	$1^\ell \circ^\ell (u \bullet^\ell v) \to u$	$(\Uparrow\Uparrow^\ell)$	$\Uparrow^\ell u \circ^\ell \Uparrow^\ell v \to \Uparrow^\ell (u \circ^\ell v)$
$(\bullet^\ell)$	$(u \bullet^\ell v) \circ^\ell w \to (u \circ^\ell w) \bullet^\ell (v \circ^\ell w)$	$(\Uparrow\Uparrow \circ^\ell)$	$\Uparrow^\ell u \circ^\ell (\Uparrow^\ell v \circ^\ell w) \to \Uparrow^\ell (u \circ^\ell v) \circ^\ell w$
(λ^ℓ)	$(\lambda^\ell u) \circ^\ell w \to \lambda^\ell (u \circ^\ell \Uparrow^\ell w)$	$(\Uparrow \bullet^\ell)$	$\Uparrow^\ell u \circ^\ell (v \bullet^\ell w) \to v \bullet^\ell (u \circ^\ell w)$
$(\star^\ell)$	$(u \star^\ell v) \circ^\ell w \to (u \circ^\ell w) \star^\ell (v \circ^\ell w)$	$(\Uparrow id^\ell)$	$\Uparrow^\ell id^\ell \to id^\ell$
$(Q \circ^\ell)$	$Q^\ell u \circ^\ell v \to Q^\ell u$		

(C) Evaluation rules (level $\ell \to \ell - 1$, $\ell \geq 1$):

$(\mathtt{ev}\lambda^\ell)$	$\mathtt{ev}^\ell (\lambda^\ell u) w \to \begin{cases} \lambda x \cdot \mathtt{ev}^\ell u (x \bullet w) & \text{if } \ell = 1 \\ \lambda^{\ell-1} (\mathtt{ev}^\ell (u \circ^{\ell-1} \uparrow^{\ell-1}) (1^{\ell-1} \bullet^{\ell-1} (w \circ^{\ell-1} \uparrow^{\ell-1}))) & \text{if } \ell > 1 \end{cases}$
$(\mathtt{ev}\star^\ell)$	$\mathtt{ev}^\ell (u \star^\ell v) w \to (\mathtt{ev}^\ell u w) \star^{\ell-1} (\mathtt{ev}^\ell v w)$
$(\mathtt{ev} id^\ell)$	$\mathtt{ev}^\ell id^\ell w \to w$
$(\mathtt{ev}\circ^\ell)$	$\mathtt{ev}^\ell (u \circ^\ell v) w \to \mathtt{ev}^\ell u (\mathtt{ev}^\ell v w)$
$(\mathtt{ev} \uparrow^\ell)$	$\mathtt{ev}^\ell \uparrow^\ell w \to \begin{cases} \uparrow w & \text{if } \ell = 1 \\ \uparrow^{\ell-1} \circ^{\ell-1} w & \text{if } \ell > 1 \end{cases}$
$(\mathtt{ev} 1^\ell)$	$\mathtt{ev}^\ell 1^\ell w \to \begin{cases} 1 w & \text{if } \ell = 1 \\ 1^{\ell-1} \circ^{\ell-1} w & \text{if } \ell > 1 \end{cases}$
$(\mathtt{ev} \bullet^\ell)$	$\mathtt{ev}^\ell (u \bullet^\ell v) w \to (\mathtt{ev}^\ell u w) \bullet^{\ell-1} (\mathtt{ev}^\ell v w)$
$(\mathtt{ev} \Uparrow^\ell)$	$\mathtt{ev}^\ell (\Uparrow^\ell u)(w_1 \bullet^{\ell-1} w_2) \to w_1 \bullet^{\ell-1} (\mathtt{ev}^\ell u w_2)$
$(1\mathtt{ev} \Uparrow^\ell)$	$\begin{cases} 1(\mathtt{ev}^1 (\Uparrow^1 u) w) \to 1 w & \text{if } \ell = 1 \\ 1^{\ell-1} \circ^{\ell-1} \mathtt{ev}^\ell (\Uparrow^\ell u) w \to 1^{\ell-1} \circ^{\ell-1} w & \text{if } \ell > 1 \end{cases}$
$(\uparrow \mathtt{ev} \Uparrow^\ell)$	$\begin{cases} \uparrow (\mathtt{ev}^1 (\Uparrow^1 u) w) \to \mathtt{ev}^1 u (\uparrow w) & \text{if } \ell = 1 \\ \uparrow^{\ell-1} \circ^{\ell-1} \mathtt{ev}^\ell (\Uparrow^\ell u) w \to \mathtt{ev}^\ell u (\uparrow^{\ell-1} \circ^{\ell-1} w) & \text{if } \ell > 1 \end{cases}$
$(\mathtt{ev} \Uparrow\Uparrow^\ell)$	$\mathtt{ev}^\ell (\Uparrow^\ell u)(\mathtt{ev}^\ell (\Uparrow^\ell v) w) \to \mathtt{ev}^\ell (\Uparrow^\ell (u \circ^\ell v)) w$
$(\mathtt{ev} Q^\ell)$	$\mathtt{ev}^\ell (Q^\ell u) w \to u$

Figure 3: Reduction rules

(D) Substitution in terms at higher levels ($1 \leq \ell < \mathcal{L}$):

$(\lambda^{\mathcal{L}} \circ^{\ell})$	$(\lambda^{\mathcal{L}} u) \circ^{\ell} w \to \lambda^{\mathcal{L}}(u \circ^{\ell} w)$
$(\star^{\mathcal{L}} \circ^{\ell})$	$(u \star^{\mathcal{L}} v) \circ^{\ell} w \to (u \circ^{\ell} w) \star^{\mathcal{L}} (v \circ^{\ell} w)$
$(id^{\mathcal{L}} \circ^{\ell})$	$id^{\mathcal{L}} \circ^{\ell} w \to id^{\mathcal{L}}$
$(\circ^{\mathcal{L}} \circ^{\ell})$	$(u \circ^{\mathcal{L}} v) \circ^{\ell} w \to (u \circ^{\ell} w) \circ^{\mathcal{L}} (v \circ^{\ell} w)$
$(\uparrow^{\mathcal{L}} \circ^{\ell})$	$\uparrow^{\mathcal{L}} \circ^{\ell} w \to \uparrow^{\mathcal{L}}$
$(1^{\mathcal{L}} \circ^{\ell})$	$1^{\mathcal{L}} \circ^{\ell} w \to 1^{\mathcal{L}}$
$(\bullet^{\mathcal{L}} \circ^{\ell})$	$(u \bullet^{\mathcal{L}} v) \circ^{\ell} w \to (u \circ^{\ell} w) \bullet^{\mathcal{L}} (v \circ^{\ell} w)$
$(\Uparrow^{\mathcal{L}} \circ^{\ell})$	$(\Uparrow^{\mathcal{L}} u) \circ^{\ell} w \to \Uparrow^{\mathcal{L}} (u \circ^{\ell} w)$
$(Q^{\mathcal{L}} \circ^{\ell})$	$(Q^{\mathcal{L}} u) \circ^{\ell} w \to Q^{\mathcal{L}}(u \circ^{\ell} w)$
$(\mathtt{ev}^{\mathcal{L}} \circ^{\ell})$	$(\mathtt{ev}^{\mathcal{L}} uv) \circ^{\ell} w \to \mathtt{ev}^{\mathcal{L}}(u \circ^{\ell} w)(v \circ^{\ell} w)$

(E) Simplification of $\mathtt{ev}^{\ell}$-terms ($1 \leq \ell < \mathcal{L}$):

$(\mathtt{ev}^{\ell} \lambda^{\mathcal{L}})$	$\mathtt{ev}^{\ell}(\lambda^{\mathcal{L}} u)w \to \lambda^{\mathcal{L}-1}(\mathtt{ev}^{\ell} uw)$
$(\mathtt{ev}^{\ell} \star^{\mathcal{L}})$	$\mathtt{ev}^{\ell}(u \star^{\mathcal{L}} v)w \to (\mathtt{ev}^{\ell} uw) \star^{\mathcal{L}-1} (\mathtt{ev}^{\ell} vw)$
$(\mathtt{ev}^{\ell} id^{\mathcal{L}})$	$\mathtt{ev}^{\ell} id^{\mathcal{L}} w \to id^{\mathcal{L}-1}$
$(\mathtt{ev}^{\ell} \circ^{\mathcal{L}})$	$\mathtt{ev}^{\ell}(u \circ^{\mathcal{L}} v)w \to (\mathtt{ev}^{\ell} uw) \circ^{\mathcal{L}-1} (\mathtt{ev}^{\ell} vw)$
$(\mathtt{ev}^{\ell} \uparrow^{\mathcal{L}})$	$\mathtt{ev}^{\ell} \uparrow^{\mathcal{L}} w \to \uparrow^{\mathcal{L}-1}$
$(\mathtt{ev}^{\ell} 1^{\mathcal{L}})$	$\mathtt{ev}^{\ell} 1^{\mathcal{L}} w \to 1^{\mathcal{L}-1}$
$(\mathtt{ev}^{\ell} \bullet^{\mathcal{L}})$	$\mathtt{ev}^{\ell}(u \bullet^{\mathcal{L}} v)w \to (\mathtt{ev}^{\ell} uw) \bullet^{\mathcal{L}-1} (\mathtt{ev}^{\ell} vw)$
$(\mathtt{ev}^{\ell} \Uparrow^{\mathcal{L}})$	$\mathtt{ev}^{\ell}(\Uparrow^{\mathcal{L}} u)w \to \Uparrow^{\mathcal{L}-1} (\mathtt{ev}^{\ell} uw)$
$(\mathtt{ev}^{\ell} Q^{\mathcal{L}})$	$\mathtt{ev}^{\ell}(Q^{\mathcal{L}} u)w \to Q^{\mathcal{L}-1}(\mathtt{ev}^{\ell} uw)$
$(\mathtt{ev}^{\ell} \mathtt{ev}^{\mathcal{L}})$	$\mathtt{ev}^{\ell}(\mathtt{ev}^{\mathcal{L}} uv)w \to \mathtt{ev}^{\mathcal{L}-1}(\mathtt{ev}^{\ell} uw)(\mathtt{ev}^{\ell} vw)$

(F) Quoting ($1 \leq \ell < \mathcal{L}$):

$(Q^{\ell} \lambda^{\mathcal{L}})$	$Q^{\ell}(\lambda^{\mathcal{L}-1} u) \to \lambda^{\mathcal{L}}(Q^{\ell} u)$
$(Q^{\ell} \star^{\mathcal{L}})$	$Q^{\ell}(u \star^{\mathcal{L}-1} v) \to Q^{\ell} u \star^{\mathcal{L}} Q^{\ell} v$
$(Q^{\ell} id^{\mathcal{L}})$	$Q^{\ell} id^{\mathcal{L}-1} \to id^{\mathcal{L}}$
$(Q^{\ell} \circ^{\mathcal{L}})$	$Q^{\ell}(u \circ^{\mathcal{L}-1} v) \to Q^{\ell} u \circ^{\mathcal{L}} Q^{\ell} v$
$(Q^{\ell} \uparrow^{\mathcal{L}})$	$Q^{\ell} \uparrow^{\mathcal{L}-1} \to \uparrow^{\mathcal{L}}$
$(Q^{\ell} 1^{\mathcal{L}})$	$Q^{\ell} 1^{\mathcal{L}-1} \to 1^{\mathcal{L}}$
$(Q^{\ell} \bullet^{\mathcal{L}})$	$Q^{\ell}(u \bullet^{\mathcal{L}-1} v) \to (Q^{\ell} u) \bullet^{\mathcal{L}} (Q^{\ell} v)$
$(Q^{\ell} \Uparrow^{\mathcal{L}})$	$Q^{\ell}(\Uparrow^{\mathcal{L}-1} u) \to \Uparrow^{\mathcal{L}} (Q^{\ell} u)$
$(Q^{\ell} Q^{\mathcal{L}})$	$Q^{\ell}(Q^{\mathcal{L}-1} u) \to Q^{\mathcal{L}}(Q^{\ell} u)$
$(Q^{\ell} \mathtt{ev}^{\mathcal{L}})$	$Q^{\ell}(\mathtt{ev}^{\mathcal{L}} uw) \to \mathtt{ev}^{\mathcal{L}+1}(Q^{\ell} u)(Q^{\ell} w)$

Figure 4: Reduction rules (continued)

We can then use a simple modification of Hindley's algorithm for the simply-typed λ-calculus [Hin69] — except that unification is now order-sorted unification on the algebra of formulas. Indeed, the computation of the most general type proceeds by structural induction on u, as at most one typing rule can apply at each step, introducing meta-variables of the right sort to represent unknown formulas, and using unification to resolve constraints (as in Ax_1, Ax_2). Notice that there is no ambiguity in the rules $(\Box^\ell\Box I)$, $(\Box^\ell\Box E)$ and $(\Box^\ell Cut)$: whether Φ must be of sort $\mathcal{T}$ or $\mathcal{M}$ is given by the index, T or S, of the topmost operator of the term to type. $\Box$

The algorithm is better known as (a subset of) the typing algorithm for ML programs [Mil78], except that we don't need to generalize type variables. (The order-sorted trick is the way that equality-admitting and imperative types are handled in Standard ML [MTH90].) On the other hand, this shows that not only the λ‚‘ and the λ_{S4}-calculus, but also the $\lambda\mathtt{ev}Q$-calculus are suitable extensions of the type discipline of ML.

Theorem 3.2 (Subject Reduction) *Let u be a term of the $\lambda\mathtt{ev}Q$-calculus. If , $\vdash u : \Phi$ is derivable, and $u \longrightarrow^* v$, then , $\vdash v : \Phi$ is derivable.*

Proof: By induction on the number of rewriting steps from u to v. We need to do a case analysis on the rule used at each step, and to check that whenever the left-hand side is well-typed, the right-hand side has the same type under the same context. The summary of the types for each rule of Figure 3 is in Figures 5 and 6, where we have shown the most general typings for the left-hand side of the rules: the second column denotes the type of the assumptions, and the third column is the type of the left-hand side. We have shown on a second line what the most general typing of the right-hand side was, when different. (The level ℓ is any integer such that $\ell \geq 1$.) Also, when some types have been left as Φ, or Φ', or etc., then whether it is a term type or a metastack type is determined by the variant of the rule that we examine (i.e., the omitted indices on $\mathtt{ev}^\ell$, $\circ^\ell$ and Q^ℓ).

The only unexplained case is rule (β): there $u[v/x]$ gets some most general type τ (a term type); a simple induction on the derivation of $\ldots, x : \tau_2 \vdash u : \tau_1$ then shows that τ_1 is an instance of τ, so $u[v/x]$ has type τ_1 as well.

Similarly, the types for each rule of Figure 4 are given in Figures 7 and 8, where $1 \leq \ell < \mathcal{L}$. We have adopted the convention that any text of the form $\ldots\varsigma' \overset{\Box}{\Rightarrow} \Phi_m \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \Phi_n \overset{\Box}{\Rightarrow} \Phi\ldots$, which is already meaningful when $m \leq n$, means $\ldots\varsigma' \overset{\Box}{\Rightarrow} \Phi\ldots$ when $m = n+1$. $\Box$

Before we continue, we introduce some useful notation:

Definition 3.1 *Let n be a natural integer.*

We let pop_n^ℓ be id^ℓ when $n = 0$, and $\underbrace{\uparrow^\ell \circ^\ell(\uparrow^\ell \circ^\ell(\ldots\circ^\ell \uparrow^\ell)\ldots)}_{n \text{ times}}$ when $n \geq 1$.

We define get_n^ℓ as 1^ℓ when $n = 0$, and as $1^\ell \circ^\ell \mathsf{pop}_n^\ell$ otherwise.

Finally, for every term w, we define $\mathsf{pop}_n;^\ell w$ as w if $n = 0$, and as $\uparrow^\ell \circ^\ell(\mathsf{pop}_{n-1};^\ell w)$ if $n \geq 1$. We define $\mathsf{get}_n;^\ell w$ as $1^\ell \circ^\ell (\mathsf{pop}_n;^\ell w)$. And we define $w[\mathsf{pop}_n]^\ell$ as w if $n = 0$, and as $w \circ^\ell \mathsf{pop}_n^\ell$ otherwise.

The following says that whenever we take a λ‚‘-term u, quoting it changes its type in the expected way:

Theorem 3.3 (Quoting) *Let , be a boxed context, i.e. mapping variables to term types of the form $\varsigma \overset{\Box}{\Rightarrow} \tau$. Let u be a $\lambda\mathtt{ev}Q$-term such that the sequent:*

$$, , x_1 : \tau_1, \ldots, x_n : \tau_n \vdash u : \Phi$$

is derivable in the system of Figure 1. Then $u‘[x_1 \mapsto 0, \ldots, x_n \mapsto n-1]$ has type $\tau_n \times (\tau_{n-1} \times \ldots (\tau_1 \times \varsigma)\ldots) \overset{\Box}{\Rightarrow} \Phi$ under context , , for any stack type ς.

Proof: By structural induction on u, using the definitions of Figure 2.

If u is a variable other than $x_1, \ldots, x_n$, then $u‘[x_1 \mapsto 0, \ldots, x_n \mapsto n-1] = Q^1 u$ indeed has type $\tau_n \times (\tau_{n-1} \times \ldots (\tau_1 \times \varsigma)\ldots) \overset{\Box}{\Rightarrow} \Phi$ under context , . (In fact, any type $\varsigma' \overset{\Box}{\Rightarrow} \Phi$.) Indeed, , is a boxed context, so Φ is boxed and we can apply the typing rule for Q^1.

If u is x_i, then $u‘[x_1 \mapsto 0, \ldots, x_n \mapsto n-1] = \mathsf{get}_{n-1-i}^1$, which has type $\tau_n \times (\tau_{n-1} \times \ldots (\tau_1 \times \varsigma)\ldots) \overset{\Box}{\Rightarrow} \Phi_i$ under context , , and the claim is valid in this case, too.

