Justification Logic and History Based Computation[⋆]

Francisco Bavera[1] and Eduardo Bonelli[2]

[1] UNRC and CONICET, Argentina (fbavera@gmail.com)
[2] UNQ and CONICET, Argentina (eabonelli@gmail.com)

Abstract. Justification Logic (JL) is a refinement of modal logic that has recently been proposed for explaining well-known paradoxes arising in the formalization of Epistemic Logic. Assertions of knowledge and belief are accompanied by justifications: the formula $[\![t]\!]A$ states that t is "reason" for knowing/believing A. We study the computational interpretation of JL via the Curry-de Bruijn-Howard isomorphism in which the modality $[\![t]\!]A$ is interpreted as: t is a type derivation justifying the validity of A. The resulting lambda calculus is such that its terms are aware of the reduction sequence that gave rise to them. This serves as a basis for understanding systems, many of which belong to the security domain, in which computation is history-aware.

1 Introduction

This paper is concerned with the computational interpretation of *Justification Logic* [Art95,Art01,Art08] (**JL**). **JL** is a refinement of modal logic that has recently been proposed for explaining well-known paradoxes arising in the formalization of Epistemic Logic. Assertions of knowledge and belief are accompanied by *justifications*: the modality $[\![t]\!]A$ states that t is "reason" for knowing/believing A. The starting point of this work is the observation that if t is understood as a typing derivation of a term of type A, then a term of type $[\![t]\!]A$ should incorporate some encoding of t. Suppose this typing derivation is seen as a logical derivation in natural deduction. Then any normalisation steps applied to it would produce a new typing derivation for A and, moreover, its relation to t would have to be made explicit in order for derivations to be closed under normalisation (in type systems parlance: for Subject Reduction (SR) to hold). This suggests that the computational interpretation of **JL** is a lambda calculus, which we dub $\lambda^{\mathfrak{h}}$, that records its computation history. This work is an attempt at making these ideas precise.

We begin with some examples supplying informal explanations whenever appropriate (rigorous definitions must wait until the necessary background has been introduced). The expression $!_e^{\alpha_1,\ldots,\alpha_n} M$ is called an *audited (computation) unit*, M being the *body*, e the *history* or *trail* of computation producing M and α_i, $i \in 1..n$, the *trail variables* that are used for consulting the history. Each

[⋆] Work partially supported by ANPCyT PICT 2006-01216 and ITBA.

reduction step that takes place in M updates e accordingly (except if this step is inside a nested audited unit). Consider the unit $!^{\alpha}_{\mathit{Rfl}(s)}(\lambda a : \mathbb{N}.a)\, b$. Its body consists of the identity function over the type $\mathbb{N}$ of the natural numbers applied to a variable b; $\mathit{Rfl}(s)$ is the empty trail with s being the encoding of a type derivation of $(\lambda a : \mathbb{N}.a)\, b$; the trail variable α plays no role in this example. This term reduces to $!^{\alpha}_{\mathit{Trn}(\beta(a^{\mathbb{N}}.a,b),\mathit{Rfl}(s))} b$. The new trail $\mathit{Trn}(\beta(a^{\mathbb{N}}.a, b), \mathit{Rfl}(s))$ indicates that a β step took place at the root; the *Trn* trail constructor indicates composition of trails.

Inspection of trails is achieved by means of trail variables. These variables are affine (i.e. at most one permitted use) since each trail lookup may produce a different result. Evaluation of trail variables inside an audited unit consists in first looking up the trail and then immediately traversing it replacing each constructor of the trail with a term of the appropriate type[3]. This mapping from trail constructors to terms is called a *trail replacement*. All occurrences of trail variables are thus written $\alpha\theta$ where α is a trail variable and θ a trail replacement. For example, suppose after a number of computation steps we attain the term $!^{\alpha}_{e}\mathsf{if}\ \alpha\theta > \underline{5}\ \mathsf{then}\ \underline{1}\ \mathsf{else}\ \underline{2}$, where e denotes the current history and $\underline{n}$ a numeral. Given the following definition of θ, $\alpha\theta$ counts the number of β steps that have taken place (expressions such as *Tlk* below are other trail constructors and may be ignored for the moment):

$$\begin{array}{rlrl}
\theta(\mathit{Rfl}) = \theta(\mathit{Tlk}) & \overset{\text{def}}{=} \underline{0} & \theta(\mathit{Rpl}) & \overset{\text{def}}{=} \lambda\overline{a} : \mathbb{N}.a_1 + \ldots + a_{10} \\
\theta(\mathit{Sym}) = \theta(\mathit{Abs}) & \overset{\text{def}}{=} \lambda a : \mathbb{N}.a & \theta(\beta_{\Box}) & \overset{\text{def}}{=} \underline{0} \\
\theta(\mathit{Trn}) = \theta(\mathit{App}) = \theta(\mathit{Let}) & \overset{\text{def}}{=} \lambda a : \mathbb{N}.\lambda b : \mathbb{N}.a + b & \theta(\beta) & \overset{\text{def}}{=} \underline{1}
\end{array}$$

Thus, the term decides either to compute $\underline{1}$ or $\underline{2}$ depending on whether the number of β steps that have taken place is greater than 5 or not. An interesting feature of $\lambda^{\natural}$ is how it manages persistence of trails. It is achieved by means of the $\mathsf{let}\, u = M\, \mathsf{in}\ N$ construct (M is the argument and N the body of the let) which eliminates audited units and operates as exemplified below. Let P be $\lambda a : \mathbb{N}.\mathsf{if}\ \alpha\theta > \underline{5}\ \mathsf{then}\ a\ \mathsf{else}\ \underline{2}$ and consider the following term where the expression $\langle u; \{\alpha/\gamma\}\rangle$ consists of an audited unit variable u and a trail variable renaming $\{\alpha/\gamma\}$:

$$\mathsf{let}\, u = !^{\alpha}_{e_1} P\ \mathsf{in}\ !^{\gamma}_{\mathit{Rfl}(t)}\langle u; \{\alpha/\gamma\}\rangle\, \underline{1}$$

In a $\beta_{\Box}$ reduction step, first u is replaced by the body P of the audited unit $!^{\alpha}_{e_1} P$ together with its history e_1, then all occurrences of trail variables α are replaced by γ, and finally e_1 is merged with the trail of the new host unit:

$$!^{\gamma}_{\mathit{Trn}(e_2,\mathit{Rfl}(t'))}\, (\lambda a : \mathbb{N}.\mathsf{if}\ \gamma\theta > \underline{5}\ \mathsf{then}\ a\ \mathsf{else}\ \underline{2})\, \underline{1}$$

Trail e_2 is $\mathit{Trn}(\beta_{\Box}(u.r_1, \alpha.r_2), \mathit{App}(e'_1, \mathit{Rfl}(s)))$. Here $u.r_1$ and $\alpha.r_2$ are encodings of typing derivations for the body and argument of the let resp., e'_1 is e_1 updated with $\{\alpha/\gamma\}$, s encodes a type derivation for $\underline{1}$ and t' is t where all occurrences of u have been replaced by the $\alpha.r_2$. The $\beta_{\Box}$ trail construct reflects a reduction of a

[3] In the same way as one recurs over lists using fold in functional programming, replacing nil and cons by appropriate terms.

let redex at the root. Note how, (1) the history of the unit $!^{\alpha}_{e_1} P$ has propagated to the new host unit (as the trail $App(e_1', Rfl(s))$ which reflects that activity described by e_1' has taken place in the left argument of the application $(\lambda a : \mathbb{N}.\mathsf{if}\ \gamma\theta > \underline{5}\ \mathsf{then}\ a\ \mathsf{else}\ \underline{2})\ \underline{1}$), and (2) how the trail variable α has been replaced by γ so that all subsequent trail lookups now correctly refer to the trail of the new host unit. All these operations arise from an analysis of the normalisation of **JL** derivations.

The contributions of this paper are a proof theoretical analysis of a λ-calculus which produces a trail of its execution. This builds on ideas stemming from work on **JL**, judgemental analysis of modal logic [ML96,DP96,DP01b,DP01a] and Contextual Modal Type Theory [NPP08]. More precisely, we argue how a fragment of **JL** whose notion of validity is relative to a context of affine variables of justification equivalence may be seen, via the Curry-de Bruijn-Howard interpretation, as a type system for a calculus that records its computation history.

Related work. S. Artemov introduced **JL** in [Art95,Art01,Art08]. For natural deduction and sequent calculus presentations consult [Art01,Bre01,AB07]. Computational interpretation of proofs in **JL** are studied in [AA01,AB07,BF09], however none of these address audit trails. From a type theoretic perspective we should mention the theory of dependent types [Bar92] where types may depend on terms, in much the same way that a type $[\![s]\!]A$ depends on the proof term s. However, dependent type theory lacks a notion of internalisation of derivations as is available in **JL**.

