Diagrams for Meaning Preservation*

J. B. Wells[1], Detlef Plump[2], and Fairouz Kamareddine[1]

[1] Heriot-Watt University
http://www.cee.hw.ac.uk/ultra/
[2] University of York
http://www.cs.york.ac.uk/~det/

Abstract. This paper presents an abstract framework and multiple diagram-based methods for proving *meaning preservation*, i.e., that all rewrite steps of a rewriting system preserve the meaning given by an operational semantics based on a rewriting strategy. While previous rewriting-based methods have generally needed the treated rewriting system as a whole to have such properties as, e.g., *confluence*, *standardization*, and/or *termination* or *boundedness* of *developments*, our methods can work when *all* of these conditions fail, and thus can handle more rewriting systems. We isolate the new *lift/project* diagram as the key proof idea and show that previous rewriting-based methods (Plotkin's method based on *confluence* and *standardization* and Machkasova and Turbak's method) implicitly use this diagram. Furthermore, our framework and proof methods help reduce the proof burden substantially by, e.g., supporting separate treatment of partitions of the rewrite steps, needing only *elementary diagrams* for rewrite step interactions, excluding many rewrite step interactions from consideration, needing weaker termination properties, and providing generic support for using developments in combination with any method.

1 Discussion

1.1 Background and Motivation

A programming language is defined as a set of programs and a way to evaluate (or "execute") the programs. It is increasingly popular to define evaluation via program rewriting [18, 5, 6, 7, 2, 13, 8, 19]. In this approach, evaluation rewrite rules are repeatedly applied at program positions given by *evaluation contexts* [5].

Other kinds of program rewriting than evaluation are also desirable. Potential uses of rewriting-based program transformations include optimizing compilers, partial evaluators, and program simplifiers. These transformations may use the already existing evaluation rules in arbitrary contexts or use additional rewrite rules. Some transformations may involve global reasoning about the entire program, but many are local and a good match for rewriting-based techniques.

* This work was partly supported by NSF grants CCR 9417382, CCR 9988529, and EIA 9806745, EPSRC grants GR/L 36963 and GR/R 41545/01, and Sun Microsystems equipment grant EDUD-7826-990410-US.

It is important to know when program transformations preserve a program's meaning as given by evaluation. There are many non-rewriting based approaches, such as denotational semantics (models), logical relations, applicative bisimulation and coinduction, etc., but they will not be discussed here because this paper focuses on rewriting-based techniques. Plotkin [18] first devised a rewriting-based method to prove meaning preservation for the call-by-name and call-by-value λ-calculus using *confluence* and *standardization*. At the same time, Plotkin proved that evaluation via rewriting was equivalent to evaluation via abstract machine. Subsequently, this approach has been applied to many systems, including systems with imperative features such as assignments and continuations (examples include [6, 7, 15, 2, 13, 8, 19, 11]).

Warning 1.1 (Not Quite Same as Observational Equivalence) What we call *meaning preservation* is related to *observational equivalence* (sometimes called *observational soundness*, *operational equivalence*, *consistency*, etc.), but is only the same for *contextually closed* rewriting systems. In this paper, two terms have the same *meaning* iff evaluating them yields the same result (divergence or the same halted state). Terms t_1 and t_2 are *observationally equivalent*, usually written $t_1 \simeq t_2$, iff $C[t_1]$ and $C[t_2]$ have the same meaning for every context C where $C[t]$ places t in the hole of the context C. Proving a rewriting relation R to be meaning preserving implies that $R \subseteq \simeq$ only when R is contextually closed. This paper presents an abstract (syntax-free) framework, and contexts are syntactic, so we do not discuss observational equivalence. □

1.2 Summary of Contributions

The existing rewriting-based tools for proving meaning preservation are difficult to use and sometimes completely inapplicable. To address this problem, this paper presents an abstract framework and multiple diagram-based methods for proving meaning preservation. The new knowledge presented here improves on what is already known as follows.

1. Our methods can be used for rewriting systems that as a whole fail to have confluence, standardization, and/or *termination* or *boundedness* of *developments*. While some of our methods ask for confluence or standardization-like properties, they do so only for subsets of all rewrite steps.
2. We isolate the new *lift/project* diagram (LP in definition 4.1) and show that it is the key proof idea for previous methods for proving meaning preservation (Plotkin's method based on *confluence* and *standardization* and Machkasova and Turbak's method based on *lift* and *project* [11]). We show that the confluence & standardization method is incomparable in proving power with the lift & project method. We present new lift/project-based methods that can handle systems that previous methods can not such as systems without standardization.
3. All of the proof methods dealt with in this paper (including the earlier methods of Plotkin and Machkasova & Turbak) are presented abstractly (free of syntax). Because our methods are abstract, there are no restrictions on

the kinds of rewrite rules used. Rewrite rules may be non-left-linear, non-orthogonal (overlapping), non-first-order, etc. Also, our approach does not need a notion of closed *programs* as a subset of terms.