Rule	Assumptions	Type
Group (A)		
(β)	$(\ldots, x:\tau_2 \vdash u:\tau_1),\ v:\tau_2$	τ_1
rhs:	$(\ldots, u[v/x]:\tau)$	τ
$(\uparrow)$	$u:\tau,\ v:\varsigma$	ς
rhs:	$v:\varsigma$	ς
(1)	$u:\tau,\ v:\varsigma$	τ
rhs:	$u:\tau$	τ
Group (B)		
(β^ℓ)	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\tau_1\times\varsigma\overset{\square}{\Rightarrow}\tau_2,\ v:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma\overset{\square}{\Rightarrow}\tau_1$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma\overset{\square}{\Rightarrow}\tau_2$
$(\circ id^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma\overset{\square}{\Rightarrow}\Phi'$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma\overset{\square}{\Rightarrow}\Phi'$
rhs:	$u:\Phi$	Φ
$(id\circ^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\varsigma_2$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\varsigma_2$
rhs:	$u:\varsigma$	ς
$(\circ^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\Phi,\ v:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_2\overset{\square}{\Rightarrow}\varsigma_3,$ $w:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\varsigma_2$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\Phi$
$(\uparrow^\ell)$	$u:\overline{{\varsigma'}^{\ell}\overset{\square}{\Rightarrow}}\tau,\ v:\overline{{\varsigma'}^{\ell}\overset{\square}{\Rightarrow}}\varsigma$	$\overline{{\varsigma'}^{\ell}\overset{\square}{\Rightarrow}}\varsigma$
rhs:	$v:\varsigma'$	ς'
(1^ℓ)	$u:\overline{{\varsigma'}^{\ell}\overset{\square}{\Rightarrow}}\tau,\ v:\overline{{\varsigma'}^{\ell}\overset{\square}{\Rightarrow}}\varsigma$	$\overline{{\varsigma'}^{\ell}\overset{\square}{\Rightarrow}}\tau$
rhs:	$u:\tau'$	τ'
$(\bullet^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\tau,\ v:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\varsigma_3,$ $w:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_4\overset{\square}{\Rightarrow}\varsigma_1$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_4\overset{\square}{\Rightarrow}\tau\times\varsigma_3$
(λ^ℓ)	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\tau_1\times\varsigma_1\overset{\square}{\Rightarrow}\tau_2,\ w:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_2\overset{\square}{\Rightarrow}\varsigma_1$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_2\overset{\square}{\Rightarrow}\tau_1\Rightarrow\tau_2$
$(\star^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\tau_1\Rightarrow\tau_2,\ v:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\tau_1,$ $w:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_2\overset{\square}{\Rightarrow}\varsigma_1$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_2\overset{\square}{\Rightarrow}\tau_2$
$(1\Uparrow^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\varsigma_2$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\tau\times\varsigma_1\overset{\square}{\Rightarrow}\tau$
rhs:		$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\tau\times\varsigma_1\overset{\square}{\Rightarrow}\tau$
$(1\Uparrow\circ^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\varsigma_2,\ v:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\tau\times\varsigma_1$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\tau$
rhs:	$v:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\tau\times\varsigma_1$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\tau$
$(\uparrow\Uparrow^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\varsigma_2$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\tau\times\varsigma_1\overset{\square}{\Rightarrow}\varsigma_2$
$(\uparrow\Uparrow\circ^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\varsigma_2,\ v:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\tau\times\varsigma_1$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\varsigma_2$
$(\Uparrow\Uparrow^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\varsigma_2,\ v:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\varsigma_1$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\tau\times\varsigma_3\overset{\square}{\Rightarrow}\tau\times\varsigma_2$
$(\Uparrow\Uparrow\circ^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\varsigma_2,\ v:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\varsigma_1$ $w:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_4\overset{\square}{\Rightarrow}\tau\times\varsigma_3$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_4\overset{\square}{\Rightarrow}\tau\times\varsigma_2$
$(\Uparrow\bullet^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\varsigma_2,\ v:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\tau$ $w:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\varsigma_1$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\tau\times\varsigma_2$
$(\Uparrow id^\ell)$		$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\tau\times\varsigma\overset{\square}{\Rightarrow}\tau\times\varsigma$
rhs:		$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma'\overset{\square}{\Rightarrow}\varsigma'$
$(Q\circ^\ell)$	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\Phi,\ v:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\varsigma_4$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\varsigma_1\overset{\square}{\Rightarrow}\Phi$
rhs:	$u:\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_1\overset{\square}{\Rightarrow}\Phi$	$\overline{{\varsigma'}^{\ell-1}\overset{\square}{\Rightarrow}}\varsigma_3\overset{\square}{\Rightarrow}\varsigma_1\overset{\square}{\Rightarrow}\Phi$

Figure 5: Checking subject reduction (part 1)

Rule	Assumptions	Type
Group (C)		
$(\mathtt{ev}\lambda^{\ell})$	$u : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau_1 \times \varsigma \overset{\square}{\Rightarrow} \tau_2,\ w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau_1 \Rightarrow \tau_2$
$(\mathtt{ev}\star^{\ell})$	$u : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1 \overset{\square}{\Rightarrow} \tau_1 \Rightarrow \tau_2,\ v : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1 \overset{\square}{\Rightarrow} \tau_1,\ w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau_2$
$(\mathtt{ev}id^{\ell})$	$w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma$
rhs:	$w : \varsigma'$	ς'
$(\mathtt{ev}\circ^{\ell})$	$u : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1 \overset{\square}{\Rightarrow} \Phi,\ v : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_3 \overset{\square}{\Rightarrow} \varsigma_1$ $w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_3$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\Phi$
$(\mathtt{ev}\ \uparrow^{\ell})$	$w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau \times \varsigma$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma$
$(\mathtt{ev}1^{\ell})$	$w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau \times \varsigma$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau$
$(\mathtt{ev}\ \bullet^{\ell})$	$u : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1 \overset{\square}{\Rightarrow} \tau,\ v : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1 \overset{\square}{\Rightarrow} \varsigma_2$ $w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau \times \varsigma_2$
$(\mathtt{ev}\ \Uparrow^{\ell})$	$u : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1 \overset{\square}{\Rightarrow} \varsigma_2,\ w_1 : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau,\ w_2 : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau \times \varsigma_2$
$(1\mathtt{ev}\ \Uparrow^{\ell})$	$u : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1 \overset{\square}{\Rightarrow} \varsigma_2,\ w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau \times \varsigma_1$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau$
rhs:	$w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau \times \varsigma_1$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau$
$(\uparrow\ \mathtt{ev}\ \Uparrow^{\ell})$	$u : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1 \overset{\square}{\Rightarrow} \varsigma_2,\ w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau \times \varsigma_1$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_2$
$(\mathtt{ev}\ \Uparrow\Uparrow^{\ell})$	$u : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1 \overset{\square}{\Rightarrow} \varsigma_2,\ v : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_3 \overset{\square}{\Rightarrow} \varsigma_1,$ $w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau \times \varsigma_3$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\tau \times \varsigma_2$
$(\mathtt{ev}Q^{\ell})$	$u : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1 \overset{\square}{\Rightarrow} \Phi,\ w : \overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_3$	$\overline{{\varsigma'}^{\ell-1} \overset{\square}{\Rightarrow}}\varsigma_1 \overset{\square}{\Rightarrow} \Phi$
rhs:	$u : \Phi'$	Φ'

Figure 6: Checking subject reduction (part 2)

Rule	Assumptions	Type
Group (D) (In all rules except the rhs of $(id^{\mathcal{L}}\circ^{\ell}), (\uparrow^{\mathcal{L}}\circ^{\ell}), (1^{\mathcal{L}}\circ^{\ell})$, $w : \overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma''_{\ell} \overset{\Box}{\Rightarrow} \varsigma'_{\ell}$ is assumed.)		
$(\lambda^{\mathcal{L}}\circ^{\ell})$	$u : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\tau_1 \times \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \tau_2$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma''_{\ell} \overset{\Box}{\Rightarrow} \varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \tau_1 \Rightarrow \tau_2$
$(\star^{\mathcal{L}}\circ^{\ell})$	$u : \overline{\varsigma'^{\mathcal{L}} \overset{\Box}{\Rightarrow}}\tau_1 \Rightarrow \tau_2$ $v : \overline{\varsigma'^{\mathcal{L}} \overset{\Box}{\Rightarrow}}\tau_1$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma''_{\ell} \overset{\Box}{\Rightarrow} \varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \tau_2$
$(id^{\mathcal{L}}\circ^{\ell})$		$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma''_{\ell} \overset{\Box}{\Rightarrow} \varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}}$
$(\circ^{\mathcal{L}}\circ^{\ell})$	$u : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\varsigma_1 \overset{\Box}{\Rightarrow} \Phi$ $v : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\varsigma_3 \overset{\Box}{\Rightarrow} \varsigma_1$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma''_{\ell} \overset{\Box}{\Rightarrow} \varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \varsigma_3 \overset{\Box}{\Rightarrow} \Phi$
$(\uparrow^{\mathcal{L}}\circ^{\ell})$		$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma''_{\ell} \overset{\Box}{\Rightarrow} \varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \tau \times \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}}$
$(1^{\mathcal{L}}\circ^{\ell})$		$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma''_{\ell} \overset{\Box}{\Rightarrow} \varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \tau \times \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \tau$
$(\bullet^{\mathcal{L}}\circ^{\ell})$	$u : \overline{\varsigma'^{\mathcal{L}} \overset{\Box}{\Rightarrow}}\tau, v : \overline{\varsigma'^{\mathcal{L}} \overset{\Box}{\Rightarrow}}\varsigma$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma''_{\ell} \overset{\Box}{\Rightarrow} \varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \tau \times \varsigma$
$(\Uparrow^{\mathcal{L}}\circ^{\ell})$	$u : \overline{\varsigma'^{\mathcal{L}} \overset{\Box}{\Rightarrow}}\varsigma$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma''_{\ell} \overset{\Box}{\Rightarrow} \varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \tau \times \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \tau \times \varsigma$
$(Q^{\mathcal{L}}\circ^{\ell})$	$u : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \Phi$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma''_{\ell} \overset{\Box}{\Rightarrow} \varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \varsigma \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \Phi$
$(\mathsf{ev}^{\mathcal{L}}\circ^{\ell})$	$u : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \Phi$ $v : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\mathcal{L}}$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma''_{\ell} \overset{\Box}{\Rightarrow} \varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \Phi$
Group (E) (In all rules except the rhs of $(\mathsf{ev}^{\ell} id^{\mathcal{L}}), (\mathsf{ev}^{\ell}\uparrow^{\mathcal{L}}), (\mathsf{ev}^{\ell} 1^{\mathcal{L}})$, $w : \overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\ell}$ is assumed.)		
$(\mathsf{ev}^{\ell}\lambda^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\tau_1 \times \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \tau_2$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \tau_1 \Rightarrow \tau_2$
$(\mathsf{ev}^{\ell}\star^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}} \overset{\Box}{\Rightarrow}}\tau_1 \Rightarrow \tau_2$ $v : \overline{\varsigma'^{\mathcal{L}} \overset{\Box}{\Rightarrow}}\tau_1$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \tau_2$
$(\mathsf{ev}^{\ell} id^{\mathcal{L}})$		$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}}$
$(\mathsf{ev}^{\ell}\circ^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\varsigma_1 \overset{\Box}{\Rightarrow} \Phi$ $v : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\varsigma_3 \overset{\Box}{\Rightarrow} \varsigma_1$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \varsigma_3 \overset{\Box}{\Rightarrow} \Phi$
$(\mathsf{ev}^{\ell}\uparrow^{\mathcal{L}})$		$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \tau \times \varsigma \overset{\Box}{\Rightarrow} \varsigma$
$(\mathsf{ev}^{\ell} 1^{\mathcal{L}})$		$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \tau \times \varsigma \overset{\Box}{\Rightarrow} \tau$
$(\mathsf{ev}^{\ell}\bullet^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}} \overset{\Box}{\Rightarrow}}\tau$ $v : \overline{\varsigma'^{\mathcal{L}} \overset{\Box}{\Rightarrow}}\varsigma$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \tau \times \varsigma$
$(\mathsf{ev}^{\ell}\Uparrow^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}} \overset{\Box}{\Rightarrow}}\varsigma$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \tau \times \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \tau \times \varsigma$
$(\mathsf{ev}^{\ell} Q^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \Phi$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \varsigma \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \Phi$
$(\mathsf{ev}^{\ell}\mathsf{ev}^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\mathcal{L}} \overset{\Box}{\Rightarrow} \Phi$ $v : \overline{\varsigma'^{\mathcal{L}-1} \overset{\Box}{\Rightarrow}}\varsigma'^{\mathcal{L}}$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma'_{\ell+1} \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \Phi$

Figure 7: Checking subject reduction (part 3)

Rule	Assumptions	Type
Group (F)		
$(Q^\ell\lambda^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}-2}\overset{\Box}{\Rightarrow}}\tau_1 \times \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \tau_2$	$\overline{\varsigma'^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma \overset{\Box}{\Rightarrow} \varsigma''_\ell \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \tau_1 \Rightarrow \tau_2$
$(Q^\ell\star^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}-1}\overset{\Box}{\Rightarrow}}\tau_1 \Rightarrow \tau_2$ $v : \overline{\varsigma'^{\mathcal{L}-1}\overset{\Box}{\Rightarrow}}\tau_1$	$\overline{\varsigma'^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma \overset{\Box}{\Rightarrow} \varsigma'_\ell \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \tau_2$
$(Q^\ell id^{\mathcal{L}})$		$\overline{\varsigma'^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma \overset{\Box}{\Rightarrow} \varsigma'_\ell \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1}$
$(Q^\ell\circ^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}-2}\overset{\Box}{\Rightarrow}}\varsigma_1 \overset{\Box}{\Rightarrow} \Phi$ $v : \overline{\varsigma'^{\mathcal{L}-2}\overset{\Box}{\Rightarrow}}\varsigma_3 \overset{\Box}{\Rightarrow} \varsigma_1$	$\overline{\varsigma'^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma_4 \overset{\Box}{\Rightarrow} \varsigma'_\ell \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-2} \overset{\Box}{\Rightarrow} \varsigma_3 \overset{\Box}{\Rightarrow} \Phi$
$(Q^{\mathcal{L}}\uparrow^\ell)$		$\overline{\varsigma'^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma_3 \overset{\Box}{\Rightarrow} \varsigma'_\ell \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-2} \overset{\Box}{\Rightarrow} \tau \times \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1}$
$(Q^{\mathcal{L}}1^\ell)$		$\overline{\varsigma'^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma_3 \overset{\Box}{\Rightarrow} \varsigma'_\ell \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-2} \overset{\Box}{\Rightarrow} \tau \times \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \tau$
$(Q^\ell\bullet^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}-1}\overset{\Box}{\Rightarrow}}\tau,\ v : \overline{\varsigma'^{\mathcal{L}-1}\overset{\Box}{\Rightarrow}}\varsigma_1$	$\overline{\varsigma'^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma_2 \overset{\Box}{\Rightarrow} \varsigma'_\ell \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \tau \times \varsigma_1$
$(Q^\ell\Uparrow^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}-1}\overset{\Box}{\Rightarrow}}\varsigma_1$	$\overline{\varsigma'^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma_2 \overset{\Box}{\Rightarrow} \varsigma'_\ell \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \tau \times \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \tau \times \varsigma_1$
$(Q^\ell Q^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}-1}\overset{\Box}{\Rightarrow}}\Phi$	$\overline{\varsigma'^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma_2 \overset{\Box}{\Rightarrow} \varsigma'_\ell \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-2} \overset{\Box}{\Rightarrow} \varsigma_3 \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \Phi$
$(Q^\ell\mathtt{ev}^{\mathcal{L}})$	$u : \overline{\varsigma'^{\mathcal{L}-1}\overset{\Box}{\Rightarrow}}\varsigma_1 \overset{\Box}{\Rightarrow} \Phi$ $w : \overline{\varsigma'^{\mathcal{L}-1}\overset{\Box}{\Rightarrow}}\varsigma_1$	$\overline{\varsigma'^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma_3 \overset{\Box}{\Rightarrow} \varsigma'_\ell \overset{\Box}{\Rightarrow} \ldots \overset{\Box}{\Rightarrow} \varsigma'_{\mathcal{L}-1} \overset{\Box}{\Rightarrow} \Phi$

Figure 8: Checking subject reduction (final)

If u is $Q^\ell v$, then Φ must be of the form $\overline{\varsigma^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma' \overset{\Box}{\Rightarrow} \varsigma'' \overset{\Box}{\Rightarrow} \Phi'''$, and we must have derived $, , x_1 : \Phi_1, \ldots, x_n : \Phi_n \vdash v : \overline{\varsigma^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma'' \overset{\Box}{\Rightarrow} \Phi'''$. Let ρ be $[x_1 \mapsto 0, \ldots, x_n \mapsto n-1]$. By induction hypothesis, we can derive $, \vdash v`\rho : \tau_n \times (\tau_{n-1} \times \ldots (\tau_1 \times \varsigma) \ldots) \overset{\Box}{\Rightarrow} \overline{\varsigma^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma'' \overset{\Box}{\Rightarrow} \Phi'''$. Then, $, \vdash Q^{\ell+1}(v`\rho) : \tau_n \times (\tau_{n-1} \times \ldots (\tau_1 \times \varsigma) \ldots) \overset{\Box}{\Rightarrow} \overline{\varsigma^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma' \overset{\Box}{\Rightarrow} \varsigma'' \overset{\Box}{\Rightarrow} \Phi'''$ is also derivable, and we conclude by noticing that $u`\rho = Q^{\ell+1}(v`\rho)$.

All other cases work similarly. We deal with the case of λ-abstractions, which are a bit delicate, since they introduce variables; then we deal with the $\circ^\ell$ case, as an example of the process for all other cases. First, λ-expressions. If:

$$, , x_1 : \tau_1, \ldots, x_n : \tau_n \vdash \lambda x \cdot u : \Phi$$

is derivable, then Φ has the form $\tau' \Rightarrow \tau''$ and:

$$, , x_1 : \tau_1, \ldots, x_n : \tau_n, x : \tau' \vdash u : \tau''$$

is derivable. So, by induction hypothesis, $, \vdash u`[x_1 \mapsto 0, \ldots, x_n \mapsto n-1, x \mapsto n] : \tau' \times (\tau_n \times (\tau_{n-1} \times \ldots (\tau_1 \times \varsigma) \ldots)) \overset{\Box}{\Rightarrow} \tau''$ is derivable. Hence, $, \vdash \lambda^1(u`[x_1 \mapsto 0, \ldots, x_n \mapsto n-1, x \mapsto n]) : \tau_n \times (\tau_{n-1} \times \ldots (\tau_1 \times \varsigma) \ldots) \overset{\Box}{\Rightarrow} \tau' \Rightarrow \tau''$ is derivable, and we conclude by noticing that the latter term is $(\lambda x \cdot u)`[x_1 \mapsto 0, \ldots, x_n \mapsto n-1]$.

All the other cases work similarly to the $\circ^\ell$ case. If:

$$, , x_1 : \tau_1, \ldots, x_n : \tau_n \vdash u \circ^\ell v : \Phi$$

is derivable, then Φ must have the form $\overline{\varsigma^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma' \overset{\Box}{\Rightarrow} \Phi'$, and:

$$, , x_1 : \tau_1, \ldots, x_n : \tau_n \vdash u : \overline{\varsigma^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma'' \overset{\Box}{\Rightarrow} \Phi'$$
$$, , x_1 : \tau_1, \ldots, x_n : \tau_n \vdash v : \overline{\varsigma^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma' \overset{\Box}{\Rightarrow} \varsigma''$$

must have been derived. By induction hypothesis, if we let ρ be $[x_1 \mapsto 0, \ldots, x_n \mapsto n-1]$ and τ^n be $\tau_n \times (\tau_{n-1} \times \ldots (\tau_1 \times \varsigma) \ldots)$, we know that $u`\rho$ has type $\tau^n \overset{\Box}{\Rightarrow} \overline{\varsigma^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma'' \overset{\Box}{\Rightarrow} \Phi'$ under context $,$ and that $v`\rho$ has type $\tau^n \overset{\Box}{\Rightarrow} \overline{\varsigma^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma' \overset{\Box}{\Rightarrow} \varsigma''$ under $,$, hence $(u`\rho)\circ^{\ell+1}(v`\rho)$, that is $(u \circ^\ell v)`\rho$, has type $\tau^n \overset{\Box}{\Rightarrow} \overline{\varsigma^{\ell-1}\overset{\Box}{\Rightarrow}}\varsigma' \overset{\Box}{\Rightarrow} \Phi'$ under context $,$. $\Box$

Hence quoting ℓ times yields formulas at the ℓth level in the sense that it is of some type $\overline{\varsigma^{\ell}\overset{\Box}{\Rightarrow}}\Phi$, where ς_1, $\ldots$, ς_ℓ are sets of assumptions from sequents that we have embedded into the formula.