Structure of the paper. Sec. 2 introduces $\mathbf{JL}^{\bullet}$, an affine fragment of **JL**. Sec. 3 studies normalisation in this system. We then introduce a term assignment for this logic in order to obtain a lambda calculus with computation history trails. This calculus is endowed with a call-by-value operational semantics and type safety of this semantics w.r.t. the type system is proved. Sec. 5 addresses strong normalisation. Finally, we conclude and suggest further avenues for research.

2 The Logic

JL (formerly, the *Logic of Proofs*) is a modal logic of provability which has a sound and complete arithmetical semantics. This section introduces a natural deduction presentation for a fragment[4] of **JL**. The inference schemes we shall define give meaning to *hypothetical judgements with explicit evidence* $\Delta; \Gamma; \Sigma \vdash A \mid s$ whose intended reading is: "*s is evidence that A is true under validity hypothesis Δ, truth hypothesis Γ and equivalence hypothesis Σ*. Hypothesis in Γ are the standard term variables $a, b, \ldots$, those in Δ are audited unit variables $u, v, \ldots$, and those in Σ are trail variables $\alpha, \beta, \ldots$. These last hypothesis are often referred to as *equivalence hypothesis* since the type of a trail variable is a proposition that states the equivalence of two typing derivations. The syntax of each component of the judgement is as follows:

[4] Intuitionistic propositional **JL** without the "plus" polynomial proof constructor.

$$\frac{a : A \in \Gamma}{\Delta; \Gamma; \Sigma \vdash A \mid a}\,\mathsf{oVar} \qquad \frac{\Delta; \Gamma, a : A; \Sigma \vdash B \mid s}{\Delta; \Gamma; \Sigma \vdash A \supset B \mid \lambda a : A.s} \supset \mathsf{I}$$

$$\frac{\begin{array}{c}\Delta; \Gamma_1; \Sigma_1 \vdash A \supset B \mid s \\ \Delta; \Gamma_2; \Sigma_2 \vdash A \mid t\end{array}}{\Delta; \Gamma_{1,2}; \Sigma_{1,2} \vdash B \mid s \cdot t} \supset \mathsf{E} \qquad \frac{u : A[\Sigma] \in \Delta \quad \Sigma\sigma \subseteq \Sigma'}{\Delta; \Gamma; \Sigma' \vdash A \mid \langle u; \sigma \rangle}\,\mathsf{mVar}$$

$$\frac{\begin{array}{c}\Delta; \cdot; \Sigma \vdash A \mid s \\ \Delta; \cdot; \Sigma \vdash \mathsf{Eq}(A, s, t) \mid e\end{array}}{\Delta; \Gamma; \Sigma' \vdash [\![\Sigma.t]\!]A \mid \Sigma.t}\,\Box\mathsf{I} \qquad \frac{\begin{array}{c}\Delta; \Gamma_1; \Sigma_1 \vdash [\![\Sigma.r]\!]A \mid s \\ \Delta, u : A[\Sigma]; \Gamma_2; \Sigma_2 \vdash C \mid t\end{array}}{\Delta; \Gamma_{1,2}; \Sigma_{1,2} \vdash C^u_{\Sigma.r} \mid \text{LET}\, u : A[\Sigma] = s \,\text{IN}\, t}\,\Box\mathsf{E}$$

$$\frac{\alpha : \mathsf{Eq}(A) \in \Sigma \quad \Delta; \cdot; \cdot \vdash \mathcal{T}^B \mid \theta^w}{\Delta; \Gamma; \Sigma \vdash B \mid \alpha\theta^w}\,\mathsf{Tlk} \qquad \frac{\Delta; \Gamma; \Sigma \vdash A \mid s \quad \Delta; \Gamma; \Sigma \vdash \mathsf{Eq}(A, s, t) \mid e}{\Delta; \Gamma; \Sigma \vdash A \mid t}\,\mathsf{Eq}$$

Fig. 1. Explanation for Hypothetical Judgements with Explicit Evidence

$$\begin{array}{rlrl}
\text{Propositions } A & ::= P \mid A \supset A \mid [\![\Sigma.s]\!]A & \text{Renaming } \sigma & ::= \{\alpha_1/\beta_1, \ldots, \alpha_n/\beta_n\} \\
\text{Validity ctxt } \Delta & ::= \cdot \mid \Delta, u : A[\Sigma] & \text{Evidence } s & ::= a \mid \lambda a : A.s \mid s \cdot s \\
\text{Truth ctxt } \Gamma & ::= \cdot \mid \Gamma, a : A & & \mid\ \langle u; \sigma \rangle \mid \Sigma.s \\
\text{Equiv. ctxt } \Sigma & ::= \cdot \mid \Sigma, \alpha : \mathsf{Eq}(A) & & \mid\ \text{LET}\, u : A[\Sigma] = s \,\text{IN}\, s \mid \alpha\theta^w
\end{array}$$

Both Γ and Σ are affine hypothesis whereas those in Δ are intuitionistic. Contexts are considered multisets; "$\cdot$" denotes the empty context. In $\Delta; \Gamma; \Sigma$ we assume all variables to be fresh. Variables in Σ are assigned a type of the form $\mathsf{Eq}(A)$[5]. A proposition is either a propositional variable P, an implication $A \supset B$ or a modality $[\![\Sigma.s]\!]A$. In $[\![\Sigma.s]\!]A$, "$\Sigma.$" binds all occurrences of trail variables in s and hence may be renamed at will. We refer to an encoding of a type derivation as *evidence*. Evidence bear witness to proofs of propositions, they encode each possible scheme that may be applied: truth hypothesis, abstraction, audited computation unit variable, audited computation unit introduction and elimination, and trail look-up. The expression θ^w ('w' is for 'witness') will be explained shortly. We write σ for trail variable (bijective) renamings. Free truth variables of s ($\mathsf{fvT(s)}$), free validity variables of s ($\mathsf{fvV(s)}$) and free trail variables of s ($\mathsf{fvTrl(s)}$) are defined as expected. We write s^a_t for the substitution of all free occurrences of a in s by t. Substitution of validity variables is denoted $s^u_{\Sigma.t}$. Its definition is standard except perhaps for the case $\langle u; \sigma \rangle^u_{\Sigma.s}$: here all occurrences of Σ in s are renamed via σ.

The meaning of hypothetical judgements with explicit evidence is given in Fig. 1 and determine the $\mathbf{JL}^\bullet$ *system*. The axiom scheme oVar states that judgement "$\Delta; \Gamma; \Sigma \vdash A \mid a$" is evident in itself: if we assume a is evidence that

[5] The type $\mathsf{Eq}(A)$ in an assignment $\alpha : \mathsf{Eq}(A)$ may informally be understood as $\exists x, y.\mathsf{Eq}(A, x, y)$ (where x, y stand for arbitrary type derivations of propositions of type A) since α stands for a proof of equivalence of two type derivations of propositions of type A about which nothing more may be assumed. These type derivations are hidden since trail lookup may take place at any time.

$$\dfrac{\dfrac{\dfrac{\dfrac{\pi_1}{\Delta; a : A \vdash B \mid s}}{\Delta; \cdot \vdash A \supset B \mid \lambda a : A.s}\supset\mathsf{I} \qquad \dfrac{\pi_2}{\Delta; \cdot \vdash A \mid t}}{\Delta; \cdot \vdash B \mid (\lambda a : A.s) \cdot t}\supset\mathsf{E}}{\Delta; \Gamma \vdash [\![(\lambda a : A.s) \cdot t]\!]B \mid !(\lambda a : A.s) \cdot t}\Box\mathsf{I} \qquad \dfrac{\dfrac{\pi_3}{\Delta; \cdot \vdash B \mid s^a_t}}{\Delta; \Gamma \vdash \begin{array}{c}[\![(\lambda a : A.s) \cdot t]\!]B \\ !(\lambda a : A.s) \cdot t\end{array} \mid}\Box\mathsf{I}$$

Fig. 2. Failure of subject reduction for naive modal introduction scheme

proposition A is true, then we may immediately conclude that A is true with evidence a. The introduction scheme for the modality internalises meta-level evidence into the object logic. It arises from addressing the shortcomings of the more naive scheme (in which Σ is ignored) for introducing this modality:

$$\dfrac{\Delta; \cdot \vdash A \mid s}{\Delta; \Gamma \vdash [\![s]\!]A \mid !s}\Box\mathsf{I}$$

The resulting system is not closed under substitution of derivations. Eg. contraction of the derivation in Fig. 2 (left) would produce the invalid (since evidence s^a_t and $(\lambda a : A.s) \cdot t$ do not coincide) derivation of Fig. 2 (right) where π_3 is obtained from $\pi_{1,2}$ and an appropriate substitution principle. Subject Reduction may be regained, however, by introducing a judgement stating equivalence of evidence $\Delta; \Gamma; \Sigma \vdash \mathsf{Eq}(A, s, t) \mid e$, e is dubbed *equivalence witness*, together with the scheme Eq (Fig. 1). A consequence of this is that normalisation gives rise to instances of Eq appearing in any part of the derivation complicating metatheoretic reasoning. However, since this scheme can be permuted past all other schemes *except* for the introduction of the modality (as may easily be verified), this suggests postulating a hypothesis of evidence equivalence in the introduction scheme for the modality and results in the current scheme $\Box\mathsf{I}$ (Fig. 1). Normalisation steps performed in the derivation ending in the leftmost hypothesis are encoded by equivalence witness e. Finally, $\Box\mathsf{E}$ allows the discharging of validity hypothesis: to discharge the validity hypothesis $v : A[\Sigma]$, a proof of the validity of A under derivation equivalence assumptions Σ is required. In our system, this requires proving that $[\![\Sigma.r]\!]A$ is true with some evidence s.