4. All our methods support partitioning the rewrite steps into subsets treated separately with different methods as needed. These subsets need only be closed under (an informal notion of) "residuals with respect to evaluation steps". This partitioning also makes proving termination properties easier.
5. Our framework provides generic support for using developments (i.e., rewriting contracting only preexisting marked redexes) together with any method, so each method only needs to be used for marked rewrite steps. This eases the burden of proving termination properties. No notion of *residuals* is needed, which is important for systems with wildly overlapping rules where it is unclear how to define residuals.
6. In addition to a number of high-level diagram-based methods for proving meaning preservation, we also present low-level methods that are easier to use for people who are not researchers in rewriting. We give as many as possible of the details needed for the non-specialist to use and adapt the proof methods. These low-level methods use simple termination properties and diagrams.
   (a) Termination properties are only needed for ordinary rewriting, not for rewriting of rewrite step sequences (perhaps this should be called meta-rewriting?) as in some abstract standardization methods [14]. The different termination properties that each method requires are simple, ranging over strong (SN) and weak (WN) normalization and a bound on the number of evaluation steps in any rewrite sequence (BE in definition 5.1).
   (b) For analyzing rewrite step interactions, each method needs only the completion of *elementary diagrams*, i.e., diagrams where the only given edges are two adjacent single rewrite steps. In contrast, some abstract standardization methods require completing *cubes* [9, 14]. The method choice can depend on which elementary diagrams are completable. All of our methods exclude many rewrite step interactions from consideration.
7. To help rewriting researchers, as much as possible we identify intermediate diagrams to make it easier for new diagrams to be added as needed.
8. Our methods use only the simplest notion of standardization, that a rewrite sequence $t_1 \twoheadrightarrow t_2$ can be rearranged into a sequence $t_1 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_3 \xrightarrow{\mathbb{N}}\!\!\!\!\rightarrow t_2$ where $\mathbb{E}$ and $\mathbb{N}$ indicate respectively evaluation and non-evaluation steps. Standardization in the literature is a rich and interesting notion [14], but other standardization definitions always imply our definition and the extra details are not useful here, so they are omitted.

1.3 Acknowledgements

An early informal presentation by Elena Machkasova on the *lift* and *project* diagrams gave significant inspiration, although this work has subsequently proceeded independently. Zena Ariola is partly responsible for this work by convincing us to use rewrite rules for letrec that are difficult to prove correct.

2 Mathematical Definitions

Let $S \uplus S'$ denote $S \cup S$ if $S \cap S = \emptyset$ and otherwise be undefined. In a proof, "IH" means "by the induction hypothesis" and "w/o.l.o.g." means "without loss of generality".

Let R range over binary relations. Let $\xrightarrow{R}$ and $\underset{R}{\rightarrow}$ be alternate notations for R which are usable infix, i.e., both $a \xrightarrow{R} b$ and $a \underset{R}{\rightarrow} b$ stand for $R(a, b)$ which in turn stands for $(a, b) \in R$.

Define the following operators on binary relations. Let $R; R'$ be the composition of R with R' (i.e., $\{\, (a, b) \mid \exists c.\ R(a, c) \text{ and } R'(c, b) \,\}$). Let $\xrightarrow{R,0} = R^0$ be equality at the type intended for R. Let $\xrightarrow{R,i+1} = R^{i+1} = (R^i; R)$ when $0 \leq i$. Let $\xrightarrow{R,\geq k} = R^{\geq k} = \bigcup_{i \geq k} R^i$. Let $\xrightarrow{R,\leq k} = R^{\leq k} = \bigcup_{i \leq k} R^i$. Let $\xrightarrow{R,j,\leq k} = R^j \cap R^{\leq k}$ (useful in diagrams when j is existentially quantified). Let $\xrightarrow{R}\!\!\!\!\rightarrow = R^* = R^{\geq 0}$ (the transitive, reflexive closure). Let $\xleftarrow{R} = R^{-1}$ be the inverse of R (i.e., $\{\, (a, b) \mid R(b, a) \,\}$). Let $\xleftarrow{R}\!\!\!\!\leftarrow = (R^{-1})^{\geq 0}$. Let $\xleftrightarrow{R} = (R \cup R^{-1})$ (the symmetric closure). When $R = R^{-1}$ (i.e., R is symmetric), let $\xleftrightarrow{R} = R$. Let $\xleftrightarrow{R,k} = (\xleftrightarrow{R})^k$. Let $\xleftrightarrow{R}\!\!\!\!\leftrightarrow = (\xleftrightarrow{R})^{\geq 0}$.