3.2 Properties of Eval and Quote

Our next goal is to prove that any cut-elimination step in **LS4** can be simulated by reductions in the simply-typed $\lambda\mathtt{ev}Q$-calculus. We establish this by proving a series of lemmas, the goal of which is in fact to show that we can simulate the reduction rules of the λ_{S4}-calculus. The rest of this section uses the language of the untyped $\lambda\mathtt{ev}Q$-calculus; as already announced, we won't mention any sorts either.

First, we define what we mean by the level of a term:

Definition 3.2 (Level) *We define the* out-level, *or simply* level $\mathcal{L}(u)$ *of a* $\lambda\mathtt{ev}Q$*-term* u *as follows:*

- *The level of a variable* x *is* 0.
- *Terms of the form* $\lambda x \cdot u$, uv, $()$, $\uparrow u$, $1u$, *or* $u \bullet v$ *are at level* 0.
- *For any* ℓ, $\ell \geq 1$, $\lambda^\ell u$, $u \star^\ell v$, $\uparrow^\ell$, 1^ℓ, $u \bullet^\ell v$, id^ℓ, $u \circ^\ell v$, $\Uparrow^\ell u$ *and* $Q^\ell u$ *are at level* ℓ.
- *For any* ℓ, $\ell \geq 1$, *the level of* $\mathtt{ev}^\ell uw$ *is* $\ell - 1$.

First, we show that the G-translation respects the equivalence relation $\sim$ on λ_{S4}-terms. To show this, we temporarily come back on our decision to equate $\sim$-equivalent or α-equivalent terms. Equality is now syntactic equality until further notice.

Lemma 3.4 *Let* u *be a* $\lambda\mathtt{ev}Q$*-term, and* x *and* y *be two distinct variables, with* x *not free in* u. *Then, for every environment* ρ *of cardinality* n, $(u[x/y])^{\lq}(\rho[x \mapsto n]) = u^{\lq}(\rho[y \mapsto n])$.

Proof: We prove the more general statement that for any ρ of cardinality n, for any variables $y_1, \ldots, y_m$ not in the domain of ρ and distinct from x and y:

$$(u[x/y])^{\lq}(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m]) = u^{\lq}(\rho[y \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m])$$

This is needed because of the case of λ-expressions.

If u is a variable, then we have five cases. In the first case, u is y, and both sides of the equation are $1^1 \underbrace{\circ^1(\uparrow^1 \circ^1(\uparrow^1 \circ^1 \ldots \circ^1 \uparrow^1)\ldots)}_{m \text{ times}}$. In the second case, u is a variable y_i, $1 \leq i \leq m$, so both sides are equal to $1^1 \underbrace{\circ^1(\uparrow^1 \circ^1(\uparrow^1 \circ^1 \ldots \circ^1 \uparrow^1)\ldots)}_{m-i \text{ times}}$. In the third case, u is a variable in the domain of ρ, so both sides are equal to $1^1 \underbrace{\circ^1(\uparrow^1 \circ^1(\uparrow^1 \circ^1 \ldots \circ^1 \uparrow^1)\ldots)}_{j \text{ times}}$, with $j \geq m+1$. In the fourth case, u is variable outside the domain of ρ, and distinct from x, y, and y_i, $1 \leq i \leq m$, then both sides are equal to $Q^1 u \circ^1 \underbrace{(\uparrow^1 \circ^1(\uparrow^1 \circ^1 \ldots \circ^1 \uparrow^1)\ldots)}_{n+m+1 \text{ times}}$. In the fifth and final case, u is x; but this is impossible since x was assumed not to be free in u.

If u is a λ-abstraction $\lambda z \cdot u'$, then by induction hypothesis $(u'[x/y])^{\lq}(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m, z \mapsto n+m+1]) = (u')^{\lq}(\rho[y \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m, z \mapsto n+m+1])$; indeed, by the variable naming convention $z \not\equiv x$, so that x is not free in u'. The claim follows.

All other cases are trivial uses of the induction hypothesis on the subterms. □

Lemma 3.5 *If* u *and* v *are* α*-convertible* $\lambda\mathtt{ev}Q$*-terms, then* $u^{\lq}\rho = v^{\lq}\rho$ *for every environment* ρ.

Proof: By structural induction on u (or equivalently, v). The only difficult case is when $u = \lambda x \cdot u'$; then $v = \lambda y \cdot v'$. If $x = y$, then u' is α-convertible to v', so we can apply the induction hypothesis: $u'^{\lq}(\rho[x \mapsto n]) = v'^{\lq}(\rho[x \mapsto n])$, where n is the cardinality of ρ. If $x \neq y$, then x is not free in v' (otherwise, x would be free in v, hence in u since u and v are α-convertible; this cannot happen, by the variable naming convention) and $v'[x/y]$ is α-convertible to u', so again we apply the induction hypothesis: $u'^{\lq}(\rho[x \mapsto n]) = (v'[x/y])^{\lq}(\rho[x \mapsto n])$, where n is the cardinality of ρ. By Lemma 3.4, and since x is not free in v', $(v'[x/y])^{\lq}(\rho[x \mapsto n]) = v'^{\lq}(\rho[y \mapsto n])$. In both cases, $\lambda^1(u'^{\lq}(\rho[x \mapsto n])) = \lambda^1(v'^{\lq}(\rho[y \mapsto n]))$, i.e., $(\lambda x \cdot u')^{\lq}\rho = (\lambda y \cdot v')^{\lq}\rho$. □

Lemma 3.6 *Let u be a $\lambda\mathtt{ev}Q$-term, with free variables $x_1, \ldots, x_m$, and let $x'_1, \ldots, x'_m$ be pairwise distinct variables. Then:*

(i) $\mathrm{fv}(u^{\lq}[]) = \mathrm{fv}(u)$;

(ii) $(u^{\lq}[])[x'_1/x_1, \ldots, x'_m/x_m] = (u[x'_1/x_1, \ldots, x'_m/x_m])^{\lq}[]$.

Proof: We prove by structural induction on u the more general result (i'):

$$\mathrm{fv}(u^{\lq}[y_1 \mapsto 0, \ldots, y_n \mapsto n-1]) = \mathrm{fv}(u) \setminus \{y_1, \ldots, y_n\}$$

and (ii'):

$$\begin{aligned} &(u^{\lq}[y_1 \mapsto 0, \ldots, y_n \mapsto n-1])[x'_1/x_1, \ldots, x'_m/x_m] \\ &= (u[x'_1/x_1, \ldots, x'_m/x_m])^{\lq}[y_1 \mapsto 0, \ldots, y_n \mapsto n-1] \end{aligned}$$

where $x_1, \ldots, x_m, y_1, \ldots, y_n$ are pairwise distinct and contain all the free variables of u.

If u is a variable, then either u is one of the y_i's, $1 \leq i \leq n$, so that $u^{\lq}[y_1 \mapsto 0, \ldots, y_n \mapsto n-1]$ is $1^1 \underbrace{\circ^1(\uparrow^1 \circ^1(\uparrow^1 \circ^1 \ldots \circ^1 \uparrow^1)\ldots)}_{n-i \text{ times}}$, which is closed (so (i') follows); since $y_i[x'_1/x_1, \ldots, x'_m/x_m] = y_i$, (ii') holds, too. If u is not one of the y_i's, the only free variable in $u^{\lq}[y_1 \mapsto 0, \ldots, y_n \mapsto n-1]$ is u itself, which is not an y_i, so (i') holds; then, u must be some x_i, $1 \leq i \leq m$, so:

$$\begin{aligned} &(u^{\lq}[y_1 \mapsto 0, \ldots, y_n \mapsto n-1])[x'_1/x_1, \ldots, x'_m/x_m] \\ &= Q^1 x_i \underbrace{\circ^1(\uparrow^1 \circ^1(\uparrow^1 \circ^1 \ldots \circ^1 \uparrow^1)\ldots)}_{n \text{ times}}[x'_1/x_1, \ldots, x'_m/x_m] \\ &= Q^1 x'_i \underbrace{\circ^1(\uparrow^1 \circ^1(\uparrow^1 \circ^1 \ldots \circ^1 \uparrow^1)\ldots)}_{n \text{ times}} \\ &= Q^1(x_i[x'_1/x_1, \ldots, x'_m/x_m]) \underbrace{\circ^1(\uparrow^1 \circ^1(\uparrow^1 \circ^1 \ldots \circ^1 \uparrow^1)\ldots)}_{n \text{ times}} \\ &= (x_i[x'_1/x_1, \ldots, x'_m/x_m])^{\lq}[y_1 \mapsto 0, \ldots, y_n \mapsto n-1] \end{aligned}$$

so (ii') follows.

If u is a λ-abstraction $\lambda z \cdot u'$, then by induction hypothesis, part (i'), $\mathrm{fv}(u'^{\lq}[y_1 \mapsto 0, \ldots, y_n \mapsto n-1, z \mapsto n]) = \mathrm{fv}(u') \setminus \{y_1, \ldots, y_n, z\}$. But the former is just $\mathrm{fv}(u^{\lq}[y_1 \mapsto 0, \ldots, y_n \mapsto n-1]$, and the latter is just $\mathrm{fv}(u) \setminus \{y_1, \ldots, y_n\}$, so that (i') follows. Similarly, by induction hypothesis, part (ii'), (ii') follows.

All other cases are trivial uses of the induction hypothesis. □

Lemma 3.7 *Let u be a λ_{S4}-term with free variables $x_1, \ldots, x_m$, and let $x'_1, \ldots, x'_m$ be pairwise distinct variables. Then:*

(i) $\mathrm{fv}(G(u)) = \mathrm{fv}(u)$;

(ii) $G(u)[x'_1/x_1, \ldots, x'_m/x_m] = G(u[x'_1/x_1, \ldots, x'_m/x_m])$.

Proof: By structural induction on u, using Lemma 3.6 when u is a box term. □

Lemma 3.8 *If u and v are α-convertible λ_{S4}-terms, then $G(u)$ and $G(v)$ are α-convertible.*

Proof: By structural induction on u (resp. v). The only difficult cases are:

- When u is an abstraction $\lambda x \cdot u'$; then $v = \lambda y \cdot v'$, and u' and $v'[x/y]$ are α-convertible. By induction hypothesis, $G(u')$ and $G(v'[x/y])$ are α-convertible, hence $G(u) = \lambda x \cdot G(u')$ and $G(v) = \lambda y \cdot G(v') = \lambda y \cdot G(v'[x/y])[y/x]$ are α-convertible. Indeed, $G(v') = G(v'[x/y])[y/x]$ follows from Lemma 3.7 (ii).

- When u is a box term:
  $$\mathsf{box}\ u'\ \mathsf{with}\ u_1, \ldots, u_n\ \mathsf{for}\ x_1, \ldots, x_n$$
  Then:
  $$v = \mathsf{box}\ v'\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ y_1, \ldots, y_n$$
  where u_i and v_i are α-convertible for every i, $1 \leq i \leq n$, and u' and $v'[x_1/y_1, \ldots, x_n/y_n]$ are α-convertible. By induction hypothesis, $G(u')$ and $G(v'[x_1/y_1, \ldots, x_n/y_n])$ are α-convertible, hence $(G(u'))^{\lq}[] = (G(v'[x_1/y_1, \ldots, x_n/y_n]))^{\lq}[]$ by Lemma 3.5. By induction hypothesis again, $G(u_i)$ and $G(v_i)$ are α-convertible for every i, $1 \leq i \leq n$, so $G(u) = ((G(u'))^{\lq}[])[G(u_1)/x_1, \ldots, G(u_n)/x_n]$ and $((G(v'[x_1/y_1, \ldots, x_n/y_n]))^{\lq}[])[G(v_1)/x_1, \ldots, G(v_n)/x_n]$ are α-convertible. The latter is, in turn, α-convertible to $(((G(v'[x_1/y_1, \ldots, x_n/y_n]))^{\lq}[])[y_1/x_1, \ldots, y_n/x_n])[G(v_1)/y_1, \ldots, G(v_n)/y_n]$. By Lemma 3.6 (ii), this is $((G(v'[x_1/y_1, \ldots, x_n/y_n][y_1/x_1, \ldots, y_n/x_n]))^{\lq}[])[G(v_1)/y_1, \ldots, G(v_n)/y_n]$. Because of the variable naming convention, for each i, $1 \leq i \leq n$, either $x_i = y_i$ or x_i is not free in v', so that the latter is exactly v.

□

Theorem 3.9 ($\sim$) *Let u and v be two λ_{S4}-terms (*not *classes modulo $\sim$-equivalence). If $u \sim v$, then $G(u)$ and $G(v)$ are α-convertible.*

Proof: Recall that $\sim$ is the smallest congruence containing α and the commuting conversion. We prove the claim by induction on the length of a proof of $u \sim v$ using the axioms of reflexivity, symmetry, transitivity and congruence (stability by context application) for $\sim$, and the rules of α-conversion and commuting conversion.

If $u = v$ (reflexivity), the claim is obvious.

If $u \sim v$ because we have a shorter proof of $v \sim u$ (symmetry), this is again obvious.

If $u \sim v$ follows from shorter proofs of $u \sim w$ and $w \sim v$ for some term w (transitivity), this is again obvious.

If $u \sim v$ follows from a congruence argument, i.e. for example if $u = u_1 u_2$, $v = v_1 v_2$ and we have shorter proofs of $u_1 \sim v_1$ and $u_2 \sim v_2$, then this is obvious, except in the case where u and v are quotations. Then, $u = \mathsf{box}\ u'\ \mathsf{with}\ u_1, \ldots, u_n\ \mathsf{for}\ x_1, \ldots, x_n$, $v = \mathsf{box}\ v'\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n$, and we have strictly shorter proofs of $u' \sim v'$, $u_1 \sim v_1$, $\ldots$, $u_n \sim v_n$. By induction hypothesis, $G(u')$ and $G(v')$ are α-equivalent. By Lemma 3.5, $(G(u'))^{\lq}[] = (G(v'))^{\lq}[]$, so $((G(u'))^{\lq}[])[G(u_1)/x_1, \ldots, G(u_n)/x_n]$ and $((G(v'))^{\lq}[])[G(v_1)/x_1, \ldots, G(v_n)/x_n]$ are α-convertible, by using the induction hypothesis n times. The result follows.

If $u \sim v$ because u and v are α-convertible, then by Lemma 3.8, $G(u)$ and $G(v)$ are α-convertible.

Finally, if $u \sim v$ comes from the fact that
$$u = \mathsf{box}\ u'\ \mathsf{with}\ u_1, \ldots, u_n\ \mathsf{for}\ x_1, \ldots, x_n$$
and
$$v = \mathsf{box}\ u'\ \mathsf{with}\ u_{\pi(1)}, \ldots, u_{\pi(n)}\ \mathsf{for}\ x_{\pi(1)}, \ldots, x_{\pi(n)}$$
for some permutation π, then
$$G(u) = (u'^{\lq}[])[G(u_1)/x_1, \ldots, G(u_n)/x_n]$$
and
$$G(v) = (u'^{\lq}[])[G(u_{\pi(1)})/x_{\pi(1)}, \ldots, G(u_{\pi(n)})/x_{\pi(n)}]$$
But these two terms are the same, since the substitutions we apply are the same. □

Therefore, the G transformation is well-defined on $\sim$-equivalence classes of λ_{S4}-terms. Notice in particular how the commuting conversion of λ_{S4} corresponds to the fact that (implicit, usual) substitutions can be written in any order we wish.

We now resume reasoning modulo equivalences of the various systems: $\sim$-equivalence for λ_{S4}, α-equivalence for $\lambda \mathsf{ev} Q$.

We then show that rules (gc) and (ctract) are interpreted as equalities in $\lambda \mathsf{ev} Q$.

Lemma 3.10 (gc) *If $x_i \notin \mathrm{fv}(u)$, then:*

$$\begin{aligned}&G(\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n)\\ &= G(\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)\end{aligned}$$

Proof: Trivial consequence of Lemma 3.7. □

Lemma 3.11 (ctract) *When $v_i = v_j$ for some $i \neq j$:*

$$\begin{aligned}&G(\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_i, \ldots, v_j, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n)\\ &= G(\mathsf{box}\ u[x_i/x_j]\ \mathsf{with}\ v_1, \ldots, v_i, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_i, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)\end{aligned}$$

Proof: Trivial. □

Although we won't need the following notions right away (we could be less precise, and use $\longrightarrow$ intead of $\xrightarrow{\rq_\ell}$), we shall need them in Sections 5 and in Part III. In particular, we shall need to identify notions of residuals of rules in groups (C), (E) or (F). To do this, we enrich our language with operators $\mathbf{ev}^\ell_*$ and Q^ℓ_*, where the star marks the residual. We let (C^ℓ_*), (E^ℓ_*) be the set of rules obtained from (C) (at level exactly ℓ) and (E) (at levels ℓ exactly, and $\mathcal{L}$, for all $\mathcal{L} > \ell$) by replacing textually all occurrences of "$\mathbf{ev}^\ell$" by "$\mathbf{ev}^\ell_*$" (for instance, the rules ($\mathbf{ev}^\ell\mathbf{ev}^\mathcal{L}$) for all $\mathcal{L} > \ell$ become rules ($\mathbf{ev}^\ell_*\mathbf{ev}^\mathcal{L}$: $\mathbf{ev}^\ell_*(\mathbf{ev}^\mathcal{L} uv)w \to \mathbf{ev}^{\mathcal{L}-1}(\mathbf{ev}^\ell_* uw)(\mathbf{ev}^\ell_* vw)$). We let ($\mathrm{F}^\ell_*$) be the set of rules obtained from (F) (at levels ℓ exactly, and $\mathcal{L}$, for all $\mathcal{L} > \ell$) by replacing textually all occurrences of "Q^ℓ" by "Q^ℓ_*" (for instance, the rules ($Q^\ell Q^\mathcal{L}$) for all $\mathcal{L} > \ell$ become rules ($Q^\ell_* Q^\mathcal{L}$): $Q^\ell_*(Q^{\mathcal{L}-1}u) \to Q^\mathcal{L}(Q^\ell_* u)$). In practice, we will adopt the convention that the star is a mere decoration of certain occurrences of $\mathbf{ev}^\ell$ or of Q^ℓ.

Definition 3.3 ($\circ_\ell$, $\lq_\ell$, $\rq_\ell$) *For every $\ell \geq 1$, let $\circ_\ell$ be the set of all rules in group (B) at level ℓ except (β^ℓ), plus all rules in group (D) at levels ℓ and $\mathcal{L}$ for all integers $\mathcal{L} > \ell$. Let $\xrightarrow{\circ_\ell}$ be the associated reduction relation, $\xrightarrow{\circ_\ell}^+$ its transitive closure and $\xrightarrow{\circ_\ell}^*$ its reflexive transitive closure.*

Let $\lq_\ell$ be the set of rules in group (F^ℓ_*). Let $\xrightarrow{\overset{\lq}{\ell}}$ be the associated reduction relation, $\xrightarrow{\overset{\lq}{\ell}}^+$ its transitive closure, and $\xrightarrow{\overset{\lq}{\ell}}^*$ its reflexive transitive closure.