A sample of the schemes defining evidence equivalence are given in Fig. 3. There are four evidence equivalence axioms (EqRefl, $\mathsf{Eq}\beta$, $\mathsf{Eq}\beta_\Box$ and EqTlk; the third is not exhibited) and six inference schemes (the rest). The axioms are used for recording principle contractions (Sec. 3) at the root of a term and schemes EqAbs, EqApp, EqLet and EqRpl (only second exhibited) enable the same recording but under each of the term constructors. In accordance with the abovementioned discussion on permutation of Eq past other schemes, there is no congruence scheme for the modality. Equivalence witness may be one of the following where $\mathit{Rpl}(e_1, \ldots, e_{10})$ is usually abbreviated $\mathit{Rpl}(\overline{e})$:

$$\begin{array}{rl} e ::= & \mathit{Rfl}(s) \mid \mathit{Sym}(e) \mid \mathit{Trn}(e, e) \mid \beta(a^A.s, s) \mid \beta_\Box(u^{A[\Sigma]}.s, \Sigma.s) \\ \mid & \mathit{Trl}(\theta^w, \alpha) \mid \mathit{Abs}(a^A.e) \mid \mathit{App}(e, e) \mid \mathit{Let}(u^{A[\Sigma]}.e, e) \mid \mathit{Rpl}(e_1, \ldots, e_{10}) \end{array}$$

$$\frac{\Delta;\Gamma;\Sigma \vdash A \mid s}{\Delta;\Gamma;\Sigma \vdash \mathsf{Eq}(A,s,s) \mid \mathit{Rfl}(s)}\ \mathsf{EqRefl} \qquad \frac{\begin{array}{c}\Delta;\Gamma;\Sigma \vdash \mathsf{Eq}(A,s_1,s_2) \mid e_1\\ \Delta;\Gamma;\Sigma \vdash \mathsf{Eq}(A,s_2,s_3) \mid e_2\end{array}}{\Delta;\Gamma;\Sigma \vdash \mathsf{Eq}(A,s_1,s_3) \mid \mathit{Trn}(e_1,e_2)}\ \mathsf{EqTrans}$$

$$\frac{\begin{array}{c}\Delta;\Gamma_1,a:A;\Sigma_1 \vdash B \mid s\\ \Delta;\Gamma_2;\Sigma_2 \vdash A \mid t\end{array}}{\Delta;\Gamma_{1,2};\Sigma_{1,2} \vdash \mathsf{Eq}(B,s^a_t,(\lambda a:A.s)\cdot t) \mid \beta(a^A.s,t)}\ \mathsf{Eq}\beta$$

$$\frac{\Delta;\cdot;\Sigma_1 \vdash \mathsf{Eq}(A,s,t) \mid e \quad \Delta;\cdot;\cdot \vdash \mathcal{T}^B \mid \theta^w \quad \alpha:\mathsf{Eq}(A)\in\Sigma_2}{\Delta;\Gamma;\Sigma_2 \vdash \mathsf{Eq}(B,e\theta^w,\alpha\theta^w) \mid \mathit{Trl}(\theta^w,\alpha)}\ \mathsf{EqTlk}$$

$$\frac{\begin{array}{c}\Delta;\Gamma_1;\Sigma_1 \vdash \mathsf{Eq}(A\supset B,s_1,s_2) \mid e_1\\ \Delta;\Gamma_2;\Sigma_2 \vdash \mathsf{Eq}(A,t_1,t_2) \mid e_2\end{array}}{\Delta;\Gamma_{1,2};\Sigma_{1,2} \vdash \mathsf{Eq}(B,s_1\cdot t_1,s_2\cdot t_2) \mid \mathit{App}(e_1,e_2)}\ \mathsf{EqApp}$$

Fig. 3. Sample schemes defining evidence equivalence judgement

Regarding trail look-up (Tlk in Fig. 1) recall from the introduction that we append each reference to a trail variable with a trail replacement. Therefore, the evidence for look-ups has to be accompanied by proofs of propositions corresponding to each term that is to replace equivalence witness constructors. The evidence for each of these proofs is then encoded as θ^w. This requirement is reflected by the hypothesis $\Delta;\cdot;\cdot \vdash \mathcal{T}^B \mid \theta^w$ which is a shorthand for $\Delta;\cdot;\cdot \vdash \mathcal{T}^B(c) \mid \theta^w(c)$, for each c in the set of equivalence witness constructors $\{\mathit{Rfl},\mathit{Sym},\mathit{Trn},\beta,\beta_\Box,\mathit{Tlk},\mathit{Abs},\mathit{App},\mathit{Let},\mathit{Rpl}\}$, where $\mathcal{T}^B(c)$ is the type of term that replaces the trail constructor c. These types are defined as one might expect (for example, $\mathcal{T}^B(\mathit{Trn}) \stackrel{\text{def}}{=} B\supset B\supset B$ and $\mathcal{T}^B(\beta) \stackrel{\text{def}}{=} B$).

Some basic meta-theoretic results about $\mathbf{JL}^\bullet$ are presented next. The judgements in the statement of these results are decorated with terms (eg. M) which may safely be ignored for the time being (they are introduced in Sec. 4).

Lemma 1 (Weakening).

1. *If $\Delta;\Gamma;\Sigma \vdash M:A \mid s$ is derivable, then so is $\Delta';\Gamma';\Sigma' \vdash M:A \mid s$, where $\Delta\subseteq\Delta'$, $\Gamma\subseteq\Gamma'$ and $\Sigma\subseteq\Sigma'$.*
2. *If $\Delta;\Gamma;\Sigma \vdash \mathsf{Eq}(A,s,t) \mid e$ is derivable, then so is $\Delta';\Gamma';\Sigma' \vdash \mathsf{Eq}(A,s,t) \mid e$, where $\Delta\subseteq\Delta'$, $\Gamma\subseteq\Gamma'$ and $\Sigma\subseteq\Sigma'$.*

We abbreviate Γ_1,Γ_2 with $\Gamma_{1,2}$. If $\Gamma=\Gamma_1,a:A,\Gamma_3$, we write $\Gamma^a_{\Gamma_2}$ for $\Gamma_{1,2,3}$.

Lemma 2 (Subst. Principle for Truth Hypothesis). *Suppose $\Delta;\Gamma_2;\Sigma_2 \vdash N:A \mid t$ is derivable and $a:A\in\Gamma_1$.*

1. *If $\Delta;\Gamma_1;\Sigma_1 \vdash M:B \mid s$, then $\Delta;{\Gamma_1}^a_{\Gamma_2};\Sigma_{1,2} \vdash M^a_{N,t}:B \mid s^a_t$.*
2. *If $\Delta;\Gamma_1;\Sigma_1 \vdash \mathsf{Eq}(B,s_1,s_2) \mid e$, then $\Delta;{\Gamma_1}^a_{\Gamma_2};\Sigma_{1,2} \vdash \mathsf{Eq}(B,(s_1)^a_t,(s_2)^a_t) \mid e^a_t$.*

In the substitution principle for validity variables, note that substitution of $u : A[\Sigma_1]$ requires not only a derivation of $\Delta_{1,2}; \cdot; \Sigma_1 \vdash M : A \mid s$, but also its normalisation history $\Delta_{1,2}; \cdot; \Sigma_1 \vdash \mathsf{Eq}(A, s, t) \mid e_1$ (cf. substitution of validity variables, in particular the clause for $\langle u; \sigma \rangle$, in Sec. 4).