Let $\rq_\ell$ be the set of:

- *all rules in group (C^ℓ_*),*
- *all rules in group (E^ℓ_*),*
- *if $\ell = 1$, then rules $(\uparrow)$, (1) and $(\circ^1)$,*
- *if $\ell > 1$, then all rules of $\circ_{\ell-1}$ and rule $(\circ^\ell)$.*

Let $\xrightarrow{\rq_\ell}$ be the associated rewrite relation, $\xrightarrow{\rq_\ell}^+$ its transitive closure and $\xrightarrow{\rq_\ell}^*$ its reflexive transitive closure.

We have the expected fact that $\mathbf{ev}^1$ operates as a left inverse to $\_\lq$. This will be used to show that rule (unbox) can be simulated in $\lambda\mathbf{ev}Q$.

Theorem 3.12 (Evaluation) *Let ρ_n be the environment $[x_1 \mapsto 0, x_2 \mapsto 1, \ldots, x_n \mapsto n-1]$, and u, w_0, w_1, $\ldots$, w_n be $n+2$ $\lambda\mathbf{ev}Q$-terms.*

Then:

$$\mathbf{ev}^1(u\lq\rho_n)(w_n \bullet (w_{n-1} \bullet \ldots \bullet (w_1 \bullet w_0)\ldots)) \longrightarrow^+ u[w_1/x_1, w_2/x_2, \ldots, w_n/x_n]$$

More precisely:

$$\mathbf{ev}^1_*(u\lq\rho_n)(w_n \bullet (w_{n-1} \bullet \ldots \bullet (w_1 \bullet w_0)\ldots))\xrightarrow{\rq_1}^+ u[w_1/x_1, w_2/x_2, \ldots, w_n/x_n]$$

Proof: By structural induction on u, using the definition by structural recursion of $u\lq\rho_n$.

Most cases are boring, and merely propagate the induction from the subterms to the term. We examine the cases of variables, of λ-abstractions and of λ^ℓ-abstractions, which are the most interesting. Let $\overline{w^i}$ be the stack $w_i \bullet (w_{i-1} \bullet \ldots \bullet (w_1 \bullet w_0)\ldots)$, and let σ_i be the substitution $[w_1/x_1, \ldots, w_i/x_i]$.

If u is a variable x other than the x_i's, then $x`\rho_n = Q^1 x$. By rule $(\mathtt{ev}_* Q^1)$, $\mathtt{ev}^1_*(x`\rho_n)\overline{w^n} \xrightarrow{'1} x = u\sigma_n$.

If u is one of the x_i's, then $x_i`\rho_n = 1^1 \underbrace{\circ^1(\uparrow^1 \circ^1(\uparrow^1 \circ^1 \ldots \circ^1 \uparrow^1)\ldots)}_{n-i \text{ times}}$, so:

$$
\begin{array}{ll}
\mathtt{ev}^1_*(x_i`\rho_n)\overline{w^n} & \\
\xrightarrow{'1}{}^* \mathtt{ev}^1_* 1^1 \underbrace{(\mathtt{ev}^1_* \uparrow^1 (\ldots \mathtt{ev}^1_* \uparrow^1}_{n-i \text{ times}} \overline{w^n})\ldots) & \text{by } (\mathtt{ev}_* \circ^1)\ n-i \text{ times} \\
\xrightarrow{'1}{}^+ 1 \underbrace{(\uparrow (\ldots \uparrow}_{n-i \text{ times}} \overline{w^n})\ldots) & \text{by } (\mathtt{ev}_* 1^1) \text{ once and } (\mathtt{ev}_* \uparrow^1)\ n-i \text{ times} \\
\xrightarrow{'1} 1 \underbrace{(\uparrow (\ldots \uparrow}_{n-i-1 \text{ times}} \overline{w^{n-1}})\ldots) & \text{by } (\uparrow) \\
\xrightarrow{'1} 1 \underbrace{(\uparrow (\ldots \uparrow}_{n-i-2 \text{ times}} \overline{w^{n-2}})\ldots) & \text{by } (\uparrow) \\
\ldots & \\
\xrightarrow{'1} 1 \overline{w^i} & \\
\xrightarrow{'1} w_i & \text{by } (1) \\
= u\sigma_n &
\end{array}
$$

If u is $\lambda x_{n+1} \cdot v$, then $\mathtt{ev}^1_*(u`\rho_n)\overline{w^n}$ is $\mathtt{ev}^1_*(\lambda^1(v`\rho_{n+1}))\overline{w^n}$, where $\rho_{n+1} = \rho_n[x_{n+1} \mapsto n]$. This rewrites by rule $(\mathtt{ev}_*\lambda^1)$ to $\lambda x \cdot \mathtt{ev}^1_*(v`\rho_{n+1})(x \bullet \overline{w^n})$. By induction hypothesis, this rewrites in at least one step to $\lambda x \cdot v[w_1/x_1, \ldots, w_n/x_n, x/x_{n+1}]$, i.e. $u\sigma_n$.

If u is $\lambda^\ell v$, $\ell \geq 1$, then $u`\rho_n = \lambda^{\ell+1}(v`\rho_n)$. Then $\mathtt{ev}^1_*(u`\rho_n)\overline{w^n}$ rewrites to $\lambda^\ell(\mathtt{ev}^1_*(v`\rho_n)\overline{w^n})$ by rule $(\mathtt{ev}^1_*\lambda^{\ell+1})$, then to $\lambda^\ell(v\sigma_n)$, i.e. $u\sigma_n$, by induction hypothesis.

All other cases work in exactly the same way as the latter. □

Lemma 3.13 *For any λ_{S4}-terms u, $v_1, \ldots, v_n$, we have:*

$$G(u[v_1/x_1, \ldots, v_n/x_n]) = G(u)[G(v_1)/G(x_1), \ldots, G(v_n)/G(x_n)]$$

Proof: By structural induction on u. This is clear for variables, and for all cases but box terms. If $u = \mathsf{box}\ u'$ with $w_1, \ldots, w_m$ for $y_1, \ldots, y_m$, this follows by the induction hypothesis on $w_1, \ldots, w_m$, because by definition no x_i can be free in u', nor in $G(u')`[]$ by Lemmas 3.7 and 3.6. □

Corollary 3.14 (unbox) *For any λ_{S4}-terms u, $v_1, \ldots, v_n$, we have:*

$$G(\mathsf{unbox}\ (\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n)) \longrightarrow^+ G(u[v_1/x_1, \ldots, v_n/x_n])$$

Proof:

$$
\begin{array}{ll}
G(\mathsf{unbox}\ (\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n)) & \\
= \mathtt{ev}^1((G(u)`[])[G(v_1)/x_1, \ldots, G(v_n)/x_n])() & \text{by definition}
\end{array}
$$

Now, by Theorem 3.12, and erasing all star marks, $\mathtt{ev}^1(G(u)`[])()$ reduces to $G(u)$. By induction on the length of the latter derivation, $(\mathtt{ev}^1(G(u)`[])())[G(v_1)/x_1, \ldots, G(v_n)/x_n]$ reduces to $G(u)[G(v_1)/x_1, \ldots, G(v_n)/x_n]$. By Lemma 3.13, the latter is exactly $G(u[v_1/x_1, \ldots, v_n/x_n])$. □

Before we prove Theorem 3.18 (below), we first have to prove a few technical lemmas:

Lemma 3.15 *For every $p \in \mathbb{N}$, for every term w, we let $\Uparrow^{\ell p}(w)$ be w if $p = 0$, and $\Uparrow^\ell (\Uparrow^{\ell p-1}(w))$ otherwise. Then:*

(i) for every $n \in \mathbb{N}$, for every term w, $\mathsf{pop}^\ell_n \circ^\ell w \longrightarrow^* \mathsf{pop}_n ;^\ell w$; (using only rules $(id\circ^\ell)$ and $(\circ^\ell)$)

(ii) for every $n, p \in \mathbb{N}$, for every term w, $\mathsf{pop}_{n+p} ;^\ell \Uparrow^{\ell p}(w) \longrightarrow^* \mathsf{pop}_n ;^\ell (w[\mathsf{pop}_p]^\ell)$; (using only rules $(\uparrow\Uparrow^\ell)$, $(\uparrow\Uparrow \circ^\ell)$ and $(\circ^\ell)$)

(iii) *for every $m, n, p \in \mathbb{N}$, $\mathsf{pop}_{n+p};^{\ell} \Uparrow^{\ell p}(\mathsf{pop}^{\ell}_{m}) \longrightarrow^{*} \mathsf{pop}^{\ell}_{m+n+p}$; (using only rules $(\uparrow\Uparrow^{\ell})$, $(\uparrow\Uparrow \circ^{\ell})$, $(id\circ^{\ell})$ and $(\circ^{\ell})$)*

(iv) *for every $n, p \in \mathbb{N}$, with $n < p$, for every term w, $\mathsf{pop}_{n};^{\ell} \Uparrow^{\ell p}(w) \longrightarrow^{*} \Uparrow^{\ell p-n}(w)[\mathsf{pop}_{n}]^{\ell}$; (using only rules $(\uparrow\Uparrow^{\ell})$, $(\uparrow\Uparrow \circ^{\ell})$ and $(\circ^{\ell})$)*

(v) *for every $n, p \in \mathbb{N}$, with $n < p$, for every term w, $\mathsf{get}_{n};^{\ell} \Uparrow^{\ell p}(w) \longrightarrow^{*} \mathsf{get}^{\ell}_{n}$. (using only rules $(\uparrow\Uparrow^{\ell})$, $(\uparrow\Uparrow \circ^{\ell})$, $(1 \Uparrow^{\ell})$ and $(\circ^{\ell})$)*

Proof:

(i) If $n = 0$, then this follows by using $(id\circ^{\ell})$. If $n \geq 1$, this follows by applying rule $(\circ^{\ell})$ $n-1$ times.

(ii) If $p = 0$, then the left-hand side equals the right-hand side. If $p \neq 0$, then:

$$\begin{array}{ll}
\mathsf{pop}_{n+p};^{\ell} \Uparrow^{\ell p}(w) & \\
= \underbrace{\uparrow^{\ell} \circ^{\ell}(\uparrow^{\ell} \circ^{\ell} \ldots (\uparrow^{\ell} \circ^{\ell}}_{n+p \text{ times}}(\Uparrow^{\ell p}(w))) \ldots) & \\
\longrightarrow \underbrace{\uparrow^{\ell} \circ^{\ell}(\uparrow^{\ell} \circ^{\ell} \ldots (\uparrow^{\ell} \circ^{\ell}}_{n+p-1 \text{ times}}(\Uparrow^{\ell p-1}(w) \circ^{\ell} \uparrow^{\ell})) \ldots) & \text{by } (\uparrow\Uparrow^{\ell}) \\
\longrightarrow \underbrace{\uparrow^{\ell} \circ^{\ell}(\uparrow^{\ell} \circ^{\ell} \ldots (\uparrow^{\ell} \circ^{\ell}}_{n+p-2 \text{ times}}(\Uparrow^{\ell p-2}(w) \circ^{\ell} (\uparrow^{\ell} \circ^{\ell} \uparrow^{\ell}))) \ldots) & \text{by } (\uparrow\Uparrow \circ^{\ell}) \\
\longrightarrow^{+} \underbrace{\uparrow^{\ell} \circ^{\ell}(\uparrow^{\ell} \circ^{\ell} \ldots (\uparrow^{\ell} \circ^{\ell}}_{n+p-3 \text{ times}}(\Uparrow^{\ell p-3}(w) \circ^{\ell} \mathsf{pop}^{\ell}_{3})) \ldots) & \text{by } (\uparrow\Uparrow \circ^{\ell}),\ (\circ^{\ell}) \text{ twice} \\
\ldots & \ldots \\
\longrightarrow^{+} \underbrace{\uparrow^{\ell} \circ^{\ell}(\uparrow^{\ell} \circ^{\ell} \ldots (\uparrow^{\ell} \circ^{\ell}}_{n \text{ times}}(w \circ^{\ell} \mathsf{pop}^{\ell}_{p})) \ldots) & \text{by } (\uparrow\Uparrow \circ^{\ell}),\ (\circ^{\ell})\ p-1 \text{ times} \\
= \mathsf{pop}_{n};^{\ell} (w[\mathsf{pop}_{p}]\ell) &
\end{array}$$

where the steps that use $(\uparrow\Uparrow \circ^{\ell})$ are skipped if $p = 1$.

(iii) By (ii), $\mathsf{pop}_{n+p};^{\ell} \Uparrow^{\ell p}(\mathsf{pop}^{\ell}_{m}) \longrightarrow^{*} \mathsf{pop}_{n};^{\ell} (\mathsf{pop}^{\ell}_{m} \circ^{\ell} \mathsf{pop}^{\ell}_{p})$. By (i), this reduces to $\mathsf{pop}_{n};^{\ell} (\mathsf{pop}_{m};^{\ell} \mathsf{pop}^{\ell}_{p})$, which is exactly $\mathsf{pop}^{\ell}_{m+n+p}$.

(iv) If $n = 0$, then the left-hand side is equal to the right-hand side. If $n \geq 1$, then:

$$\begin{array}{ll}
\mathsf{pop}_{n};^{\ell} \Uparrow^{\ell p}(w) & \\
= \underbrace{\uparrow^{\ell} \circ^{\ell}(\uparrow^{\ell} \circ^{\ell} \ldots (\uparrow^{\ell} \circ^{\ell}}_{n \text{ times}}(\Uparrow^{\ell p}(w))) \ldots) & \\
\longrightarrow \underbrace{\uparrow^{\ell} \circ^{\ell}(\uparrow^{\ell} \circ^{\ell} \ldots (\uparrow^{\ell} \circ^{\ell}}_{n-1 \text{ times}}(\Uparrow^{\ell p-1}(w) \circ^{\ell} \uparrow^{\ell})) \ldots) & \text{by } (\uparrow\Uparrow^{1}) \\
\longrightarrow^{+} \underbrace{\uparrow^{\ell} \circ^{\ell}(\uparrow^{\ell} \circ^{\ell} \ldots (\uparrow^{\ell} \circ^{\ell}}_{n-2 \text{ times}}(\Uparrow^{\ell p-2}(w) \circ^{\ell} (\uparrow^{\ell} \circ^{\ell} \uparrow^{\ell}))) \ldots) & \text{by } (\uparrow\Uparrow \circ^{\ell}),\ (\circ^{\ell}) \\
\longrightarrow^{+} \underbrace{\uparrow^{\ell} \circ^{\ell}(\uparrow^{\ell} \circ^{\ell} \ldots (\uparrow^{\ell} \circ^{\ell}}_{n-3 \text{ times}}(\Uparrow^{\ell p-3}(w) \circ^{\ell} \mathsf{pop}^{\ell}_{3})) \ldots) & \text{by } (\uparrow\Uparrow \circ^{\ell}),\ (\circ^{\ell}) \text{ twice} \\
\ldots & \ldots \\
\longrightarrow^{+} \Uparrow^{\ell p-n}(w) \circ^{\ell} \mathsf{pop}^{\ell}_{n} & \text{by } (\uparrow\Uparrow \circ^{\ell}),\ (\circ^{\ell})\ n-1 \text{ times} \\
= \Uparrow^{\ell p-n}(w)[\mathsf{pop}_{n}]^{\ell} &
\end{array}$$

where the steps that use $(\uparrow\Uparrow \circ^{\ell})$ are skipped if $n = 1$.

(v) $\mathsf{get}_{n};^{\ell} \Uparrow^{\ell p}(w) = 1^{\ell} \circ^{\ell} (\mathsf{pop}_{n};^{\ell} \Uparrow^{\ell p}(w))$ reduces to $1^{\ell} \circ^{\ell} (\Uparrow^{\ell p-n}(w)[\mathsf{pop}_{n}]^{\ell})$ by (iv). If $n = 0$, this reduces to $1^{\ell} = \mathsf{get}^{\ell}_{0}$ by $(1 \Uparrow^{\ell})$. If $n \geq 1$, this reduces to $1^{\ell} \circ^{\ell} \mathsf{pop}^{\ell}_{n} = \mathsf{get}^{\ell}_{n}$ by $(1 \Uparrow \circ^{\ell})$.

$\square$

Definition 3.4 **(Σ)** *Let Σ be the set of all rules but (β), (β^ℓ), $\ell \geq 1$, $\xrightarrow{\Sigma}$ be the associated reduction relation, $\xrightarrow{\Sigma}^+$ its transitive closure and $\xrightarrow{\Sigma}^*$ its reflexive transitive closure.*

Lemma 3.16 *Let v be a $\lambda\mathsf{ev}Q$-term, $m \in \mathbb{N}$, and ρ be an environment of cardinality n. Then, for any variables $y_1, \ldots, y_m$ outside the domain of ρ and not free in v:*

$$(v‘\rho)[\mathsf{pop}_m]^1 \xrightarrow{\circ_1}^+ v‘(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1])$$

Proof: The claim is obvious when $m = 0$. Otherwise, we prove it by structural induction on v. In fact, we prove the following more general statement (this is necessary because of λ-abstractions, see later), that for any $m \geq 1$ and for any $p \in \mathbb{N}$:

$$\begin{aligned}&v‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m)\\ &\xrightarrow{\circ_1}^+ v‘(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n+m, \ldots, x_p \mapsto n+m+p-1])\end{aligned}$$

for every $p \in \mathbb{N}$ and every variables $x_1, \ldots, x_p$, where no y_i is free in v.

If v is at level $\mathcal{L}'$, $\mathcal{L}' \geq 1$, then we use the rules in Group (D) with $\ell = 1$, $\mathcal{L} = \mathcal{L}'+1$ (resp. $\mathcal{L} = \mathcal{L}'+2$ if v is of the form $\mathsf{ev}^{\mathcal{L}'+1}v'w'$; recall that since v is at level $\mathcal{L}'$, its quoted version is at level $\mathcal{L}'+1$); then, we apply the induction hypothesis straightforwardly. For example, if $v = \lambda^{\mathcal{L}'}w$, then $v‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) = \lambda^{\mathcal{L}'+1}(w‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]))$, so $v‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m)$ rewrites by rule $(\lambda^{\mathcal{L}'+1}\circ^1)$ to $\lambda^{\mathcal{L}'+1}(w‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m))$, which rewrites by induction hypothesis to $\lambda^{\mathcal{L}'+1}(w‘(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n+m, \ldots, x_p \mapsto n+m+p-1]))$, i.e. $v‘(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n+m, \ldots, x_p \mapsto n+m+p-1]))$.

If v is at level 0, we have several cases:

- If v is of the form $\mathsf{ev}^1v'w'$, then we use rule $(\mathsf{ev}^2\circ^1)$ and apply the induction hypothesis.
- If v is a variable inside the domain of ρ, say $\rho(v) = i$, with $0 \leq i \leq n-1$, then $v‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1])$ is $\mathsf{get}^1_{n+p-i-1}$, that is, $1^1 \circ^1 \mathsf{pop}^1_{n+p-i-1}$. Then:

$$\begin{array}{ll}v‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m) & \\ = (1^1 \circ^1 \mathsf{pop}^1_{n+p-i-1}) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m) & \\ \xrightarrow{\circ_1} 1^1 \circ^1 (\mathsf{pop}^1_{n+p-i-1} \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m)) & \text{by } (\circ^1) \\ \xrightarrow{\circ_1}^+ 1^1 \circ^1 \mathsf{pop}^1_{m+p+n-i-1} & \text{by Lemma 3.15 } (iii) \\ = \mathsf{get}^1_{m+p+n-i-1} & \\ = v‘(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n+m, \ldots, x_p \mapsto n+m+p-1]) & \end{array}$$

- If v is a variable outside the domain of ρ and other than the x_i's, then it is also a variable x other than any y_i, since y_i is not free in v. Then, $v‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1])$ is Q^1x. Therefore $v‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m)$ reduces to Q^1x by rule $(Q\circ^1)$, and the latter equals $v‘(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n+m, \ldots, x_p \mapsto n+m+p-1])$ by definition.
- If v is one of the x_i's, $1 \leq i \leq p$, then $v‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) = \mathsf{get}^1_{p-i}$, so $v‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m)$ rewrites by rule $(\circ^1)$ and Lemma 3.15 (i) to $\mathsf{get}_{p-i};^1 \Uparrow^{1^p}(\mathsf{pop}^1_m)$, then by Lemma 3.15 (v) to get^1_{p-i}. The latter is equal by definition to $v‘(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n+m, \ldots, x_p \mapsto n+m+p-1])$.
- If v is $()$, then $v‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1])$ is pop^1_p. By Lemma 3.15 (iii), $v‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m) \xrightarrow{\circ_1}^+ \mathsf{pop}^1_{m+p}$, and the latter is precisely $v‘(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n+m, \ldots, x_p \mapsto n+m+p-1])$.
- If v is of the form uw, then $(uw)‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m)$ is $((u‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1])) \star^1 (w‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]))) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m)$, which rewrites to $(u‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m)) \star^1 (w‘(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m))$ by

rule ($\star^1$), hence to $(u`(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n+m, \ldots, x_p \mapsto n+m+p-1])) \star^1 (w`(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n+m, \ldots, x_p \mapsto n+m+p-1]))$ by induction hypothesis, that is to $v`(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n+m, \ldots, x_p \mapsto n+m+p-1])$.

If v is $\uparrow u$, $1u$, or $u \bullet v$, the argument is similar: we use rule ($\circ^1$) in the first two cases, and rule ($\bullet^1$) in the third case.

- If v is of the form $\lambda x \cdot u$, then $v`(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1]) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m)$ is $\lambda^1(u`(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1, x \mapsto n+p])) \circ^1 \Uparrow^{1^p}(\mathsf{pop}^1_m)$, which rewrites to $\lambda^1(u`(\rho[x_1 \mapsto n, \ldots, x_p \mapsto n+p-1, x \mapsto n+p])) \circ^1 \Uparrow^{1^{p+1}}(\mathsf{pop}^1_m)$ by rule ($\lambda^1$), therefore to $\lambda^1(u`(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n, \ldots, x_p \mapsto n+p-1, x \mapsto n+p]))$ by induction hypothesis, that is to $v`(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x_1 \mapsto n+1, \ldots, x_p \mapsto n+p])$.

$\square$

Lemma 3.17 *For every environment ρ of cardinality n, for every terms u and v, for every variable x outside the domain of ρ:*

$$(u`(\rho[x \mapsto n])) \circ^1 (v`\rho \bullet^1 id^1) \xrightarrow{\Sigma}{}^* (u[v/x])`\rho$$

Proof: Because of the way that λ-abstractions are quoted, we shall prove a more general result, namely that for every integer $m \geq 0$, for every variables $y_1, \ldots, y_m$ outside the domain of ρ and not free in v, and for every variable x outside the domain of ρ:

$$\begin{array}{l}(u`(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m])) \circ^1 \Uparrow^{1^m}(v`\rho \bullet^1 id^1) \\ \xrightarrow{\Sigma}{}^* (u[v/x])`(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1])\end{array}$$

We prove this by structural induction on u.

All cases except the one where u is a variable or a λ-abstraction are trivial, and simply propagate the induction hypothesis inwards, using the rules in group (B) (except (β^ℓ)) and in group (D) to propagate the composition inwards.

The case of a λ-abstraction works as follows. Let u be $\lambda y \cdot w$, then $u`(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m])$ is $\lambda^1(w`(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m, y \mapsto n+m+1]))$, so $(u`(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m])) \circ^1 \Uparrow^{1^m}(v`\rho \bullet^1 id^1)$ rewrites to $\lambda^1((w`(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m, y \mapsto n+m+1])) \circ^1 \Uparrow^{1^{m+1}}(v`\rho \bullet^1 id^1))$ by rule (λ^1). By induction hypothesis, the latter rewrites to $\lambda^1((w[v/x])`(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, y \mapsto n+m]))$, since by our naming conventions $y$ is not free in $v$. And the latter is precisely $(u[v/x])`(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1])$.

When u is a variable, we have four cases:

- When u is a variable other than x, y_1, ..., y_m, and outside the domain of ρ, $u`(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m])$ is $Q^1 u$, so $u`(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m]) \circ^1 \Uparrow^{1^m}(v`\rho \bullet^1 id^1)$ rewrites by ($Q\circ^1$) to $Q^1 u$. But the latter is precisely $(u[v/x])`(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1])$, since u is a variable other than x (so that $u[v/x] = u$) and u is a variable other than y_1, ..., y_m.

- When u is in the domain of ρ, then $u`(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m])$ is $\mathsf{get}^1_{m+1+j}$, with $j \geq 0$. Then, $u`(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m]) \circ^1 \Uparrow^{1^m}(v`\rho \bullet^1 id^1)$ rewrites by ($\circ^1$) and Lemma 3.15 ($i$) to $1^1 \circ^1 (\mathsf{pop}_{m+1+j} ;^1 \Uparrow^{1^m}(v`\rho \bullet^1 id^1))$, hence to $1^1 \circ^1 (\mathsf{pop}_{1+j} ;^1 (v`\rho \bullet^1 id^1)[\mathsf{pop}_m]^1))$ by Lemma 3.15 ($ii$). Then, this rewrites to $1^1 \circ^1 (\mathsf{pop}_{1+j} ;^1 (v`\rho[\mathsf{pop}_m]^1 \bullet^1 \mathsf{pop}^1_m))$, by rule ($\bullet^1$) if $m \geq 1$ (if $m = 0$, this is equal to this term). By rule ($\uparrow^1$), this rewrites to $1^1 \circ^1 (\mathsf{pop}_j ;^1 \mathsf{pop}^1_m)$, and this is equal to get^1_{j+m} if $m \geq 1$, or rewrites to it by ($\circ id^1$) if $m = 0$. But, since u is a variable that cannot be x, this is precisely $(u[v/x])`(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1])$.

- When u is y_i, $1 \leq i \leq m$, then $u`(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m])$ is $\mathsf{get}^1_j$ with $j < m$. Then, $u`(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m]) \circ^1 \Uparrow^{1^m}(v`\rho \bullet^1 id^1)$ rewrites by ($\circ^1$) and Lemma 3.15 ($i$) to $\mathsf{get}_j ;^1 \Uparrow^{1^m}(v`\rho \bullet^1 id^1)$, then by Lemma 3.15 ($v$) to get^1_j, that is, to $(u[v/x])`(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1])$. (Recall that $u = y_i$ is a variable other that x.)

- When u is x, by the same argument, $u‘(\rho[x \mapsto n, y_1 \mapsto n+1, \ldots, y_m \mapsto n+m]) \circ^1 \Uparrow^{1^m}(v‘\rho \bullet^1 id^1) = \mathsf{get}^1_m \circ^1 \Uparrow^{1^m}(v‘\rho \bullet^1 id^1)$ rewrites to $1^1 \circ^1 (\mathsf{pop}_m ;^1 \Uparrow^{1^m}(v‘\rho \bullet^1 id^1))$ by $(\circ^1)$ and Lemma 3.15 (i). The latter rewrites to $1^1 \circ^1 (v‘\rho \bullet^1 id^1)[\mathsf{pop}_m]^1$ by Lemma 3.15 (ii). The latter is (when $m = 0$), or rewrites by $(\bullet^1)$ (when $m \geq 1$) to $1^1 \circ^1 (v‘\rho[\mathsf{pop}_m]^1 \bullet^1 \mathsf{pop}^1_m)$, which rewrites by (1^1) to $v‘\rho[\mathsf{pop}_m]^1$. Finally, the latter rewrites to $v‘(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1])$ by Lemma 3.16, that is to $(u[v/x])‘(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1])$.

$\square$

The following says that reduction works (more precisely, can be simulated) under quotations. To handle residual marks, we adopt the convention that $(\mathsf{ev}^\ell_* uv)‘\rho = \mathsf{ev}^{\ell+1}_*(u‘\rho)(v‘\rho)$, and $(Q^\ell_* u)‘\rho = Q^{\ell+1}_*(u‘\rho)$.

Theorem 3.18 (Reduction Under Quote) *Let $\mathcal{R}$ be either the set of all $\lambda\mathsf{ev}Q$-rules, or Σ, or $\circ_\ell$, or $‘_\ell$, or $’_\ell$ (for some given $\ell \geq 1$). Let $\mathcal{R}‘$ be the set of all $\lambda\mathsf{ev}Q$-rules, or Σ, or $\circ_{\ell+1}$, or $‘_{\ell+1}$, or $’_{\ell+1}$ respectively.*

For all $\lambda\mathsf{ev}Q$-terms u and v (possibly with star marks), if $u \xrightarrow{\mathcal{R}} v$, then $u‘\rho \xrightarrow{\mathcal{R}‘}{}^{+} v‘\rho$ for all environments ρ.

In particular, if $u \xrightarrow{\mathcal{R}}{}^{*} v$, then $u‘\rho \xrightarrow{\mathcal{R}‘}{}^{*} v‘\rho$.

Proof: The latter claim follows from the former by induction on the length of the $\mathcal{R}$ derivation from u to v. We now prove the former claim by structural induction on u. The base case is when u is itself the redex, and the induction case is by case analysis on u.

The induction case is mostly trivial: we deal with applications, as all other cases are similar. If u is not the redex that is reduced to v and u is an application $u'u''$, then $v = v'v''$, where either $u' = v'$ and $u'' \longrightarrow v''$, or $u' \longrightarrow v'$ and $u'' = v''$. Then $u‘\rho = (u'‘\rho) \star^1 (u''‘\rho)$, which rewrites by induction hypothesis to $(v'‘\rho) \star^1 (v''‘\rho) = u‘\rho$.

We now turn to the base case, namely when u is itself the redex that is reduced to yield v. If the rule that was used is anything but (β) or $(\mathsf{ev}\lambda^1)$ (or $(\mathsf{ev}_*\lambda^1)$), then we can draw the more precise conclusion that $u‘\rho \longrightarrow v‘\rho$ (in one step). The correspondence between the rule R used to rewrite u into v and the rule $R‘$ used to rewrite $u‘\rho$ into $v‘\rho$ is as follows: if $R = (\uparrow)$, then $R‘ = (\uparrow^1)$; if $R = (1)$, then $R‘ = (1^1)$; if R is a rule in groups (B), (C) (except for $(\mathsf{ev}\lambda^1)$) at level ℓ, then $R‘$ is the same rule at level $\ell+1$ (it is a bit less obvious for rules $(\mathsf{ev} \uparrow^\ell)$, $(\mathsf{ev}1^\ell)$, $(1\mathsf{ev} \Uparrow^\ell)$ and $(\uparrow \mathsf{ev} \Uparrow^\ell)$); if R is a rule in groups (D), (E) or (F) at levels ℓ and $\mathcal{L}$, $1 \leq \ell < \mathcal{L}$, then $R‘$ is the same rule at levels $\ell+1$ and $\mathcal{L}+1$.

If u is a (β)-redex that reduces to v (this applies only to the case where $\mathcal{R}$ is the set of all rules), then $u = (\lambda x \cdot u')u''$ and $v = u'[u''/x]$, and $u‘\rho = (\lambda^1(u'‘(\rho[x \mapsto n]))) \star^1 (u''‘\rho)$. The latter rewrites by (β^1) to $(u'‘(\rho[x \mapsto n])) \circ^1 (u''‘\rho \bullet^1 id^1)$, which rewrites by Lemma 3.17 to $(u'[u''/x])‘\rho$, that is, $v‘\rho$.

The last case is when u is an $(\mathsf{ev}\lambda^1)$-redex (or $(\mathsf{ev}_*\lambda^1)$-redex: this applies when $\mathcal{R}$ is the set of all rules or $’_1$). Let's deal with the $(\mathsf{ev}_*\lambda^1)$ case, the $(\mathsf{ev}\lambda^1)$ case is similar. Then $u = \mathsf{ev}^1_*(\lambda^1 u')w$, $v = \lambda x \cdot \mathsf{ev}^1_* u'(x \bullet w)$, and $u‘\rho = \mathsf{ev}^2_*(\lambda^2(u'‘\rho))(w‘\rho)$. This rewrites by rule $(\mathsf{ev}_*\lambda^2)$ to $\lambda^1(\mathsf{ev}^2_*((u'‘\rho)\circ^1 \uparrow^1)(1^1 \bullet^1 w‘\rho\circ^1 \uparrow^1))$. But, by Lemma 3.16 (with $m = 1$), $(u'‘\rho)\circ^1 \uparrow^1$ reduces to $u'‘(\rho[x \mapsto n])$ and $(w‘\rho)\circ^1 \uparrow^1$ reduces to $w‘(\rho[x \mapsto n])$, so that $u‘\rho$ reduces to $\lambda^1(\mathsf{ev}^2_*(u'‘(\rho[x \mapsto n]))(1^1 \bullet^1 w‘(\rho[x \mapsto n])))$, i.e. to $v‘\rho$. $\square$

Corollary 3.19 *Let $\mathcal{R}$ and $\mathcal{R}‘$ be as in Theorem 3.18.*

Let u and v be arbitrary $\lambda\mathsf{ev}Q$-terms (possibly with star marks), and σ be an arbitrary substitution.
If $u \xrightarrow{\mathcal{R}}{}^{*} v$, then $(u‘[])\sigma \xrightarrow{\mathcal{R}‘}{}^{*} (v‘[])\sigma$.

Proof: Apply Theorem 3.18 with $\rho = []$. Then, notice that if $u_1 \xrightarrow{\mathcal{R}}{}^{*} u_2$, then $u_1\sigma \xrightarrow{\mathcal{R}}{}^{*} u_2\sigma$, by induction on the length of the derivation. (The only non-trivial case is for (β) steps, which are handled by the variable naming convention.) $\square$

For the next results, we need the following notion, which is independent of being typable. Recall that we have adopted Barendregt's naming convention. We ignore star marks in this definition: more precisely, this definition considers the star-erasures of terms.

Definition 3.5 (Well-staged) *We define the* in-level $\ell(u)$ *of non-variable $\lambda\mathsf{ev}Q$-terms u as follows:*

- *The in-level of $\lambda x \cdot u$, uv, $()$, $\uparrow u$, $1u$, $u \bullet v$ is 0.*
- *The in-level of $\lambda^\ell u$, $u \star^\ell v$, $\uparrow^\ell$, 1^ℓ, $u \bullet^\ell v$, id^ℓ, $u \circ^\ell v$, $\Uparrow^\ell u$, $\ell \geq 1$, is ℓ.*
- *The in-level of $\mathsf{ev}^\ell uv$ or of $Q^\ell u$, $\ell \geq 1$, is $\ell - 1$.*

We define the immediate subterms *of any non-variable term as: u and v are those of uv, $u \bullet v$, $u \star^\ell v$, $u \bullet^\ell v$, $u \circ^\ell v$, or $\mathsf{ev}^\ell uv$; u is the only immediate subterm of $\lambda x \cdot u$, $\uparrow u$, $1u$, $\lambda^\ell u$, $\Uparrow^\ell u$, or $Q^\ell u$; all other terms have no immediate subterm.*

We say that a $\lambda \mathsf{ev} Q$-term u is well-staged *if and only if for every non-variable subterm v of u, for every immediate subterm v' of v, the out-level of v' is at least the in-level of v, i.e. $\mathcal{L}(v') \geq \ell(v)$.*

This definition is justified by the following lemmas:

Lemma 3.20 *Let u be a term at level ℓ, $\ell \geq 0$, and ρ be an environment.*
Then $u`\rho$ is at level $\ell + 1$.

Proof: An easy case analysis on the definition of $\_`$ in Figure 2. □

Lemma 3.21 *Let u be a well-staged term, $v_1, \ldots, v_n$ be n well-staged terms, and $x_1, \ldots, x_n$ be pairwise distinct variables.*
For every environment ρ, $(u`\rho)[v_1/x_1, \ldots, v_n/x_n]$ is well-staged.

Proof: By structural induction on u, using Lemma 3.20. The only non-trivial cases are when u is a variable, or a λ-abstraction.

If u is a variable in the domain of ρ, then $(u`\rho)[v_1/x_1, \ldots, v_n/x_n]$ is $\mathsf{get}_i^1$ for some integer $i$, which is well-staged. If $u$ is a variable outside the domain of $\rho$, and is not one of the $x_i$'s, then $(u`\rho)[v_1/x_1, \ldots, v_n/x_n]$ is $Q^1 u$, which is well-staged (the condition on $Q^1 u$, that u be at level at least 0, is vacuously satisfied). Finally, if u is x_i for some i, $1 \leq i \leq n$, and is outside the domain of ρ, then $(u`\rho)[v_1/x_1, \ldots, v_n/x_n]$ is $Q^1 v_i$, which is again well-staged, because v_i is well-staged (and at level at least 0).

If u is a λ-abstraction $\lambda x \cdot v$, then $u`\rho = \lambda^1(v`(\rho[x \mapsto n]))$, where n is the cardinality of ρ. Now, since u is well-staged, v is also well-staged and by induction hypothesis $v`(\rho[x \mapsto n])$ is well-staged. Moreover, it is at level at least 1 by Lemma 3.20, so $u`\rho$ is also well-staged.

All other cases are trivial. For example, if $u = \lambda^\ell v$, then by well-stagedness of u, v is well-staged and at level at least ℓ. Now, $u`\rho = \lambda^{\ell+1}(v`\rho)$, where $v`\rho$ is well-staged by induction hypothesis, and at level at least $\ell + 1$ by Lemma 3.20, so that $u`\rho$ is well-staged. □

Finally:

Lemma 3.22 *Every translation of a λ_{S4}-term by the rules of Figure 2 is well-staged.*

Proof: By structural induction on the λ_{S4}-term, using Lemma 3.21 for the case of quotations, i.e. terms of the form $\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n$. □

Not all typable terms are well-staged: for example, $Q^2 x$ is not well-staged, since its in-level is 1, but x is at level 0; and it is typable of type $\Phi_1 \stackrel{\Box}{\Rightarrow} \Phi_2 \stackrel{\Box}{\Rightarrow} \Phi_3 \stackrel{\Box}{\Rightarrow} \Phi_4$, as soon as x is assumed of type $\Phi_1 \stackrel{\Box}{\Rightarrow} \Phi_3 \stackrel{\Box}{\Rightarrow} \Phi_4$. (Quoting x twice, on the other hand, yields $Q^2(Q^1 x)$, which is both typable and well-staged.) Conversely, not all well-staged terms are typable: for example, xx is well-staged but not typable.

Moreover, although typability is preserved by the reduction rules, well-stagedness is not, as most rules in Group (E) produce ill-staged terms. For example, by rule $(\mathsf{ev}^1\lambda^2)$, $\mathsf{ev}^1(\lambda^2 u)w$ reduces to $\lambda^1(\mathsf{ev}^1 uw)$, which is not well-staged.