Lemma 3 (Subst. Principle for Validity Hypothesis). *Suppose judgements $\Delta_{1,2}; \cdot; \Sigma_1 \vdash M : A \mid s$ and $\Delta_{1,2}; \cdot; \Sigma_1 \vdash \mathsf{Eq}(A, s, t) \mid e_1$ are derivable. Let $\Delta \stackrel{\text{def}}{=} \Delta_1, u : A[\Sigma_1], \Delta_2$. Then:*

1. *If $\Delta; \Gamma; \Sigma_2 \vdash N : C \mid r$, then $\Delta_{1,2}; \Gamma; \Sigma_2 \vdash N^u_{\Sigma_1.(M,t,e_1)} : C^u_{\Sigma_1.t} \mid r^u_{\Sigma_1.t}$.*
2. *If $\Delta; \Gamma; \Sigma_2 \vdash \mathsf{Eq}(C, s_1, s_2) \mid e_2$, then $\Delta_{1,2}; \Gamma; \Sigma_2 \vdash \mathsf{Eq}(C^u_{\Sigma_1.t}, {s_1}^u_{\Sigma_1.t}, {s_2}^u_{\Sigma_1.t}) \mid {e_2}^u_{\Sigma_1.t}$.*

The last ingredient we require before discussing normalisation is the following lemma which is used for computing the results of trail look-up. The expression $e\theta$ produces a term by replacing each equivalence witness constructor c in e by its correesponding term $\theta(c)$. For example, $\beta(a^A.r, t)\theta \stackrel{\text{def}}{=} \theta(\beta)$ and $\mathit{Trn}(e_1, e_2)\theta \stackrel{\text{def}}{=} \theta(\mathit{Trn})\, e_1\theta\, e_2\theta$. In contrast, $e\theta^w$ produces evidence by replacing each equivalence witness constructor c in e with $\theta^w(c)$.

Lemma 4. *$\Delta; \cdot; \cdot \vdash \mathcal{T}^B \mid \theta^w$ and $\Delta; \cdot; \Sigma_2 \vdash \mathsf{Eq}(A, s, t) \mid e$ implies $\Delta; \cdot; \cdot \vdash e\theta : B \mid e\theta^w$.*

3 Normalisation

Normalisation equates derivations and since $\mathbf{JL}^\bullet$ internalises its own proofs, normalisation steps must explicitly relate evidence in order for SR to hold. Normalisation is modeled as a two step process. First a *principle contraction* is applied, then a series of *permutation conversions* follow. Principle contractions introduce explicit witnesses of derivation equivalence. Permutation conversions standardize derivations by moving these witnesses to the innermost $\Box$ introduction scheme. There are three principal contractions (β, $\beta_\Box$ and Tlk-contraction), the first two of which rely on the substitution principles discussed earlier. The first replaces a derivation of the form:

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Delta; \Gamma_1, a : A; \Sigma_1 \vdash B \mid s}}{\Delta; \Gamma_1; \Sigma_1 \vdash A \supset B \mid \lambda a : A.s}\supset\mathsf{I} \qquad \dfrac{\pi_2}{\Delta; \Gamma_2; \Sigma_2 \vdash A \mid t}}{\Delta; \Gamma_{1,2}; \Sigma_{1,2} \vdash B \mid (\lambda a : A.s) \cdot t}\supset\mathsf{E}$$

by the following, where π_3 is a derivation of $\Delta; \Gamma_{1,2}; \Sigma_{1,2} \vdash B \mid s^a_t$ resulting from π_1 and π_2 and the Substitution Principle for Truth Hypothesis:

$$\dfrac{\pi_3 \qquad \dfrac{\dfrac{\pi_1}{\Delta; \Gamma_1, a : A; \Sigma_1 \vdash B \mid s} \qquad \dfrac{\pi_2}{\Delta; \Gamma_2; \Sigma_2 \vdash A \mid t}}{\Delta; \Gamma_{1,2}; \Sigma_{1,2} \vdash \mathsf{Eq}(B, s^a_t, (\lambda a : A.s) \cdot t) \mid \beta(a^A.s, t)}}{\Delta; \Gamma_{1,2}; \Sigma_{1,2} \vdash B \mid (\lambda a : A.s) \cdot t}\mathsf{Eq}$$

The second contraction replaces:

$$\dfrac{\dfrac{\begin{array}{c}\Delta;\cdot;\Sigma\vdash A\mid s\\ \Delta;\cdot;\Sigma\vdash \mathsf{Eq}(A,s,t)\mid e_1\end{array}}{\Delta;\Gamma_1;\Sigma_1\vdash [\![\Sigma.t]\!]A\mid\Sigma.t}\,\Box\mathsf{I}\qquad \Delta,u:A[\Sigma];\Gamma_2;\Sigma_2\vdash C\mid r}{\Delta;\Gamma_{1,2};\Sigma_{1,2}\vdash C^u_{\Sigma.t}\mid \textsc{let}\,u:A[\Sigma]=\Sigma.t\,\textsc{in}\,r}\,\Box\mathsf{E}$$

with the following derivation where π is a derivation of $\Delta;\Gamma_{1,2};\Sigma_{1,2}\vdash C^u_{\Sigma.t}\mid t^u_{\Sigma.t}$ resulting from the Substitution Principle for Validity Hypothesis followed by weakening (of Γ_1 and Σ_1) and e_2 is $\beta_\Box(u^{A[\Sigma_1]}.r,\Sigma.t)$:

$$\dfrac{\pi\qquad \dfrac{\begin{array}{c}\Delta;\cdot;\Sigma\vdash A\mid s\\ \Delta;\cdot;\Sigma\vdash \mathsf{Eq}(A,s,t)\mid e_1\\ \Delta,u:A[\Sigma];\Gamma_2;\Sigma_2\vdash C\mid r\end{array}}{\Delta;\Gamma_{1,2};\Sigma_{1,2}\vdash \mathsf{Eq}(C^u_{\Sigma.t},r^u_{\Sigma.t},\textsc{let}\,u:A[\Sigma]=\Sigma.t\,\textsc{in}\,r)\mid e_2}\,\mathsf{Eq}\beta_\Box}{\Delta;\Gamma_{1,2};\Sigma_{1,2}\vdash C^u_{\Sigma.t}\mid \textsc{let}\,u:A[\Sigma]=\Sigma.t\,\textsc{in}\,r}\,\mathsf{Eq}$$

Tlk-contraction models audit trail look-up. Consider the following derivation, where $\Sigma_1\subseteq\Sigma_2$, $\Delta'\subseteq\Delta$ and the branch from the depicted instance of Tlk in π_1 to its conclusion has no instances of $\Box\mathsf{I}$:

$$\dfrac{\begin{array}{c}\dfrac{\begin{array}{c}\alpha:\mathsf{Eq}(A)\in\Sigma_1\\ \Delta;\cdot;\cdot\vdash\mathcal{T}^B\mid\theta^w\end{array}}{\Delta;\Gamma;\Sigma_1\vdash B\mid\alpha\theta^w}\,\mathsf{Tlk}\\ \vdots\,\pi_1\\ \Delta';\cdot;\Sigma_2\vdash A\mid s\end{array}\qquad \dfrac{\pi_2}{\Delta';\cdot;\Sigma_2\vdash\mathsf{Eq}(A,s,t)\mid e}}{\Delta';\Gamma';\Sigma'\vdash[\![\Sigma_2.t]\!]A\mid\Sigma_2.t}\,\Box\mathsf{I}$$

The instance of Tlk in π_1 is replaced by the following derivation where π_2' is obtained from π_2 by resorting to Lem. 4 and Lem. 1. Also, $\Delta;\cdot;\Sigma_2\vdash\mathsf{Eq}(A,s,t)\mid e$ is obtained from $\Delta';\cdot;\Sigma_2\vdash\mathsf{Eq}(A,s,t)\mid e$ by Lem. 1.

$$\dfrac{\dfrac{\pi_2'}{\Delta;\Gamma;\Sigma_1\vdash B\mid e\theta^w}\qquad\dfrac{\begin{array}{c}\Delta;\cdot;\Sigma_2\vdash\mathsf{Eq}(A,s,t)\mid e\\ \Delta;\cdot;\cdot\vdash\mathcal{T}^B\mid\theta^w\end{array}}{\Delta;\Gamma;\Sigma_1\vdash\mathsf{Eq}(B,e\theta^w,\alpha\theta^w)\mid \mathit{Trl}(\theta^w,\alpha)}\,\mathsf{EqTlk}}{\Delta;\Gamma;\Sigma_1\vdash B\mid\alpha\theta^w}\,\mathsf{Eq}$$

As for the permutation conversions, they indicate how Eq is permuted past any of the inference schemes in $\{\supset\mathsf{I},\supset\mathsf{E},\Box\mathsf{E},\mathsf{Eq},\mathsf{Tlk}\}$. Also, there is a conversion that fuses Eq just above the left hypothesis in an instance of $\Box\mathsf{I}$ with the trail of the corresponding unit is also coined. As an example Eq permutes past $\supset\mathsf{I}$ by replacing:

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Delta;\Gamma,a:A;\Sigma\vdash B\mid s}\qquad\dfrac{\pi_2}{\Delta;\Gamma,a:A;\Sigma\vdash\mathsf{Eq}(B,s,t)\mid e}}{\Delta;\Gamma,a:A;\Sigma\vdash B\mid t}\,\mathsf{Eq}}{\Delta;\Gamma;\Sigma\vdash A\supset B\mid\lambda a:A.t}\supset\mathsf{I}$$

with the following derivation where π_3 is a derivation of $\Delta; \Gamma; \Sigma \vdash A \supset B \mid \lambda a : A.s$ obtained from π_1 and $\supset \mathsf{I}$:

$$\dfrac{\pi_3 \qquad \dfrac{\Delta; \Gamma, a : A; \Sigma \vdash \mathsf{Eq}(B, s, t) \mid e}{\Delta; \Gamma; \Sigma \vdash \mathsf{Eq}(A \supset B, \lambda a : A.s, \lambda a : A.t) \mid Abs(a^A.e)}\ \mathsf{EqAbs}}{\Delta; \Gamma; \Sigma \vdash A \supset B \mid \lambda a : A.t}\ \mathsf{Eq}$$

4 Term Assignment

Computation by normalisation is non-confluent, as one might expect (audit trail look-up affects computation), hence a strategy is required. This section introduces the call-by-value $\lambda^{\natural}$-calculus. It is obtained via a term assignment for $\mathbf{JL}^{\bullet}$. The syntax of $\lambda^{\natural}$ terms is:

$$M ::= a \mid \lambda a : A.M \mid M\, M \mid \langle u; \sigma \rangle \mid !^{\Sigma}_{e} M \mid \mathsf{let}\, u : A[\Sigma] = M \,\mathsf{in}\, M \mid \alpha\theta \mid e \rhd M$$

In addition to term variables, abstraction and application we also have audited computation unit variables, audited computation units, audited computation unit substitution, trail look-up and terms decorated with equivalence witnesses. We occasionally drop the type decoration in let construct for readability. Since terms may be decorated with equivalence witnesses, substitution (both for truth and validity hypothesis) substitutes free occurrences of variables with both terms and evidence. We write $M^{a}_{N,t}$ for substitution of truth variables and $M^{u}_{\Sigma.(N,t,e)}$ for substitution of validity variables (similar notions apply to substitution in propositions, evidence and equivalence witnesses). Note that "Σ." in $\Sigma.(N, t, e)$ binds all free occurrences of trail variables from Σ which occur in N, t and e. For illustration we give the definition of $M^{u}_{\Sigma.(N,t,e)}$, where $s^{u}_{\Sigma.t}$ traverses the structure of s replacing $\langle u; \sigma \rangle^{u}_{\Sigma.s}$ with $s\sigma$ and $e^{u}_{\Sigma.t}$ traverses the structure of e until it reaches one of $Rfl(r_1), \beta(a^A.r_1, r_2)$ or $\beta_{\Box}(v^{A[\Sigma']}.r_1, \Sigma'.r_2)$ in which case it resorts to substitution over the r_is. Note how the fourth clause of the definition of $M^{u}_{\Sigma.(N,t,e)}$ below substitutes $\langle u; \sigma \rangle$ with $e\sigma \rhd N\sigma$, thus propagating the history.

$$\begin{array}{rclrcl}
b^{u}_{\Sigma.(N,t,e)} & \overset{\text{def}}{=} & b & (!^{\Sigma'}_{e'} M)^{u}_{\Sigma.(N,t,e)} & \overset{\text{def}}{=} & !^{\Sigma'}_{e'^{u}_{\Sigma.t}} M^{u}_{\Sigma.(N,t,e)} \\
(\lambda b : A.M)^{u}_{\Sigma.(N,t,e)} & \overset{\text{def}}{=} & \lambda b : A.M^{u}_{\Sigma.(N,t,e)} & (\text{LET}\, v = P & \overset{\text{def}}{=} & \text{LET}\, v = P^{u}_{\Sigma.(N,t,e)} \\
(P\, Q)^{u}_{\Sigma.(N,t,e)} & \overset{\text{def}}{=} & P^{u}_{\Sigma.(N,t,e)}\, Q^{u}_{\Sigma.(N,t,e)} & \text{IN}\, Q)^{u}_{\Sigma.(N,t,e)} & & \text{IN}\, Q^{u}_{\Sigma.(N,t,e)} \\
\langle u; \sigma \rangle^{u}_{\Sigma.(N,t,e)} & \overset{\text{def}}{=} & e\sigma \rhd N\sigma & (\alpha\theta)^{u}_{\Sigma.(N,t,e)} & \overset{\text{def}}{=} & \alpha(\theta^{u}_{\Sigma.(N,t,e)}) \\
\langle v; \sigma \rangle^{u}_{\Sigma.(N,t,e)} & \overset{\text{def}}{=} & \langle v; \sigma \rangle & (e' \rhd M)^{u}_{\Sigma.(N,t,e)} & \overset{\text{def}}{=} & e'^{u}_{\Sigma.t} \rhd M^{u}_{\Sigma.(N,t,e)}
\end{array}$$

The typing judgement $\Delta; \Gamma; \Sigma \vdash M : A \mid s$ is defined by means of the *typing schemes* obtained from decorating the inference schemes of Fig. 1 with terms. Sample schemes are given in Fig. 4. A term M is said to be *typable* if there exists $\Delta, \Gamma, \Sigma, A, s$ s.t. $\Delta; \Gamma; \Sigma \vdash M : A \mid s$ is derivable. The operational semantics of $\lambda^{\natural}$ is specified by a binary relation over typed terms called *reduction* ($M \to N$). In order to define reduction we first introduce two intermediate notions, namely *principle reduction* ($M \mapsto N$) and *permutation reduction* ($M \curvearrowright N$) . The former corresponds to principle contraction and the latter to permutation conversions of the normalisation procedure. The set of *evaluation contexts* and *values* are:

$$\frac{\Delta;\cdot;\Sigma\vdash M:A\mid s \quad \Delta;\cdot;\Sigma\vdash \mathsf{Eq}(A,s,t)\mid e}{\Delta;\Gamma;\Sigma'\vdash !^{\Sigma}_{e}M:[\![\Sigma.t]\!]A\mid \Sigma.t}\ \mathsf{TBox} \qquad \frac{\alpha:\mathsf{Eq}(A)\in\Sigma \quad \Delta;\cdot;\cdot\vdash\theta:\mathcal{T}^{B}\mid\theta^{w}}{\Delta;\Gamma;\Sigma\vdash\alpha\theta:B\mid\alpha\theta^{w}}\ \mathsf{TTlk}$$

$$\frac{\Delta;\Gamma_1;\Sigma_1\vdash M:[\![\Sigma.r]\!]A\mid s \quad \Delta,u:A[\Sigma];\Gamma_2;\Sigma_2\vdash N:C\mid t}{\Delta;\Gamma_{1,2};\Sigma_{1,2}\vdash \begin{array}{c}\mathsf{let}\,u:A[\Sigma]=M\,\mathsf{in}\,N:C^{u}_{\Sigma.r}\\ \text{LET}\,u:A[\Sigma]=s\,\text{IN}\,t\end{array}\mid}\ \mathsf{TLetB} \qquad \frac{\Delta;\Gamma;\Sigma\vdash M:A\mid s \quad \Delta;\Gamma;\Sigma\vdash\mathsf{Eq}(A,s,t)\mid e}{\Delta;\Gamma;\Sigma\vdash e\rhd M:A\mid t}\ \mathsf{TEq}$$

Fig. 4. Sample typing schemes for $\lambda^{\natural}$

$$\begin{array}{llll}
\mathcal{E} ::= \Box\mid\mathcal{E}\ M\mid V\ \mathcal{E}\mid\mathsf{let}\,u:A[\Sigma]=\mathcal{E}\,\mathsf{in}\,M & & V ::= a\mid\langle u;\sigma\rangle\mid\lambda a:A.M \\
\quad\mid\ !^{\Sigma}_{e}\mathcal{E}\mid\alpha\{c_1/V_1,\ldots,c_j/V_j,c_{j+1}/\mathcal{E},\ldots\} & & \quad\mid\ !^{\Sigma}_{e}V \\
\mathcal{F} ::= \Box\mid\mathcal{F}\ M\mid V\ \mathcal{F}\mid\mathsf{let}\,u:A[\Sigma]=\mathcal{F}\,\mathsf{in}\,M & & \theta_V ::= \{c_1/V_1,\ldots,c_{10}/V_{10}\}
\end{array}$$