Before we can prove that $\lambda \mathsf{ev} Q$ can simulate rule (box), we have to define another notion:

Definition 3.6 *A $\lambda \mathsf{ev} Q$-term u is said to be* Q^1-pure *if and only if, for every subterm of u of the form $Q^1 v$, v is a variable.*

Lemma 3.23 *Let u be any $\lambda \mathsf{ev} Q$-term. Then, for every environment ρ, $u`\rho$ is Q^1-pure.*

Proof: By structural induction on u. All cases are trivial, except when u is a variable x. Then $u‘\rho$ is either get^1_n for some integer n, which is Q^1-pure; or it is $Q^1 x$, which is again Q^1-pure. □

Lemma 3.24 *For any well-staged and Q^1-pure $\lambda\mathsf{ev}Q$-term u of level at least 1 (and without star marks), $Q^1 u \longrightarrow^+ u‘[]$.*

More precisely, $Q^1_* u \overset{‘}{\xrightarrow{1}}{}^+ u‘^*[]$, where $u‘^*\rho$ is defined as $u‘\rho$, except in the case where u is a variable x outside the domain of ρ, where we define $x‘^*\rho$ as $Q^1_* x$.

Proof: The former result follows from the latter by star erasure. So, let's prove the latter claim, by structural induction on u.

Most cases are trivial: for example, if $u = v \bullet^\ell w$, $\ell \geq 1$, then $Q^1_* u \overset{‘}{\xrightarrow{1}} (Q^1_* v) \bullet^{\ell+1} (Q^1_* w)$ by rule $(Q^1_* \bullet^{\ell+1})$. Since u is well-staged, v and w are well-staged and at level at least ℓ, in particular at least 1. Moreover, v and w are Q^1-pure. We can therefore apply the induction hypothesis, and conclude that $Q^1_* u \overset{‘}{\xrightarrow{1}}{}^+ (v‘[]) \bullet^{\ell+1} (w‘[]) = (v \bullet^\ell w)‘[] = u‘[]$. The argument is similar for all other cases (except when $u = Q^1 v$, with v at level 0).

If $u = Q^1 v$, with v at level 0, then v must be a variable since u is Q^1-pure. Then, $Q^1_* u \overset{‘}{\xrightarrow{1}} Q^2(Q^1_* v)$ by rule $(Q^1_* Q^2)$; but the latter is precisely $(Q^1 v)‘^*[]$. □

Lemma 3.25 *Let u be a $\lambda\mathsf{ev}Q$-term, and ρ be an environment whose domain does not contain any free variable of u. Then $u‘\rho = u‘[]$, and $u‘^*\rho = u‘^*[]$.*

Proof: By star erasure, it is enough to prove the latter claim.

We prove the more general result that for any such environment ρ, and for any variables $y_1, \ldots, y_m$ outside the domain of ρ:

$$u‘^*(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1]) = u‘^*[y_1 \mapsto 0, \ldots, y_m \mapsto m-1]$$

where n is the cardinality of ρ, by structural induction on u.

If u is a variable, then we have two cases. If u is some y_i, $1 \leq i \leq m$, then both sides are get^1_{m-i}. Otherwise, u is not in the domain of $\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1]$ (since it is not in the domain of ρ) or of $[y_1 \mapsto 0, \ldots, y_m \mapsto m-1]$, so both sides are $Q^1_* u$.

If u is a λ-abstraction $\lambda x \cdot v$, then by induction hypothesis $v‘^*(\rho[y_1 \mapsto n, \ldots, y_m \mapsto n+m-1, x \mapsto n+m]) = u‘^*[y_1 \mapsto 0, \ldots, y_m \mapsto m-1, x \mapsto m]$, hence the result. All other cases are trivial induction cases. □

Lemma 3.26 *Let u be a $\lambda\mathsf{ev}Q$-term, v be a well-staged and Q^1-pure term of level at least 1, x be a variable. Let ρ be an environment whose domain does not contain x or any free variable of v.*

Then $(u‘\rho)[v/x] \longrightarrow^+ (u[v/x])‘\rho$.

More precisely, $(u‘^*\rho)[v/x] \overset{‘}{\xrightarrow{1}}{}^+ (u[v/x])‘^*\rho$.

Proof: By structural induction on u.

When u is x, since x is not in the domain of ρ, $u‘^*\rho = Q^1_* x$; so $(u‘\rho)[v/x] = Q^1_* v$ rewrites by Lemma 3.24 to $v‘^*[]$. By Lemma 3.25, the latter is equal to $v‘^*\rho$.

When u is another variable, this is trivial. All other cases of the induction are straightforward. Observe that we need the generality on ρ in the case of λ-abstractions. □

Theorem 3.27 (box) *Let u, $v_1, \ldots, v_n$ be $n+1$ λ_{S4}-terms, $x_1, \ldots, x_n$ be n distinct variables. If for some i, $1 \leq i \leq n$, v_i is of the form* $\mathsf{box}\ v\ \mathsf{with}\ w_1, \ldots, w_m\ \mathsf{for}\ y_1, \ldots, y_m$*, then:*

$$G(\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n)$$
$$\overset{‘}{\xrightarrow{1}}{}^+ G(\mathsf{box}\ u[v‘/x_i]\ \mathsf{with}\ v_1, \ldots, v_{i-1}, w_1, \ldots, w_m, v_{i+1}, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_{i-1}, y_1, \ldots, y_m, x_{i+1}, \ldots, x_n)$$

in $\lambda\mathsf{ev}Q$.

Proof: Since v_i is a box-term, write it $\mathsf{box}\ v\ \mathsf{with}\ w_1, \ldots, w_m\ \mathsf{for}\ y_1, \ldots, y_m$. By definition,

$$G(\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n)$$

equals:

$$\begin{array}{ll}((G(u))^{\lq}[]) & [G(v_1)/x_1, \ldots, G(v_{i-1})/x_{i-1}, \\ & ((G(v))^{\lq}[])[G(w_1)/y_1, \ldots, G(w_m)/y_m]/x_i, \\ & G(v_{i+1})/x_{i+1}, \ldots, G(v_n)/x_n]\end{array}$$

Because of the variable naming convention, this is also equal to:

$$\begin{array}{ll}((G(u))^{\lq}[]) & [G(v_1)/x_1, \ldots, G(v_{i-1})/x_{i-1}, (G(v))^{\lq}[]/x_i, G(v_{i+1})/x_{i+1}, \ldots, G(v_n)/x_n] \\ & [G(w_1)/y_1, \ldots, G(w_m)/y_m]\end{array}$$

that is, to:

$$\begin{array}{ll}((G(u))^{\lq}[]) & [(G(v))^{\lq}[]/x_i][G(v_1)/x_1, \ldots, G(v_{i-1})/x_{i-1}, G(v_{i+1})/x_{i+1}, \ldots, G(v_n)/x_n] \\ & [G(w_1)/y_1, \ldots, G(w_m)/y_m]\end{array}$$

By Lemmas 3.22 and 3.21, $(G(v))^{\lq}[]$ is well-staged; by Lemma 3.20, it as at level at least 1; by Lemma 3.23, $(G(v))^{\lq}[]$ is Q^1-pure. So Lemma 3.26 applies with $\rho = []$, and $G(\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n)$ reduces to:

$$\begin{array}{ll}((G(u)[(G(v))^{\lq}[]/x_i])^{\lq}[]) & [G(v_1)/x_1, \ldots, G(v_{i-1})/x_{i-1}, G(v_{i+1})/x_{i+1}, \ldots, G(v_n)/x_n] \\ & [G(w_1)/y_1, \ldots, G(w_m)/y_m]\end{array}$$

which is also equal, because of the variable naming conventions, to:

$$\begin{array}{ll}((G(u)[(G(v))^{\lq}[]/x_i])^{\lq}[]) & [G(v_1)/x_1, \ldots, G(v_{i-1})/x_{i-1}, \\ & G(w_1)/y_1, \ldots, G(w_m)/y_m, \\ & G(v_{i+1})/x_{i+1}, \ldots, G(v_n)/x_n]\end{array}$$

Finally, the result follows by Lemma 3.6. □

A side note: we might have chosen to have typed Q^ℓ in the more restrictive way: $Q^\ell u : \overline{\Psi^{\ell-1}} \overset{\Box}{\Rightarrow} \top \overset{\Box}{\Rightarrow} \Phi_1 \overset{\Box}{\Rightarrow} \Phi_2$ provided $u : \overline{\Psi^{\ell-1}} \overset{\Box}{\Rightarrow} \Phi_1 \overset{\Box}{\Rightarrow} \Phi_2$. Rule $(Q\circ^\ell)$ would have to be suppressed, and Lemma 3.25 would not be valid, but all other results (including Lemma 3.26 and Theorem 3.27) would still be. In fact, the termination and confluence results of the following sections would also hold, and we would get η-like rules by adding the rule $\mathsf{ev}^{\ell+1}(Q^\ell u \circ^\ell v)w \to u \circ^\ell w$ to the η-like rules in the sequel. (This is what we did in a first version of this paper.) However, we feel that our choice here is slightly more natural.

3.3 Simulating Cut Elimination

Similarly to the categories $\underline{\mathbf{S4}}$ and $\overline{\mathbf{S4}}$ of Part I, the $\lambda\mathsf{ev}Q$-calculus defines a category, provided that we consider terms with enough type annotations:

Definition 3.7 *The set of* typed $\lambda\mathsf{ev}Q$ pre-terms *is the set of $\lambda\mathsf{ev}Q$-terms built on typed variables (that is, variables x_τ having a unique term type τ annotating them), and where subterms of the form 1^ℓ, $\uparrow^\ell$, id^ℓ, $\Uparrow^\ell u$ and $Q^\ell u$ are annotated with formulas. We won't usually show these types, unless confusion might arise; if u is annotated with formula Φ, we shall also write it $(u)_\Phi$.*

The typed $\lambda\mathsf{ev}Q$ terms *are the typed $\lambda\mathsf{ev}Q$ pre-terms that are typable when every variable x_τ is assumed of type τ, and where every annotated subterm $(u)_\Phi$ has type Φ.*

The category $\lambda\mathsf{ev}Q$ has as objects all typed $\lambda\mathsf{ev}Q$-terms, and as morphisms all derivations between terms.

That every typed $\lambda\mathsf{ev}Q$-term has a unique type is immediate. It is also clear that typed $\lambda\mathsf{ev}Q$-terms are in bijection with natural deduction proofs in the system of Figure 1, modulo weakenings. What is less clear is the set of rewrite rules that define derivations between typed terms. Consider for example the untyped rule $(1 \Uparrow^\ell)$: $1^\ell \circ^\ell \Uparrow^\ell u \to 1^\ell$. In the typed case, this rule must be written as $(1^\ell)_{\overline{\varsigma^{\ell-1}} \overset{\Box}{\Rightarrow} \tau \times \varsigma \overset{\Box}{\Rightarrow} \tau} \circ^\ell (\Uparrow^\ell u)_{\overline{\varsigma^{\ell-1}} \overset{\Box}{\Rightarrow} \tau \times \varsigma' \overset{\Box}{\Rightarrow} \tau \times \varsigma} \to$

$(1^\ell)_{\overline{\varsigma^{\ell-1}\overset{\Box}{\Rightarrow}}\tau \times \varsigma' \overset{\Box}{\Rightarrow} \tau}$. Given the types decorating subterms in the left-hand side, the type decorating the right-hand side is determined uniquely. It is routine to check that, for each rule, given a decoration of subterms of the left-hand side, there is a unique decoration of subterms of the right-hand side so that the right-hand side has the same type as the left-hand side. The most complicated case is rule $(\mathsf{ev}\lambda^\ell)$ with $\ell \geq 2$, which is:

$$\begin{aligned}\mathsf{ev}^\ell(\lambda^\ell u)w \to \quad & \lambda^{\ell-1}(\mathsf{ev}^\ell(u \circ^{\ell-1} (\uparrow^{\ell-1})_{\overline{\varsigma^{\ell-2}\overset{\Box}{\Rightarrow}}\tau_1 \times \varsigma_{\ell-1} \overset{\Box}{\Rightarrow} \varsigma_{\ell-1}}) \\ & ((1^{\ell-1})_{\overline{\varsigma^{\ell-2}\overset{\Box}{\Rightarrow}}\tau_1 \times \varsigma_{\ell-1} \overset{\Box}{\Rightarrow} \tau_1} \bullet^{\ell-1} (w \circ^{\ell-1} (\uparrow^{\ell-1})_{\overline{\varsigma^{\ell-2}\overset{\Box}{\Rightarrow}}\tau_1 \times \varsigma_{\ell-1} \overset{\Box}{\Rightarrow} \varsigma_{\ell-1}})))\end{aligned}$$

where the type of u is $\overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}}\tau_1 \times$
$varsigma' \overset{\Box}{\Rightarrow} \tau_2$, and the type of w is $\overline{\varsigma^{\ell-1} \overset{\Box}{\Rightarrow}}$
$varsigma'$.

Interpreting cut-elimination in S4 inside $\lambda\mathsf{ev}Q$ means finding a translation of S4 proofs to $\lambda\mathsf{ev}Q$-terms, and another translation from cut-eliminations in S4 (morphisms in $\underline{\mathbf{S4}}$, resp. $\overline{\mathbf{S4}}$) to derivations in $\lambda\mathsf{ev}Q$. That is, an interpretation of cut-elimination in S4 into $\lambda\mathsf{ev}Q$ is a *functor* from $\underline{\mathbf{S4}}$ (resp. $\overline{\mathbf{S4}}$) to the category $\lambda\mathsf{ev}Q$. As suggested by the theorems of Section 3.2, the G-translation will be this interpretation. It just remains to prove that types are preserved by G:

Definition 3.8 *We let G operate on types in the following way: $G(A) = A$ for every basic type A, $G(\Phi_1 \Rightarrow \Phi_2) = G(\Phi_1) \Rightarrow G(\Phi_2)$, $G(\Box\Phi) = \top \overset{\Box}{\Rightarrow} G(\Phi)$.*

Observe that every G-translated type is a term type. Then:

Theorem 3.28 *If , $\vdash u : \Phi$ is provable in the natural deduction system for S4 (Figure 4, Part I), then $G(,) \vdash G(u) : G(\Phi)$ is provable in the natural deduction system of Figure 1.*

Proof: By structural induction on the natural deduction proof. The only difficulty is the translation of the $(\Box\mathcal{I})$ rule (that is, of quotations). In this case,

$$, ' \vdash \mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_m\ \mathsf{for}\ x_1, \ldots, x_m : \Box\Phi$$

has been derived from:

$$x_1 : \Box\Phi_1, \ldots, x_m : \Box\Phi_m \vdash u : \Phi$$

and $, ' \vdash v_i : \Box\Phi_i$, $1 \leq i \leq m$.

By induction hypothesis, we have therefore derived the following sequent:

$$x_1 : \top \overset{\Box}{\Rightarrow} G(\Phi_1), \ldots, x_m : \top \overset{\Box}{\Rightarrow} G(\Phi_m) \vdash G(u) : G(\Phi)$$

By Theorem 3.3 with $n = 0$, and since the context , above is boxed, $(G(u))^{\text{‘}}[]$ has type $\top \overset{\Box}{\Rightarrow} G(\Phi)$ in context , , i.e. , $\vdash G(u^{\text{‘}}) : \top \overset{\Box}{\Rightarrow} G(\Phi)$ is derivable in the natural deduction system of Figure 1.

Then, by induction hypothesis again, the sequents $G(, ') \vdash G(v_i) : \top \overset{\Box}{\Rightarrow} G(\Phi_i)$ have also been derived for every i, $1 \leq i \leq m$. It follows by substitution that the following sequent is derivable:

$$G(, ') \vdash ((G(u))^{\text{‘}}[])[G(v_1)/x_1, \ldots, G(v_m)/x_m]$$

which is exactly what we were after. $\Box$

We need a language for describing derivations, or at least single steps in λ_{S4}. For any rule (R) transforming a contractum u into a contracted term v, we denote this step by $u \xrightarrow{(R)} v$. In general, for any context $\mathcal{C}$ (i.e. a term with exactly one hole $[]$; $\mathcal{C}[s]$ then denotes this term with the hole filled in with s), we denote by $\mathcal{C}[u] \xrightarrow{\mathcal{C}[(R)]} \mathcal{C}[v]$ the reduction of the redex located at the position of the hole in $\mathcal{C}$.

Theorem 3.29 (Simulation of Cut Elimination) *The G-translation induces a functor from $\underline{\mathbf{S4}}$ (resp. $\overline{\mathbf{S4}}$) to the category $\lambda\mathsf{ev}Q$, defined as follows. The action of G on objects is given by Figure 2; the action of G on morphisms is given by (using a vector notation as abbreviation of lists of terms or variables):*

- *$G((\lambda x \cdot u)v \xrightarrow{(\beta)} u[v/x]) = G((\lambda x \cdot u)) \xrightarrow{(\beta)} G(u[v/x])$;*

- *if u is an* (unbox)*-redex whose contractum is v, then $G(u\overset{(\mathsf{unbox}\ )}{\longrightarrow}v)$ is given by Corollary 3.14;*
- *if u is a* (box)*-redex whose contractum is v, $G(u\overset{(\mathsf{box})}{\longrightarrow}v)$ is given by Theorem 3.27;*
- *if $u\xrightarrow{d}v$ is a morphism in $\underline{\mathbf{S4}}$ (resp. $\overline{\mathbf{S4}}$), then*

$$G(\mathsf{box}\ u\ \mathsf{with}\ \vec{w}\ \mathsf{for}\ \vec{x}\overset{\mathsf{box}\ d\ \mathsf{with}\ \vec{w}\ \mathsf{for}\ \vec{x}}{\longrightarrow}\mathsf{box}\ v\ \mathsf{with}\ \vec{w}\ \mathsf{for}\ \vec{x})$$

  is defined from the morphism $G(u\xrightarrow{d}v)$ by applying Corollary 3.19;
- *if $u\xrightarrow{d}v$ is a morphism in $\underline{\mathbf{S4}}$ (resp. $\overline{\mathbf{S4}}$), and $\mathcal{C}$ is a context of height one, i.e. of the form $\lambda x\cdot[]$, $[]w$, $w[]$, $\mathsf{unbox}\ []$ or $\mathsf{box}\ w\ \mathsf{with}\ w_1,\ldots,w_{i-1},[],w_{i+1},\ldots,w_n\ \mathsf{for}\ x_1,\ldots,x_n$, then $G(\mathcal{C}[u]\xrightarrow{\mathcal{C}[d]}\mathcal{C}[v]) = G(\mathcal{C})[G(u\xrightarrow{d}v)]$, where $G(\mathcal{C})$ is respectively $\lambda x\cdot[]$, $[]w$, $w[]$, $\mathsf{ev}^1_T[]\ ()$ or $((G(w))^{\text{‘}})[G(w_1)/x_1,\ldots,G(w_{i-1})/x_{i-1},[]/x_i,G(w_{i+1})/x_{i+1},\ldots,G(w_n)/x_n]$.*

Proof: Objects of $\underline{\mathbf{S4}}$ (resp. $\overline{\mathbf{S4}}$) are equivalence classes modulo $\approx$, so to show that G is well-defined on objects of this category, we have to show that if $u\approx v$, then $G(u)=G(v)$. This follows from Theorem 3.9 and Lemmas 3.10 and 3.11.

G is well-defined on morphisms: fix a particular representative for each equivalence class (each object of $\underline{\mathbf{S4}}$, resp. $\overline{\mathbf{S4}}$) by taking their common (gc), (ctract)-normal form, then the definition of G on morphisms is a well-founded definition by structural induction on the representatives.