Evaluation contexts are represented with letters $\mathcal{E},\mathcal{E}'$, etc. Note that reduction under the audited unit constructor is allowed. Contexts $\mathcal{F}$ are required for defining $\mathcal{L}$, the principle reduction axiom for trail look-up (defined below). It differs from $\mathcal{E}$ by not allowing holes under the audited unit constructor. The set of values are standard except for $!^{\Sigma}_{e}V$: audited units with fully evaluated bodies are also values. θ_V is a trail replacement consisting entirely of values. Principle reduction is presented by means of the following principle reduction *axiom* and *congruence schemes*:

$$\begin{array}{rcl}
(\lambda a:A.M)\ V & \rightharpoonup_{\beta} & \beta(a^{A}.s,t)\rhd M^{a}_{V,t} \\
\mathsf{let}\,u:A[\Sigma]=!^{\Sigma}_{e}V\,\mathsf{in}\,N & \rightharpoonup_{\beta_\Box} & \beta_\Box(u^{A[\Sigma]}.t,\Sigma.s)\rhd N^{u}_{\Sigma.(V,s,e)} \\
!^{\Sigma}_{e}\mathcal{F}[\alpha\theta_V] & \rightharpoonup_{\mathcal{L}} & !^{\Sigma}_{e}\mathcal{F}[\mathit{Trl}(\theta^{w}_{V},\alpha)\rhd e\theta_V]
\end{array}$$

$$M\rightharpoonup N \text{ implies } \mathcal{E}[M]\mapsto\mathcal{E}[N]7$$

These schemes have been abridged by removing typing information. For example, the fully decorated presentation of β is:

$$\frac{\Delta;\Gamma_1,a:A;\Sigma_1\vdash M:B\mid s \quad \Delta;\Gamma_2;\Sigma_2\vdash V:A\mid t}{\Delta;\Gamma_{1,2};\Sigma_{1,2}\vdash(\lambda a:A.M)\ V\rightharpoonup_{\beta}\beta(a^{A}.s,t)\rhd M^{a}_{V,t}:B\mid(\lambda a:A.s)\cdot t}$$

The fully decorated presentation of $\beta_\Box$ is as follows where $O\overset{\text{def}}{=}\mathsf{let}\,u:A[\Sigma]=\,!^{\Sigma}_{e}V\,\mathsf{in}\,N$ and $P\overset{\text{def}}{=}\beta_\Box(u^{A[\Sigma]}.t,\Sigma.s)\rhd N^{u}_{\Sigma.(V,s,e)}$:

$$\frac{\Delta;\cdot;\Sigma\vdash V:A\mid r \quad \Delta;\cdot;\Sigma\vdash\mathsf{Eq}(A,r,s)\mid e_1 \quad \Delta,u:A[\Sigma];\Gamma_2;\Sigma_2\vdash N:C\mid t}{\Delta;\Gamma_{1,2};\Sigma_{1,2}\vdash O\rightharpoonup_{\beta_\Box}P:C^{u}_{\Sigma.s}\mid\text{LET}\,u:A[\Sigma]=\Sigma.s\,\text{IN}\,t}$$

Each principle reduction scheme produces a trail of its execution. Note that $\beta_\Box$ replaces all occurrences of $\langle u;\sigma\rangle$ with $e\sigma\rhd V\sigma$, correctly: (1) preserving trails and (2) rewiring trail variables so that they now refer to their host audited computation unit. Regarding permutation reduction, the original schemes obtained from the normalisation procedure are the contextual closure of the first group

$$\begin{array}{rcl}
(e \rhd M)\ N & \curvearrowright & \mathit{App}(e, \mathit{Rfl}(t)) \rhd (M\ N) \\
M\ (e \rhd N) & \curvearrowright & \mathit{App}(\mathit{Rfl}(t), e) \rhd (M\ N) \\
\lambda a : A.(e \rhd M) & \curvearrowright & \mathit{Abs}(a.e) \rhd (\lambda a : A.M) \\
\mathsf{let}\, u = (e \rhd M)\, \mathsf{in}\ N & \curvearrowright & \mathit{Let}(u.e, \mathit{Rfl}(t)) \rhd (\mathsf{let}\, u = M\, \mathsf{in}\ N) \\
\mathsf{let}\, u = M\, \mathsf{in}\ (e \rhd N) & \curvearrowright & \mathit{Let}(u.\mathit{Rfl}(s), e) \rhd (\mathsf{let}\, u = M\, \mathsf{in}\ N) \\
!^{\Sigma}_{e_2}(e_1 \rhd M) & \curvearrowright & !^{\Sigma}_{\mathit{Trn}(e_1,e_2)} M \\
e_1 \rhd (e_2 \rhd M) & \curvearrowright & \mathit{Trn}(e_1, e_2) \rhd M
\end{array}$$

$$\begin{array}{rcl}
\mathit{Trn}(\mathit{App}(e_1, e_2), \mathit{App}(e_3, e_4)) & \curvearrowright & \mathit{App}(\mathit{Trn}(e_1, e_3), \mathit{Trn}(e_2, e_4)) \\
\mathit{Trn}(\mathit{Abs}(a.e_1), \mathit{Abs}(a.e_2)) & \curvearrowright & \mathit{Abs}(a.\mathit{Trn}(e_1, e_2)) \\
\mathit{Trn}(\mathit{Let}(u.e_1, e_2), \mathit{Let}(u.e_3, e_4)) & \curvearrowright & \mathit{Let}(u.\mathit{Trn}(e_1, e_3), \mathit{Trn}(e_2, e_4)) \\
\mathit{Trn}(\mathit{Rfl}(s), e) & \curvearrowright & e \\
\mathit{Trn}(e, \mathit{Rfl}(t)) & \curvearrowright & e \\
\mathit{Trn}(\mathit{Trn}(e_1, e_2), e_3) & \curvearrowright & \mathit{Trn}(e_1, \mathit{Trn}(e_2, e_3)) \\
\mathit{Trn}(\mathit{App}(e_1, e_2), \mathit{Trn}(\mathit{App}(e_3, e_4), e_5)) & \curvearrowright & \mathit{Trn}(\mathit{App}(\mathit{Trn}(e_1, e_3), \mathit{Trn}(e_2, e_4)), e_5) \\
\mathit{Trn}(\mathit{Abs}(a.e_1), \mathit{Trn}(\mathit{Abs}(a.e_2), e_3)) & \curvearrowright & \mathit{Trn}(\mathit{Abs}(a.\mathit{Trn}(e_1, e_2)), e_3) \\
\mathit{Trn}(\mathit{Let}(u.e_1, e_2), \mathit{Trn}(\mathit{Let}(u.e_3, e_4), e_5)) & \curvearrowright & \mathit{Trn}(\mathit{Let}(u.\mathit{Trn}(e_1, e_3), \mathit{Trn}(e_2, e_4)), e_5)
\end{array}$$

Fig. 5. Permutation reduction schemes

of rules depicted in Fig. 5[6]. As in principle reduction, these schemes operate on typed terms and have been abridged. Eg. the full presentation of the first is:

$$\frac{\Delta; \Gamma_1; \Sigma_1 \vdash M : A \supset B \mid r \quad \Delta; \Gamma_1; \Sigma_1 \vdash \mathsf{Eq}(A \supset B, r, s) \mid e \quad \Delta; \Gamma_2; \Sigma_2 \vdash N : A \mid t}{\Delta; \Gamma_{1,2}; \Sigma_{1,2} \vdash (e \rhd M)\, N \curvearrowright \mathit{App}(e, \mathit{Rfl}(t)) \rhd (M\, N) : B \mid s \cdot t}$$

These schemes are easily proven to be terminating. However, they are not confluent (take, for instance, the critical pair between the first two reduction schemes and note that it is not joinable). As a consequence we complete these schemes with those in the second group depicted in Fig. 5. The full set of schemes is both confluent and terminating.

Proposition 1. $\curvearrowright$ *is confluent and terminating.*

Termination may be proved automatically by using AProVE [GTSKF04]. Confluence follows by checking local confluence and resorting to Newman's Lemma. We stress that the fact that these reduction schemes are defined over typed terms is crucial for confluence. For example, $\mathit{Trn}(\mathit{Rfl}(s), \mathit{Rfl}(t))$ is typable only in the case that $s = t$.

Definition 1 (Reduction). *Let $\Longrightarrow$ stand for permutation reduction to (the unique) normal form. Reduction ($\to$) is defined over terms in permutation-reduction normal form as $\mapsto \circ \Longrightarrow$.*

We now address safety of reduction w.r.t. the type system. This involves proving SR and Progress. SR follows from the fact that the reduction schemes originate from proof normalisation. The exception are the second group of schemes of Fig. 5 for which type preservation may also be proved seperately.