Finally, the definition defines a unique functor G: for each derivation $u_0\to u_1\to\ldots\to u_n$, its image by G is the concatenation of the G-translations of each step from u_{i-1} to u_i, $1\leq i\leq n$. □

Although G is unique once we have settled for the particular reductions given in Theorems 3.12, 3.27 and in Corollary 3.19, these reductions are not the only ones that prove the corresponding simulation theorems. That is, we have the choice between several possible choices for defining G so that it indeed simulates cut-elimination. This shows that G is not an isomophism of categories. G does not even preserve reductions, and the best we can show (see Part III) is that G defines a conservative extension of λ_{S4} inside $\lambda\mathsf{ev}Q$, in the sense that $G(u)$ and $G(v)$ are interconvertible in $\lambda\mathsf{ev}Q$ if and only if u and v are interconvertible in λ_{S4}.

4 η-Like Rules

4.1 What η-Like Rules Are

We finish this Part II by examining other ways of simplifying proofs in minimal S4. One such way consists in finding *η-like rules* associated with the modal operators. η-like rules come into play to represent the replacement of subproofs (π) ending in tautologous sequents , , $x:\Phi\vdash u:\Phi$ by an instance of (Ax), leading to , , $x:\Phi\vdash x:\Phi$. This produces rules that rewrite some term u where x may occur free into x itself.

The prominent such rule is the η-rule of the λ-calculus: $\lambda y\cdot xy\to x$, where x and y are distinct variables (the general case $\lambda y\cdot uy\to u$, where y is not free in u, follows by applying the substitution $[u/x]$). This rule arises from the case when Φ is an implication, and the subproof (π) ends in an application of ($\Rightarrow L$) followed by an application of ($\Rightarrow R$).

In the $\lambda\mathsf{ev}Q$-calculus, this would mean adding the rules:

$$\begin{array}{ll}(\eta) & \lambda x\cdot ux\to u\ \text{if}\ x\notin\text{fv}(u)\\ (\eta\lambda^\ell) & \lambda^\ell(u\star^\ell 1^\ell)\to v\ \text{if}\ v\circ^\ell\uparrow^\ell\xrightarrow{\circ_\ell}{}^{**}\xleftarrow{\circ_\ell}u\end{array}$$

for example. The $(\eta\lambda^\ell)$ rule is inspired by [Río93] and the work of Thérèse Hardin. (We won't prove anything here on $\lambda\mathsf{ev}Q$ augmented with this rule; although we conjecture that it holds, subject reduction already is not trivial.) We might also consider a variant on Briaud's [Bri95] simpler rule:

$$(\eta'\lambda^\ell)\quad\lambda^\ell(u\star^\ell 1^\ell)\to u\circ^\ell(\perp^\ell\bullet^\ell id^\ell)$$

where $\bot^\ell$ is a new constant of sort T with the new rule: $\bot^\ell \circ^\ell u \to \bot^\ell$; and naturally all rules of the form $\mathsf{ev}^\ell \bot^\ell w \to \bot^{\ell-1}$, $\bot^{\mathcal{L}} \circ^\ell w \to \bot^{\mathcal{L}}$, $\mathsf{ev}^\ell \bot^{\mathcal{L}} w \to \bot^{\mathcal{L}-1}$ and $Q^\ell \bot^{\mathcal{L}-1} \to \bot^{\mathcal{L}}$, for every $1 \leq \ell < \mathcal{L}$. The interpretation of this η-rule as a proof simplification step is then lost, however.

Similarly, in the case of $(\times L)$ followed by $(\times R)$, we get surjective pairing, namely the following rules (for $\lambda\mathsf{ev}Q$):

$$\begin{array}{ll}
(\eta\,\bullet) & (1u \bullet\!\uparrow u) \to u \\
(\eta\,\bullet^\ell) & 1^\ell \bullet^\ell \uparrow^\ell \to id^\ell \\
(\eta\,\bullet\circ^\ell) & (1^\ell \circ^\ell u) \bullet^\ell (\uparrow^\ell \circ^\ell u) \to u
\end{array}$$

And in the case of $(\Box L)$ followed by $(\Box R)$ (the only case that really matters to us here), we wish to replace a proof of the form:

$$\begin{array}{r}
\\ \\ (\Box L) \\ (\Box R)
\end{array}
\dfrac{\dfrac{\begin{array}{c}(\pi)\\ \vdots \\ , , y : \Phi \vdash y : \Phi\end{array}}{, , x : \Box\Phi \vdash \mathsf{unbox}\ x : \Phi}}{, , x : \Box\Phi, , ' \vdash (\mathsf{unbox}\ x)^* : \Box\Phi}$$

where , is a boxed context, by:

$$(Ax)\ \dfrac{}{, , x : \Box\Phi, , ' \vdash x : \Box\Phi}$$

4.2 The λ_{S4_H}-Calculus

To simulate this in λ_{S4}, we extend it as follows:

Definition 4.1 *The λ_{S4_H}-calculus is the λ_{S4}-calculus plus the rule:*

$$(\eta\mathsf{box}) \quad \mathsf{box\ unbox}\ x_i\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n \to v_i$$

for any $1 \leq i \leq n$.

We shall spend the rest of Section 4.2 proving the following:

Theorem 4.1 *Just as for the λ_{S4}-calculus, we have the following properties for the λ_{S4_H}-calculus:*

- *The reduction rules have the Church-Rosser property.*
- *Subject reduction holds.*
- *All typed terms are strongly normalizing.*

To show that the λ_{S4_H}-calculus is confluent, we do exactly as for the λ_{S4}-calculus in Part I, Definitions 5.1 and 5.2. We define a $\lambda_{S4'_H}$-calculus and a $\lambda_{S4'^*_H}$-calculus, which are respectively the $\lambda_{S4}{}'$-calculus and the $\lambda_{S4}{}'^*$-calculus of Part I, plus the $(\eta\mathsf{box})$ reduction rule.

Lemma 4.2 *Let u and v be two $\lambda_{S4'^*_H}$-terms, such that u is well-formed, the weighting of u is decreasing, and u reduces to v in one step. Then:*

(i) v is well-formed.

(ii) $W(u) \geq W(v)$, and $W(u) > W(v)$ unless the contracted redex is of the form

$$\mathsf{unbox}\ (\mathsf{box}\ s\ \mathsf{with}\ t_1, \ldots, t_n\ \mathsf{for}\ z_1, \ldots, z_n)$$

where every z_i, $1 \leq i \leq n$, is a proxy variable.

(iii) The weighting of v is decreasing.

Proof: As in Part I, Lemma 5.3. Claim (i) is trivial, and claim (iii) is proved by the same argument.

The only thing that changes for claim (ii) is when the contracted redex Δ_1 in u is an (ηbox) redex box unbox z_i with $t_1, \ldots, t_n$ for $z_1, \ldots, z_n$. Its weight is that of t_i, plus the sum of the weights of $t_1, \ldots, t_n$, which is always greater than the weight of the contractum t_i. □

Theorem 4.3 *The notion of reduction in* $\lambda_{S4'_H}$ *is strongly normalizing.*

Proof: As for Theorem 5.5 in Part I, using Lemma 4.2 instead of Lemma 5.2, Part I. □

Lemma 4.4 *The* $\lambda_{S4'_H}$*-calculus is confluent.*

Proof: By Theorem 4.3, it is enough to prove that it is locally confluent. The critical pairs that we have not already considered in Lemma 5.6, Part I, are as follows:

- Between (unbox) and (ηbox):

  $$\mathsf{unbox}\ (\mathsf{box}\ \mathsf{unbox}\ x_i\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n)$$

  reduces in one step to (unbox x_i)$[v_1/x_1, \ldots, v_n/x_n]$ by (unbox), but also to unbox v_i by (ηbox). These two terms are equal.

- Between (box) and (ηbox):

  $$\begin{array}{ll}\mathsf{box} & \mathsf{unbox}\ x_i \\ & \mathsf{with}\ v_1, \ldots, v_{j-1}, (\mathsf{box}\ u\ \mathsf{with}\ w_1, \ldots, w_m\ \mathsf{for}\ y_1, \ldots, y_m), v_{j+1}, \ldots, v_n \\ & \mathsf{for}\ x_1, \ldots, x_j, \ldots, x_n\end{array}$$

  If $i \neq j$, then this reduces to v_i by (ηbox), and to:

  $$\begin{array}{ll}\mathsf{box} & \mathsf{unbox}\ x_i \\ & \mathsf{with}\ v_1, \ldots, v_{j-1}, w_1, \ldots, w_m, v_{j+1}, \ldots, v_n \\ & \mathsf{for}\ x_1, \ldots, x_{j-1}, y_1, \ldots, y_m, x_{j+1}, \ldots, x_n\end{array}$$

  by (box); but the latter reduces to the former by (ηbox).

  If $i = j$, it reduces to box u with $w_1, \ldots, w_m$ for $y_1, \ldots, y_m$ by (ηbox), and to:

  $$\begin{array}{ll}\mathsf{box} & \mathsf{unbox}\ (u`) \\ & \mathsf{with}\ v_1, \ldots, v_{i-1}, w_1, \ldots, w_m, v_{i+1}, \ldots, v_n \\ & \mathsf{for}\ x_1, \ldots, x_{i-1}, y_1, \ldots, y_m, x_{i+1}, \ldots, x_n\end{array}$$

  by (box). The latter reduces to:

  $$\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_{i-1}, w_1, \ldots, w_m, v_{i+1}, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_{i-1}, y_1, \ldots, y_m, x_{i+1}, \ldots, x_n$$

  by (unbox), hence to box u with $w_1, \ldots, w_m$ for $y_1, \ldots, y_m$ by $n-1$ applications of (gc), since by the well-formedness constraints, the free variables of u are among $y_1, \ldots, y_m$. This completes the confluence diagram.

- Between (box) and (ηbox) again:

  $$\begin{array}{ll}\mathsf{box}\ u & \mathsf{with}\ v_1, \ldots, v_{i-1}, (\mathsf{box}\ \mathsf{unbox}\ y_j\ \mathsf{with}\ w_1, \ldots, w_m\ \mathsf{for}\ y_1, \ldots, y_m), v_{i+1}, \ldots, v_n \\ & \mathsf{for}\ x_1, \ldots, x_i, \ldots, x_n\end{array}$$

  reduces in one step to:

  $$\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_{i-1}, w_j, v_{i+1}, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n$$

by (ηbox), or to:

$$\begin{array}{ll}\textsf{box}\ u[\textsf{box unbox}\ z\ \textsf{with}\ y_j\ \textsf{for}\ z/x_i] & \textsf{with}\ v_1,\ldots,v_{i-1},w_1,\ldots,w_m,v_{i+1},\ldots,v_n \\ & \textsf{for}\ x_1,\ldots,x_{i-1},y_1,\ldots,y_m,x_{i+1},\ldots,x_n\end{array}$$

by (box).

But the latter reduces by (ηbox) to:

$$\textsf{box}\ u[y_j/x_i]\ \textsf{with}\ v_1,\ldots,v_{i-1},w_1,\ldots,w_m,v_{i+1},\ldots,v_n\ \textsf{for}\ x_1,\ldots,x_{i-1},y_1,\ldots,y_m,x_{i+1},\ldots,x_n$$

then by $m-1$ applications of (gc) to:

$$\textsf{box}\ u[y_j/x_i]\ \textsf{with}\ v_1,\ldots,v_{i-1},w_j,v_{i+1},\ldots,v_n\ \textsf{for}\ x_1,\ldots,x_{i-1},y_j,x_{i+1},\ldots,x_n$$

since y_1, …, y_{j-1}, y_{j+1}, …, y_m are not free in $u[y_j/x_i]$. This closes the confluence diagram, using α-renaming.

- Between (ηbox) and (gc):

  $$\textsf{box unbox}\ x_i\ \textsf{with}\ v_1,\ldots,v_n\ \textsf{for}\ x_1,\ldots,x_n$$

  reduces in one step either to v_i by (ηbox), or to:

  $$\textsf{box unbox}\ x_i\ \textsf{with}\ v_1,\ldots,v_{j-1},v_{j+1},\ldots,v_n\ \textsf{for}\ x_1,\ldots,x_{j-1},x_{j+1},\ldots,x_n$$

  by (gc), where $j \neq i$. The latter then reduces to v_i again, by (ηbox).

- Between (ηbox) and (ctract). If $j \neq k$ and $v_j = v_k$, then:

  $$\textsf{box unbox}\ x_i\ \textsf{with}\ \ldots,v_i,\ldots,v_j,\ldots,v_k,\ldots\ \textsf{for}\ \ldots,x_i,\ldots,x_j,\ldots,x_k,\ldots$$

  reduces in one step to v_i by (ηbox).

  If $i \neq k$, then it also reduces in one step to:

  $$\textsf{box unbox}\ x_i\ \textsf{with}\ \ldots,v_i,\ldots,v_j,\ldots,\ldots\ \textsf{for}\ \ldots,x_i,\ldots,x_j,\ldots,\ldots$$

  by (ctract), which reduces to v_i by (ηbox).

  If $i = k$, then it reduces instead to:

  $$\textsf{box unbox}\ x_j\ \textsf{with}\ \ldots,v_i,\ldots,v_j,\ldots,\ldots\ \textsf{for}\ \ldots,x_i,\ldots,x_j,\ldots,\ldots$$

  by (ctract). This reduces to v_j by (ηbox). But by assumption $v_j = v_k$, and because $i = k$, $v_j = v_i$. Again, the confluence diagram closes.

□

Theorem 4.5 *The $\lambda_{\mathrm{S4}H}$-calculus is confluent.*

Proof: As for Theorem 5.7, Part I, using Theorem 4.3 instead of Theorem 5.5, Part I, and Lemma 4.4 instead of Lemma 5.6, Part I. □

Theorem 4.6 *Subject reduction holds in $\lambda_{\mathrm{S4}H}$.*

Proof: Compared with Theorem 5.8, Part I, we only have to check it for rule (ηbox). But again, this is a trivial consequence of the identification of proofs and typed terms. □

Theorem 4.7 *All typed $\lambda_{\mathrm{S4}H}$-terms are strongly normalizing.*

(H) $(1 \leq \ell)$:

$(\eta \mathtt{ev}^\ell)$	$\mathtt{ev}^{\ell+1}(Q^\ell u)w \to u \circ^\ell w$
$(\eta \Uparrow^\ell)$	$\Uparrow^\ell w \to 1^\ell \bullet^\ell (w \circ^\ell \uparrow^\ell)$
$(\eta \bullet)$	$(1u \bullet \uparrow u) \to u$
$(\eta \bullet^\ell)$	$1^\ell \bullet^\ell \uparrow^\ell \to id^\ell$
$(\eta \bullet \circ^\ell)$	$(1^\ell \circ^\ell u) \bullet^\ell (\uparrow^\ell \circ^\ell u) \to u$

Figure 9: Modal η-like reduction rules

Proof: As in Theorem 5.9, Part I, define the following erasing transformation by structural induction on the terms:

$$D(\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n) = D(u)[D(v_1)/x_1, \ldots, D(v_n)/x_n]$$
$$D(\mathsf{unbox}\ u) = D(u)$$

and D does nothing on other constructions. Define also the erasing transformation $D(\Phi)$ on formulas Φ, by erasing all boxes $\Box$, and similarly on contexts , .

Now, let $u_1 \longrightarrow u_2 \longrightarrow \ldots \longrightarrow u_i \longrightarrow \ldots$ be a reduction starting from the well-typed λ_{S4}-term u_1. Then $D(u_1) \overset{=}{\longrightarrow} D(u_2) \overset{=}{\longrightarrow} \ldots \overset{=}{\longrightarrow} D(u_i) \overset{=}{\longrightarrow} \ldots$ in the simply-typed λ-calculus, where $\overset{=}{\longrightarrow}$ is the reflexive closure of reduction in this calculus; again, every contraction by (β) translates by D into a contraction by the same rule, and if $u_i \longrightarrow u_{i+1}$ by some other rule, then $D(u_i) = D(u_{i+1})$.

The theorem then follows by strong normalization of the simply-typed λ-calculus and Theorem 4.3. $\Box$

4.3 The $\lambda\mathtt{ev}Q_H$-Calculus

Similarly, we can extend the $\lambda\mathtt{ev}Q$-calculus with modal η-like rules. Translating rule $(\eta\mathsf{box})$ above, we see that we wish to have a new rule $\mathtt{ev}^2(Q^1 u)id^1 \to u$. To ensure that reduction under quotes still works, we have to generalize this to $\mathtt{ev}^{\ell+1}(Q^\ell u)id^\ell \to u$ for every $\ell \geq 1$.

We now find a few critical pairs. First, $\mathtt{ev}^{\ell+1}(Q^\ell u)id^\ell \circ^\ell v$ must reduce to $u \circ^\ell v$ by the above, but it also reduces to $\mathtt{ev}^{\ell+1}(Q^\ell u)v$ by $(\mathtt{ev}^{\ell+1}\circ^\ell)$, $(Q\circ^\ell)$ and $(id\circ^\ell)$. So we add the following rule:

$$\mathtt{ev}^{\ell+1}(Q^\ell u)w \to u \circ^\ell w$$

In turn, this rule generates new critical pairs. In particular, $\mathtt{ev}^{\ell+1}(Q^\ell(\lambda^\ell id^\ell))w$ reduces to $\lambda^\ell(\Uparrow^\ell w)$ by the latter rule, and rules (λ^ℓ) and $(id\circ^\ell)$; it also reduces to $\lambda^\ell(1^\ell \bullet^\ell (w\circ^\ell \uparrow^\ell))$ by $(Q^\ell\lambda^{\ell+1})$, $(\mathtt{ev}\lambda^{\ell+1})$, the new rule and $(id\circ^\ell)$. This suggests the rule $\Uparrow^\ell w \to 1^\ell \bullet^\ell (w\circ^\ell \uparrow^\ell)$. This, in turn, forces us to have surjective pairing: for instance, $\Uparrow^\ell id^\ell$ reduces to id^ℓ (by $(\Uparrow id^\ell)$) and to $1^\ell \bullet^\ell \uparrow^\ell$ (by the new rule on $\Uparrow^\ell$ and $(id\circ^\ell)$).

Therefore:

Definition 4.2 *We define the $\lambda\mathtt{ev}Q_H$-calculus by the reduction rules of the $\lambda\mathtt{ev}Q$-calculus (Figures 3 and 4), plus group (H) (Figure 9).*

The rules of group (H) will pose several technical problems. First, the termination of the rules of the $\lambda\mathtt{ev}Q_H$-calculus except (β) and (β^ℓ) is not at all obvious. Because of rule $(\eta \Uparrow^\ell)$, the $\lambda\mathtt{ev}Q_H$-calculus is not a variation on the $\lambda\sigma_\Uparrow$-calculus any longer, but rather on the $\lambda\sigma$-calculus. The rules in group (B) except (β^ℓ), plus the rules in group (H) except $(\eta\mathtt{ev}^\ell)$, with ℓ fixed, already form a group of rules that propagate substitutions downward in the $\lambda\sigma$-calculus; that they terminate is already a difficult result. (For this particular case, we can use Zantema's distribution elimination technique [Zan94], though.)

Second, confluence will also be hard to establish. Our system can be put in parallel with Ríos' $\lambda\sigma'$-calculus (see [Río93], Section 1.7), which is not confluent on terms with substitution (stack) variables [CHL95]. It is precisely the purpose of having terms of two disjoint sorts, T and S, to be able to forbid stack variables inside terms, just as in the $\lambda\sigma$-calculus. In fact, we need more to get confluence, and quite probably to consider only typed terms. Notice that we cannot forbid variables altogether (i.e. consider a ground rewrite system), because (bound) term variables are introduced by the $\lambda x\cdot$ abstraction operator.

Rule	Assumptions	Type
Group (H)		
$(\eta\mathsf{ev}^\ell)$	$u : \overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma_1 \overset{\Box}{\Rightarrow} \Phi$ $w : \overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma_3 \overset{\Box}{\Rightarrow} \varsigma_1$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma_3 \overset{\Box}{\Rightarrow} \Phi$
$(\eta \Uparrow^\ell)$	$u : \overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma_1 \overset{\Box}{\Rightarrow} \varsigma_2$	$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\tau \times \varsigma_1 \overset{\Box}{\Rightarrow} \tau \times \varsigma_2$
$(\eta \bullet)$	$u : \tau \times \varsigma$	$\tau \times \varsigma$
$(\eta \bullet^\ell)$ rhs:		$\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\tau \times \varsigma \overset{\Box}{\Rightarrow} \tau \times \varsigma$ $\overline{\varsigma'^{\ell-1} \overset{\Box}{\Rightarrow}}\varsigma' \overset{\Box}{\Rightarrow} \varsigma'$
$(\eta \bullet \circ^\ell)$	$u : \overline{\varsigma'^{\ell} \overset{\Box}{\Rightarrow}}\tau \times \varsigma$	$\overline{\varsigma'^{\ell} \overset{\Box}{\Rightarrow}}\tau \times \varsigma$

Figure 10: Checking Subject Reduction (group (H))

Theorem 4.8 (Subject Reduction) *Let u be a term of the $\lambda\mathsf{ev}Q_H$-calculus.*
If $\Gamma \vdash u : \Phi$ is derivable, and $u \longrightarrow^* v$ in the $\lambda\mathsf{ev}Q_H$-calculus, then $\Gamma \vdash v : \Phi$ is derivable.

Proof: As for Theorem 3.2. The table corresponding to the rules in group (H) is shown in Figure 10. $\Box$

We extend Theorem 3.27:

Lemma 4.9 *If $x_1 : \Box\Phi_1, \ldots, x_n : \Box\Phi_n \vdash u : \Box\Phi$, $\Gamma \vdash v_1 : \Box\Phi_1$, $\ldots$, $\Gamma \vdash v_n : \Box\Phi_n$ are derivable (in the system of Figure 4, Part I), and:*

$$\mathsf{box}\ \mathsf{unbox}\ x_i\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n \to v_i$$

by rule $(\eta\mathsf{box})$ for some i, $1 \leq i \leq n$, then $G(\mathsf{box}\ \mathsf{unbox}\ x_i\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n)$ reduces to $G(v_i)$ by rule $(\eta\mathsf{ev}^1)$ in $\lambda\mathsf{ev}Q_H$.

Proof: That the types check follows from Theorem 3.28: both G-translations then have types $\top \overset{\Box}{\Rightarrow} G(\Phi)$. As far as the terms are concerned, we have:

$$\begin{aligned} G(\mathsf{box}\ \mathsf{unbox}\ x_i\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n) &= \mathsf{ev}^2(Q^1 x_i)[G(v_1)/x_1, \ldots, G(v_n)/x_n])id^1 \\ &= \mathsf{ev}^2(Q^1(G(v_i)))id^1 \end{aligned}$$

which reduces in one $(\eta\mathsf{ev}^1)$ step to $G(v_i)$. $\Box$

Theorem 3.18 also extends without any problem:

Lemma 4.10 *For all $\lambda\mathsf{ev}Q_H$-terms u and v, if $u \longrightarrow v$ by some rule in group (H), then $u‘\rho \longrightarrow v‘\rho$ by some other rule in group (H), for all environments ρ.*

Proof: As in Theorem 3.18, $R‘$ is $(\eta\mathsf{ev}^{\ell+1})$ if R is $(\eta\mathsf{ev}^\ell)$, $(\eta \Uparrow^{\ell+1})$ if R is $(\eta \Uparrow^\ell)$, $(\eta \bullet \circ^1)$ if R is $(\eta \bullet)$, $(\eta \bullet^{\ell+1})$ if R is $(\eta \bullet^\ell)$, and $(\eta \bullet \circ^{\ell+1})$ if R is $(\eta \bullet \circ^\ell)$. $\Box$

From which the analogue of Corollary 3.19 follows, hence also that of Theorem 3.29:

Theorem 4.11 *Let $\underline{\mathbf{S4}}_H$ (resp. $\overline{\mathbf{S4}}_H$) be the category whose objects are those of $\underline{\mathbf{S4}}$ (resp. $\overline{\mathbf{S4}}$; see Definition 5.8, Part I), and whose morphisms are all morphisms in $\underline{\mathbf{S4}}$ (resp. $\overline{\mathbf{S4}}$), all reductions by rule $(\eta\mathsf{box})$, and all compositions thereof.*

Let $\lambda\mathsf{ev}Q_H$ be the category whose objects are typed $\lambda\mathsf{ev}Q$-terms, with morphisms all reductions of the $\lambda\mathsf{ev}Q_H$-calculus.

We extend the action of G (see Theorem 3.29) to $\underline{\mathbf{S4}}_H$ (resp. $\overline{\mathbf{S4}}_H$) morphisms by letting $G(u \overset{(\eta\mathsf{box})}{\longrightarrow} v)$ be defined as the reduction from $G(u)$ to $G(v)$ given by Lemma 4.9.

Then G is a functor from $\underline{\mathbf{S4}}_H$ (resp. $\overline{\mathbf{S4}}_H$) to $\lambda\mathsf{ev}Q_H$.

Proof: G is well-defined: the proof is the same as for Theorem 3.29, using Lemma 4.10 instead of Corollary 3.19. That it is a functor is immediate. $\Box$

Finally, note that the $\lambda\mathsf{ev}Q_H$-calculus really uses many unneeded operators. We might replace rule $(\eta \Uparrow^\ell)$ by simply replacing every term of the form $\Uparrow^\ell u$ by $1^\ell \bullet^\ell (u\circ^\ell \uparrow^\ell)$, and replace group (B) by a variant of the $\lambda\sigma$-calculus instead of the $\lambda\sigma_\Uparrow$-calculus. Rule $(\eta\mathsf{ev}^\ell)$ may also be dispensed with by letting $u \circ^\ell v$ be an abbreviation of $\mathsf{ev}^{\ell+1}(Q^\ell u)v$. Some rules (like $(\mathsf{ev}^\ell 1^\ell)$) have to be eliminated, and we end up with a mostly unreadable system.

5 Other Ways of Simplifying Proofs

There are other ways of eliminating cuts or, in general, of simplifying proofs. We explore how we may simulate these transformations by adding corresponding rules to λ_{S4} or to $\lambda\mathsf{ev}Q$. Let the reader be warned that the results are not pretty.

5.1 Eliminating $(\Box L)/(\Box R)$ Cuts

Consider cuts of the form:

$$\begin{array}{lccc}
 & (\pi'_1) & & (\pi'_2)\\
 & \vdots & & \vdots\\
 & ,\ _1, y_1 : \Psi_2 \vdash u_1 : \Psi_1 & & ,\ _2, x : \Psi_1 \vdash u : \Phi \\ \hline
(\Box L) & ,\ _1, x_1 : \Box\Psi_2 \vdash u_1[\mathsf{unbox}\ x_1/y_1] : \Psi_1 & (\Box R) & ,\ _2, ,\ '_2, x : \Psi_1 \vdash u^{‘} : \Box\Phi \\ \hline
(Cut) & \multicolumn{3}{c}{,\ _1, x_1 : \Box\Psi_2, ,\ _2, ,\ '_2 \vdash u^{‘}[u_1[\mathsf{unbox}\ x_1/y_1]/x] : \Box\Phi}
\end{array}$$

where , is boxed, and Ψ_1 is inactive both in the $(\Box L)$ and in the $(\Box R)$ rule. When all free variables in u_1 are of boxed type, except possibly y_1, we can consider that $,\ _1$ is a boxed context, and rewrite the latter proof into:

$$\begin{array}{lc}
 & (\pi'_1) \qquad\qquad\qquad (\pi'_2)\\
 & \vdots \qquad\qquad\qquad\qquad \vdots\\
 & ,\ _1, y_1 : \Psi_2 \vdash u_1 : \Psi_1 \qquad\qquad ,\ _2, x : \Psi_1 \vdash u : \Phi \\
(Cut) & ,\ _1, y_1 : \Psi_2, ,\ _2 \vdash u[u_1/x] : \Phi \\ \hline
(\Box L) & ,\ _1, x_1 : \Box\Psi_2, ,\ _2 \vdash u[u_1/x][\mathsf{unbox}\ x_1/y_1] : \Phi \\ \hline
(\Box R) & ,\ _1, x_1 : \Box\Psi_2, ,\ _2, ,\ '_2 \vdash (u[u_1/x][\mathsf{unbox}\ x_1/y_1])^{‘} : \Box\Phi
\end{array}$$

This way of eliminating cuts can be implemented in λ_{S4}, by adding the following rule:

$$(R_1) \qquad \mathsf{box}\ u\ \mathsf{with}\ \ldots, \mathsf{unbox}\ v, \ldots\ \mathsf{for}\ \ldots, x, \ldots \to \mathsf{box}\ u[\mathsf{unbox}\ y/x]\ \mathsf{with}\ \ldots, v, \ldots\ \mathsf{for}\ \ldots, y, \ldots$$

where y is a fresh variable.

We do not know how to do it $\lambda\mathsf{ev}Q$. First, we have to be careful: without the difference between elementary terms and stacks, we would get an inconsistent calculus. So here the T and S annotations matter. Indeed, we might be tempted to use the following argument: By (R_1), $\mathsf{box}\ x\ \mathsf{with}\ \mathsf{unbox}\ x_1\ \mathsf{for}\ x$ reduces to $(\mathsf{unbox}\ x_1)^{‘}$. Translating this by G, $Q^1(\mathsf{ev}^1 x_1())$ must reduce to $\mathsf{ev}^2(Q^1 x_1)id^1$. Applying the substitution $[u/x_1]$, it follows that $Q^1(\mathsf{ev}^1 u())$ must reduce to $\mathsf{ev}^2(Q^1 u)id^1$, whatever the term u. Let w be an arbitrary stack, apply $\mathsf{ev}^1\_\ w$ on both sides and reduce: it follows that $\mathsf{ev}^1 u()$ and $\mathsf{ev}^1 uw$ must be convertible. Choose u to be id^1, and reduce: $()$ and w must be convertible. As w is arbitrary, all terms must be equal. What we have overlooked in this argument is that the ev^1, ev^2 and Q^1 above are really indexed by T, and that u must be of sort T: therefore, id^1 cannot be substituted for it.

But the main problem is with the typed version of the calculus. Indeed, there is no way to orient the untyped critical pair $Q^1(\mathsf{ev}^1 u()) \overset{?}{=} \mathsf{ev}^2(Q^1 u)id^1$ so that it obeys subject reduction: the most general type of the left-hand side is indeed $\varsigma_1 \overset{\Box}{\Rightarrow} \varsigma_2 \overset{\Box}{\Rightarrow} \Phi_3$ assuming $u : \top \overset{\Box}{\Rightarrow} \varsigma_2 \overset{\Box}{\Rightarrow} \Phi_3$, and that of the right-hand side is $\varsigma_1 \overset{\Box}{\Rightarrow} \Phi_4$, assuming $u : \varsigma_1 \overset{\Box}{\Rightarrow} \Phi_4$.

5.2 Eliminating Other Cuts

The pecularity that made the $(\Box L)/(\Box R)$ cut elimination legal in the last subsection was that $\mathsf{unbox}\ x_1$, the term which is to be substituted for y_1, contains only one variable, x_1, and that the latter must be of boxed type. This is why the cut can be pushed over the $(\Box R)$ rule, keeping x_1 in the context above the $(\Box R)$ inference. The term $\mathsf{unbox}\ x_1$ is a product of using rule $(\Box L)$ as the vis-a-vis of $(\Box R)$. We cannot do the same with any other rule with an inactive right-hand side — namely, (Cut) or $(\Rightarrow L)$.

However, there is another case where we may permute (Cut) with $(\Box R)$, and this is when the proof is:

$$
\begin{array}{c}
\begin{array}{cc}
(\pi_1) & (\pi_2') \\
\vdots & \vdots \\
 & \Gamma \vdash u : \Phi \\
\cline{2-2}
\Gamma_1 \vdash u_1 : \Psi_1 \quad (\Box R) & \Gamma, \Gamma', x : \Psi_1 \vdash u^{‘} : \Box\Phi
\end{array} \\
\hline
(Cut) \quad \Gamma_1, \Gamma, \Gamma' \vdash u^{‘}[u_1/x] : \Box\Phi
\end{array}
$$

where Γ is boxed *and* Γ_1 *is boxed*. Then, we may transform this into the following:

$$
\begin{array}{c}
\begin{array}{cc}
(\pi_1) & x : \Psi_1, (\pi_2') \\
\vdots & \vdots \\
\Gamma_1 \vdash u_1 : \Psi_1 & \Gamma, x : \Psi_1 \vdash u : \Phi
\end{array} \\
\hline
(Cut) \quad \Gamma, \Gamma_1 \vdash u[u_1/x] : \Phi \\
\hline
(\Box R) \quad \Gamma, \Gamma_1, \Gamma' \vdash (u[u_1/x])^{‘} : \Box\Phi
\end{array}
$$

We have not considered this a reasonable cut-elimination step from the computational standpoint in part I of this paper (see case 2 of cut eliminations, Section 4.1.2 of Part I); since terms identify proofs modulo weakening and permutation of cuts, this rule really means that when Γ_1 is not boxed, we should explore the whole proof (π_1) to see whether (π_1) is in fact the weakening of some other proof which only uses boxed assumptions in Γ_1.

Representing this in λ_{S4} or in $\lambda\mathsf{ev}Q$ is therefore rather arduous.

5.3 Souped-Up η-Like Rules

We have seen in Section 4 that we could introduce an η-like rule by adding rule $(\eta\mathsf{box})$ to the λ_{S4}-calculus, or the rules of group (H) to the $\lambda\mathsf{ev}Q$-calculus.

But rule $(\eta\mathsf{box})$ looks contrived, in that the boxed term on the left-hand side must be of the form $\mathsf{unbox}\ x_i$ for a variable x_i. It seems more natural to extend it to the following rule, which we call the souped-up η-like rule:

$$\mathsf{box}\ \mathsf{unbox}\ u\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n \to u[v_1/x_1, \ldots, v_n/x_n]$$

which obeys subject reduction as well.

But adding this rule to the λ_{S4}-calculus breaks local confluence. Consider indeed the term $\mathsf{box}\ \mathsf{unbox}\ (\mathsf{box}\ u\ \mathsf{with}\ \sigma)\ \mathsf{with}\ \sigma'$. By $(\eta\mathsf{box})$, this reduces to $(\mathsf{box}\ u\ \mathsf{with}\ \sigma)\sigma'$, that is $\mathsf{box}\ u\ \mathsf{with}\ \sigma\sigma'$, where $\sigma\sigma'$ is the composition of σ and σ'. On the other hand, it also reduces by $(\mathsf{unbox}\ )$ to $\mathsf{box}\ u\sigma\ \mathsf{with}\ \sigma'$.

Now, take u to be a variable x, σ to be $[yz/x]$ and σ' to be $[y'/y, z'/z]$. Then the critical pair is $\mathsf{box}\ x\ \mathsf{with}\ y'z'\ \mathsf{for}\ x \overset{?}{=} \mathsf{box}\ yz\ \mathsf{with}\ y', z'\ \mathsf{for}\ y, z$, and both sides are normal but not equal modulo α-conversion and $\sim$.

Joining the resulting critical pairs (in the untyped case) means adding the rule:

$$\mathsf{box}\ u\ \mathsf{with}\ \sigma \to (u\sigma)^{‘}$$

which is quite unfortunate, since it entails that substitutions are allowed to go through the box barriers. The latter would not bar anything, which means that the typed version of the calculus cannot obey subject reduction (see the discussion of the λ’‘-calculus in part I). The resulting untyped calculus could also be simplified: it would be an extension of the λ-calculus with two new constants unbox and box, such that $\mathsf{box}\ (\mathsf{unbox}\ u) \to u$ and $\mathsf{unbox}\ (\mathsf{box}\ u) \to u$.

Therefore, any (locally) confluent calculus extending λ_{S4} and containing the souped-up η-like rule above must include type annotations.

The situation for the $\lambda\mathsf{ev}Q$-calculus is even worse, as it includes the cases of the previous sections. Indeed, from our considerations on the λ_{S4}-calculus with the souped-up η-like rule, we have the following critical pair:

$$\mathsf{box}\ u\ \mathsf{with}\ \sigma\sigma' \overset{?}{=} \mathsf{box}\ u\sigma\ \mathsf{with}\ \sigma'$$

Take σ' to be a substitution $[v_1/x_1, \ldots, v_n/x_n]$, and a variable x that is not free in any v_i, $1 \leq i \leq n$. Let y be some x_i, $1 \leq i \leq n$, and take σ to be $[\mathsf{unbox}\ y/x]$. The critical pair becomes:

$$\mathsf{box}\ u\ \mathsf{with}\ v_1, \ldots, \mathsf{unbox}\ v_i, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n, x \overset{?}{=} \mathsf{box}\ u[\mathsf{unbox}\ x_i/x]\ \mathsf{with}\ v_1, \ldots, v_n\ \mathsf{for}\ x_1, \ldots, x_n$$

which is an unoriented version of rule $(R1)$.

There are so many critical pairs, and most of them are non-orientable, that we shan't consider these souped-up rules.

References

[ACCL90] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. In *Proceedings of the 17th Annual ACM Symposium on Principles of Programming Languages*, pages 31–46, San Francisco, California 1990. January.

[Bar84] Henk Barendregt. *The Lambda Calculus, Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, Amsterdam, 1984.

[Bri95] Daniel Briaud. An explicit *eta* rewrite rule. In M. Dezani-Ciancaglini and G. Plotkin, editors, *2nd International Conference on Typed Lambda-Calculi and Applications (TLCA'95)*, pages 94–108, Edinburgh, UK, April 1995. Springer Verlag LNCS 902.

[CHL95] Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM*, 1995. To appear; INRIA report #1617, 1992.

[Cur86] Pierre-Louis Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming*. Pitman, London, 1986.

[Hin69] J. R. Hindley. The principal type scheme of an object in combinatory logic. *Transations of the American Mathematical Society*, 146:29–60, 1969.

[HL89] Thérèse Hardin and Jean-Jacques Lévy. A confluent calculus of substitutions. In *France-Japan Artificial Intelligence and Computer Science Symposium*, December 1989.

[JK90] Jean-Pierre Jouannaud and Claude Kirchner. Solving equations in abstract algebras: a rule-based survey of unification. Technical report, LRI, CNRS UA 410: Al Khowarizmi, March 1990.

[Klo80] Jan Willem Klop. *Combinatory Reduction Systems*. Number 27 in Mathematical Center Tracts. Centrum voor Wiskunde en Informatica, 1980.

[Mil78] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

[MTH90] Robin Milner, Mads Tofte, and Robert Harper. *The Definition of Standard ML*. MIT Press, 1990.

[Río93] Alejandro Ríos. *Contributions à l'étude des lambda-calculs avec substitutions explicites*. PhD thesis, École Normale Supérieure, December 1993.

[Zan94] Hans Zantema. Termination of term rewriting: Interpretation and type elimination. *Journal of Symbolic Computation*, 17:23–50, 1994.