[6] Type decorations in equivalence witnesses omitted for the sake of readability.

$$
\begin{array}{rlcrl}
\mathcal{D} ::= & \Box \mid \lambda a : A.\mathcal{D} \mid \mathcal{D}\ M \mid M\ \mathcal{D} & \quad & \mathcal{C} ::= & \Box \mid \lambda a : A.\mathcal{C} \mid \mathcal{C}\ M \mid M\ \mathcal{C} \\
\mid & \mathsf{let}\, u : A[\Sigma] = \mathcal{D}\,\mathsf{in}\ M & & \mid & \mathsf{let}\, u : A[\Sigma] = \mathcal{C}\,\mathsf{in}\ M \\
\mid & \mathsf{let}\, u : A[\Sigma] = M\,\mathsf{in}\ \mathcal{D} \mid !^{\Sigma}_{e}\mathcal{D} \mid e \rhd \mathcal{D} & & \mid & \mathsf{let}\, u : A[\Sigma] = M\,\mathsf{in}\ \mathcal{C} \\
\mid & \alpha\{c_1/M_1, \ldots, c_j/M_j, c_{j+1}/\mathcal{D}, \ldots\} & & \mid & e \rhd \mathcal{C}
\end{array}
$$

$$
\begin{array}{rcl}
(\lambda a : A.M)\ N & \overset{f}{\rightharpoonup}_{\beta} & \beta(a^{A}.s, t) \rhd M^{a}_{N,t} \\
\mathsf{let}\, u : A[\Sigma] = !^{\Sigma}_{e} M\,\mathsf{in}\ N & \overset{f}{\rightharpoonup}_{\beta\Box} & \beta_{\Box}(u^{A[\Sigma]}.t, \Sigma.s) \rhd N^{u}_{\Sigma.(M,s,e)} \\
!^{\Sigma}_{e}\mathcal{C}[\alpha\theta] & \overset{f}{\rightharpoonup}_{\mathcal{L}} & !^{\Sigma}_{e}\mathcal{C}[\mathit{Trl}(\theta^{w}, \alpha) \rhd e\theta] \\
\multicolumn{3}{c}{M \overset{f}{\rightharpoonup} N \text{ implies } \mathcal{D}[M] \overset{f}{\mapsto} \mathcal{D}[N]}
\end{array}
$$

Fig. 6. Full principle reduction

Proposition 2 (Subject Reduction). *$\Delta; \Gamma; \Sigma \vdash M : A \mid s$ and $M \to N$ implies $\Delta; \Gamma; \Sigma \vdash M : A \mid s$.*

Before addressing Progress we introduce some auxiliary notions. A term is *look-up-blocked* if it is of the form $\mathcal{F}[\alpha\theta_V]$. A term M is *tv-closed* if $\mathsf{fvT}(M) = \mathsf{fvV}(M) = \emptyset$. It is *closed* if it is tv-closed and $\mathsf{fvTrl}(M) = \emptyset$.

Lemma 5 (Canonical forms). *Assume $\cdot; \cdot; \Sigma \vdash V : A \mid s$. Then (1) if $A = A_1 \supset A_2$, then $V = \lambda a : A_1.M$ for some a, M; and (2) if $A = [\![\Sigma'.t]\!]A_1$, then $V = !^{\Sigma'}_{e} V'$ for some e, V'.*

Proposition 3. *Suppose M is in permutation reduction-normal form, is typable and tv-closed. Then (1) M is a value or; (2) there exists N s.t. $M \mapsto N$ or; (3) M is look-up-blocked.*

Since a closed term cannot be look-up-blocked:

Corollary 1 (Progress). *Suppose M is in permutation reduction normal form, is typable and closed. Then either M is a value or there exists N s.t. $M \to N$.*

5 Strong Normalisation

Full reduction is defined as the union of full principle reduction ($\overset{f}{\mapsto}$, Fig. 6) and permutation reduction ($\curvearrowright$). We address strong normalisation (SN) of a restriction of full reduction, a result which entails SN of a similar restriction of $\lambda^{\natural}$. The restriction consists in requiring that M in the principle reduction axiom $\beta_{\Box}$ not have occurrences of the audited computation unit constructor "!". In the sequel, we write $\overset{rf}{\mapsto}$ for this restricted notion of reduction.

We first note that $\overset{f}{\mapsto}_{\beta,\beta_{\Box}}$ is SN. This can be proved by defining a translation $\mathcal{S}(\bullet)$ on $\lambda^{\natural}$ types that "forget" the modal connective and a similar translation from terms in $\lambda^{\natural}$ to terms of the simply typed lambda calculus (with constants) such that: (1) it preserves typability; and (2) it maps full reduction to reduction in the simply typed lambda calculus. Since we already know that $\curvearrowright$ is SN and that reduction in the simply typed lambda calculus is SN, our first result reads:

Proposition 4. *$\overset{f}{\mapsto}_{\beta,\beta_\Box} \cup \curvearrowright$ is SN.*

Therefore, an infinite $\overset{f}{\mapsto} \cup \curvearrowright$ reduction sequence must include an infinite number of $\overset{f}{\mapsto}_{\mathcal{L}}$ steps. Next we show that for $\overset{rf}{\mapsto}$ this is not possible. More precisely, we show that in an infinite $\overset{rf}{\mapsto} \cup \curvearrowright$ reduction sequence, there can only be a finite number of $\overset{f}{\mapsto}_{\mathcal{L}}$ steps. This entails:

Proposition 5. *$\overset{rf}{\mapsto} \cup \curvearrowright$ is SN. Hence $\lambda^{\natural}$, with the same restriction, is SN.*

We now address the proof of the main lemma on which Prop. 5 relies (Lem. 7). We introduce weight functions which strictly decrease by each application of a $\overset{f}{\mapsto}_{\mathcal{L}}$-step and which decreases with each application of a $\overset{rf}{\mapsto}_{\beta,\beta_\Box}$-step or $\curvearrowright$-step. A word on notation: $\langle\!\langle\ \rangle\!\rangle$ is the empty multiset; $\uplus$ is multiset union; and $n \uplus \mathcal{M}$ is the union of the multiset $\langle\!\langle n \rangle\!\rangle$ and $\mathcal{M}$, for $n \in \mathbb{N}$. We use the standard multiset extension $\prec$ of the well-founded ordering $<$ on natural numbers which is also well-founded. For each $n \in \mathbb{N}$ we define $\mathcal{W}_n(M)$ as the multiset given by the following inductive definition on M:

$$\begin{aligned}
\mathcal{W}_n(a) &\overset{\text{def}}{=} \langle\!\langle\ \rangle\!\rangle \\
\mathcal{W}_n(\lambda a : A.M) &\overset{\text{def}}{=} \mathcal{W}_n(M) \\
\mathcal{W}_n(M\,N) &\overset{\text{def}}{=} \mathcal{W}_n(M) \uplus \mathcal{W}_n(N) \\
\mathcal{W}_n(\langle u;\sigma\rangle) &\overset{\text{def}}{=} \langle\!\langle\ \rangle\!\rangle
\end{aligned}
\qquad
\begin{aligned}
\mathcal{W}_n(!_e^{\Sigma} M) &\overset{\text{def}}{=} n * \mathcal{W}^t(M) \uplus \\
&\qquad \uplus \mathcal{W}_{n*\mathcal{W}^t(M)}(M) \\
\mathcal{W}_n(\mathsf{let}\, u = M \,\mathsf{in}\, N) &\overset{\text{def}}{=} \mathcal{W}_n(M) \uplus \mathcal{W}_n(N) \\
\mathcal{W}_n(\alpha\theta) &\overset{\text{def}}{=} \biguplus_{i\in 1..10} \mathcal{W}_n(\theta(c_i)) \\
\mathcal{W}_n(e \rhd M) &\overset{\text{def}}{=} \mathcal{W}_n(M)
\end{aligned}$$

where $\mathcal{W}^t(M)$ is the number of free trail variables in M plus 1. Note that $\mathcal{W}^t(e \rhd M) \overset{\text{def}}{=} \mathcal{W}^t(M)$. The weight functions informally count the number of trail variables that are available for look-up in audited computation units. The principle reduction axiom β either erases the argument N or substitutes exactly one copy, given the affine nature of truth hypothesis. However, multiple copies of M can arise from $\beta_\Box$ reduction (cf. Fig. 6), possibly under "!" constructors (hence our restriction in item 2 below). Finally, we must take into account that although an trail variable is consumed by $\mathcal{L}$ it also copies the terms in θ (which may contain occurrences of the "!" constructor). In contrast to $\beta_\Box$ however, the consumed trail variable can be used to make the copies of "!" made by $e\theta$ weigh less than the outermost occurrence of "!" on the left-hand side of $\mathcal{L}$.

Lemma 6. *1. $\mathcal{W}_n((\lambda a : A.M)\,N) \succeq \mathcal{W}_n(M^a_{N,t})$.*
2. *If M has no occurrences of the modal term constructor, then $\mathcal{W}_n(\mathsf{let}\, u : A[\Sigma] = !_e^{\Sigma} M \,\mathsf{in}\, N) \succ \mathcal{W}_n(\beta_\Box(u^{A[\Sigma]}.t, \Sigma.s) \rhd N^u_{\Sigma.(M,s,e)})$.*
3. *$\mathcal{W}_n(!_e^{\Sigma}\mathcal{C}[\alpha\theta]) \succ \mathcal{W}_n(!_e^{\Sigma}\mathcal{C}[\mathit{Trl}(\theta^w, \alpha) \rhd e\theta])$.*

From these results follow:

Lemma 7. *(1) $M \overset{rf}{\mapsto}_{\beta,\beta_\Box} N$ implies $\mathcal{W}_n(M) \succeq \mathcal{W}_n(N)$; (2) $M \overset{f}{\mapsto}_{\mathcal{L}} N$ implies $\mathcal{W}_n(M) \succ \mathcal{W}_n(N)$; and (3) $M \curvearrowright N$ implies $\mathcal{W}_n(M) = \mathcal{W}_n(N)$.*

6 Example of History Based Access Control

We can also model other phenomena such as Abadi and Fournet's [AF03] mechanism for access control based on execution history. A top-level function declaration is an expression of the form $\mathbf{f} \doteq M$ where $\Delta; \Gamma; \Sigma \vdash M : A \mid s$ together with a typing scheme (left) and evidence equivalence scheme (right):

$$\frac{}{\Delta; \Gamma; \Sigma \vdash \mathbf{f} : A \mid \mathbf{f}}\ \mathsf{TFunc} \qquad \frac{\Delta; \Gamma; \Sigma \vdash A \mid s}{\Delta; \Gamma; \Sigma \vdash \mathsf{Eq}(A, s, \mathbf{f}) \mid \delta_{\mathbf{f}}(s)}\ \mathsf{EqFunc} \tag{1}$$

Also, we have the principle contraction in which the derivation on the left of (1) contracts to:

$$\frac{\Delta; \Gamma; \Sigma \vdash M : A \mid s \qquad \dfrac{\Delta; \Gamma; \Sigma \vdash A \mid s}{\Delta; \Gamma; \Sigma \vdash \mathsf{Eq}(A, s, \mathbf{f}) \mid \delta_{\mathbf{f}}(s)}\ \mathsf{EqFunc}}{\Delta; \Gamma; \Sigma \vdash M : A \mid \mathbf{f}}\ \mathsf{Eq}$$

If $\mathbf{f} \doteq !_e^{\boldsymbol{\alpha}} \lambda \boldsymbol{a} : \boldsymbol{A}.M$, then we abbreviate $\mathsf{let}\, u = \mathbf{f}\, \mathsf{in}\, \langle u; \boldsymbol{\alpha}/\boldsymbol{\beta} \rangle\, \boldsymbol{N}$ with $\mathbf{f}\, \boldsymbol{\beta}\, \boldsymbol{N}$. Consider the following top-level declarations:

$$\begin{array}{ll}
\mathbf{delete} \doteq !^{\alpha_d}_{\mathit{Rfl}(q)} \lambda a.\mathsf{if}\ \mathit{FileIOPerm} \in \theta\alpha_d & \mathbf{cleanup} \doteq !^{\alpha_c}_{\mathit{Rfl}(r)} \lambda a.\mathbf{delete}\ \alpha_c\ a; \\
\qquad\qquad\qquad\quad \mathsf{then}\, \mathbf{Win32Delete}\, a & \mathbf{bad} \quad\ \doteq !^{\alpha_b}_{\mathit{Rfl}(s)} \mathbf{cleanup}\ \alpha_b \\
\qquad\qquad\qquad\quad \mathsf{else}\, \mathbf{securityException}; & \qquad\qquad\quad \text{“}..\backslash \mathit{passwd}'';
\end{array}$$

where the definition of θ requires we first define *perms*, a function assigning a set of (static) permissions to top-level functions: $\mathit{perms}(\mathbf{bad}) \stackrel{\mathrm{def}}{=} \emptyset$, $\mathit{perms}(\mathbf{cleanup}) \stackrel{\mathrm{def}}{=} \{\mathit{FileIOPerm}\}$ and $\mathit{perms}(\mathbf{delete}) \stackrel{\mathrm{def}}{=} \{\mathit{FileIOPerm}\}$:

$$\begin{array}{rr}
\theta(\mathit{Rfl}) = \theta(\mathit{Trl}) \stackrel{\mathrm{def}}{=} \emptyset & \theta(\mathit{Rpl}) \stackrel{\mathrm{def}}{=} \lambda \overline{a} : \mathbb{N}.a_1 \cap .. \cap a_{10} \\
\theta(\mathit{Sym}) = \theta(\mathit{Abs}) \stackrel{\mathrm{def}}{=} \lambda a : \mathbb{N}.a & \theta(\beta) = \theta(\beta_{\Box}) \stackrel{\mathrm{def}}{=} \emptyset \\
\theta(\mathit{Trn}) = \theta(\mathit{App}) = \theta(\mathit{Let}) \stackrel{\mathrm{def}}{=} \lambda a : \mathbb{N}.\lambda b : \mathbb{N}.a \cap b & \theta(\delta_{\mathbf{f}}) \stackrel{\mathrm{def}}{=} \{\mathit{perms}(\mathbf{f})\}
\end{array}$$

Then evaluation of the term $!^{\alpha}_{\mathit{Rfl}(s)} \mathbf{bad}\ \alpha$ will produce a security exception since $\delta_{\mathbf{bad}}(s')$ occurs in the trail consulted by **delete**, for some s'. This term is based on the first example of Sec.4.1 in [AF03]. The second example of that same section (illustrating a case where stack inspection falls short and history based access control has advantages) consists in adding the top-level declaration $\mathbf{badTempFile} \doteq$ “$..\backslash \mathit{passwd}''$ and extending *perms* by declaring $\mathit{perms}(\mathbf{badTempFile}) \stackrel{\mathrm{def}}{=} \emptyset$. Then evaluation of $!^{\alpha}_{\mathit{Rfl}(s)} \mathbf{delete}\ \alpha\ \mathbf{badTempFile}$ will also produce a security exception.

7 Conclusions

We have presented a proof theoretical analysis of a functional computation model that keeps track of its computation history. A Curry-de Bruijn-Howard isomorphism of an affine fragment of Artemov's Justification Logic yields a lambda calculus $\lambda^{\natural}$ which models audited units of computation. Reduction in these units generates audit trails that are confined within them. Moreover, these units may look-up these trails and make decisions based on them. We prove type safety for $\lambda^{\natural}$ and strong normalisation for a restriction of it. It would be nice to lift the restriction in the proof of strong normalisation that M in the principle reduction axiom $\beta_{\Box}$ not have occurrences of the audited computation unit constructor "!". Also, it would make sense to study audited computation in a classical setting where, based on audit trail look-up, the current continuation could be disposed of in favour of a more judicious computation. Finally, although examples from the security domain seem promising more are needed in order to better evaluate the applicability of these ideas.

Acknowledgements. To Peter Thiemann for fruitful discussions.

References

[AA01] J. Alt and S. Artemov. Reflective λ-calculus. In *Proceedings of the Dagstuhl-Seminar on Proof Theory in CS*, volume 2183 of *LNCS*, 2001.

[AB07] S. Artemov and E. Bonelli. The intensional lambda calculus. In *LFCS*, volume 4514 of *LNCS*, pages 12–25. Springer, 2007.

[AF03] M. Abadi and C. Fournet. Access control based on execution history. In *NDSS*. The Internet Society, 2003.

[Art95] S. Artemov. Operational modal logic. Technical Report MSI 95-29, Cornell University, 1995.

[Art01] S. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, 2001.

[Art08] S. Artemov. Justification logic. In Steffen Hölldobler, Carsten Lutz, and Heinrich Wansing, editors, *JELIA*, volume 5293 of *Lecture Notes in Computer Science*, pages 1–4. Springer, 2008.

[Bar92] H. Barendregt. *Handbook of Logic in Computer Science*, chapter Lambda Calculi with Types. Oxford University Press, 1992.

[BF09] E. Bonelli and F. Feller. The logic of proofs as a foundation for certifying mobile computation. In *LFCS*, volume 5407 of *LNCS*, pages 76–91. Springer, 2009.

[Bre01] V. Brezhnev. On the logic of proofs. In *Proceedings of the Sixth ESSLLI Student Session*, pages 35–45, 2001.

[DP96] R. Davies and F. Pfenning. A modal analysis of staged computation. In *23rd POPL*, pages 258–270. ACM Press, 1996.

[DP01a] R. Davies and F. Pfenning. A judgmental reconstruction of modal logic. *Journal of MSCS*, 11:511–540, 2001.

[DP01b] R. Davies and F. Pfenning. A modal analysis of staged computation. *Journal of the ACM*, 48(3):555–604, May 2001.

[GTSKF04] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Automated termination proofs with aprove. In *RTA*, volume 3091 of *LNCS*, pages 210–220. Springer, 2004.

[ML96] P. Martin-Löf. On the meaning of the logical constants and the justifications of the logical laws. *Nordic J. of Philosophical Logic 1*, 1:11–60, 1996.

[NPP08] A. Nanevski, F. Pfenning, and B. Pientka. Contextual modal type theory. *ACM Trans. Comput. Log.*, 9(3), 2008.