Let an entity a *be* a R-*normal form*, written $\mathsf{is\text{-}nf}(R, a)$, iff there does not exist some entity b such that $a \xrightarrow{R} b$. Let an entity a *have* a R-normal form, written $\mathsf{has\text{-}nf}(R, a)$, iff there exists some b such that $a \xrightarrow{R}\!\!\!\!\rightarrow b$ and $\mathsf{is\text{-}nf}(R, b)$. Let $\xrightarrow{R,\mathsf{nf}}\!\!\!\!\rightarrow$ be the relation such that $a \xrightarrow{R,\mathsf{nf}}\!\!\!\!\rightarrow b$ iff $a \xrightarrow{R}\!\!\!\!\rightarrow b$ and $\mathsf{is\text{-}nf}(R, b)$. A relation R is *strongly normalizing*, written $\mathsf{SN}(R)$, iff for every entity a there is some $k \geq 0$ such that there does not exist an entity b such that $R^k(a, b)$. A relation R is *weakly normalizing*, written $\mathsf{WN}(R)$, iff for every entity a there is some entity b such that $a \xrightarrow{R,\mathsf{nf}}\!\!\!\!\rightarrow b$. Note that $\mathsf{SN}(R) \Rightarrow \mathsf{WN}(R)$.

Diagrams make statements about relations where solid and dotted edges indicate quantification. Metavariables already mentioned outside the diagram are unquantified. Other metavariables (e.g., for node names or used in edge labels) are universally quantified if attached to a solid edge and existentially quantified if attached only to dotted edges. As an example, in a context where R_1 and R_2 have already been given, the following equivalence holds:

$$\begin{tikzcd} a \arrow[r, "R_1,k"] \arrow[d, "R_2"'] & b \arrow[d, dotted, "R_1"] \\ c \arrow[r, dotted, "R_1,\leq k"] & d \end{tikzcd} \iff \forall a, b, c, k.\ (a \xrightarrow{R_1,k} b \wedge a \xrightarrow{R_2} c) \Rightarrow \exists d.\ c \xrightarrow{R_1,\leq k} d \wedge b \xrightarrow{R_1}\!\!\!\!\rightarrow d$$

In proofs, the reason for each diagram polygon will usually be written inside it.

3 Abstract Evaluation Systems

An *abstract evaluation system* (AES) is a tuple

$$(\mathbb{T}, \mathbb{S}, \mathbb{R}, \mathsf{endpoints}, \mathbb{E}, \mathsf{result})$$

satisfying the conditions given below by axioms 3.3 and 3.4 and the immediately following conditions. The carriers of an AES are the sets $\mathbb{T}$, $\mathbb{S}$, and $\mathbb{R}$. The function $\mathsf{endpoints}$ maps $\mathbb{S}$ to $\mathbb{T} \times \mathbb{T}$. The set $\mathbb{E}$ is a subset of $\mathbb{S}$. The function result maps $\mathbb{T}$ to $\mathbb{R}$. Let t range over $\mathbb{T}$, let s range over $\mathbb{S}$, let r range over $\mathbb{R}$, and let $\mathcal{S}$ range over subsets of $\mathbb{S}$.

The intended meaning is as follows. $\mathbb{T}$ should be a set of terms. $\mathbb{S}$ should be a set of rewrite steps. $\mathbb{R}$ should be a set of evaluation results which by axiom 3.4(1) will most likely contain the symbol $\mathsf{diverges}$ and one or more other members, typically symbols such as halt, error, etc. The halt case might be subdivided into possible constant values of final results. If $\mathsf{endpoints}(s) = (t_1, t_2)$, this should mean that step s rewrites term t_1 to term t_2. The members of $\mathbb{E}$ are the rewrite steps used for evaluation. Let $\mathbb{N} = \mathbb{S} \setminus \mathbb{E}$ (where "$\mathbb{N}$" stands for "non-evaluation"). If $\mathsf{result}(t) = r$, this should mean that r is the observable result of evaluating term t, where $\mathsf{diverges}$ is reserved by axiom 3.4(1) for non-halting evaluations.

Convention 3.1 *In this paper, wherever no specific AES is being considered, statements are about every possible AES.* □

Let rewriting notation be defined as follows. Given a rewrite step set $\mathcal{S}$, let $\llbracket\mathcal{S}\rrbracket$ be the binary relation $\{\, (t, t') \mid \exists s \in \mathcal{S}.\ \mathsf{endpoints}(s) = (t, t') \,\}$. Thus, $t \xrightarrow{\llbracket\mathcal{S}\rrbracket} t'$ iff there exists $s \in \mathbb{S}$ such that $\mathsf{endpoints}(s) = (t, t')$. When a rewrite step set $\mathcal{S}$ is used in a context *requiring* a binary relation on $\mathbb{T}$, then let $\mathcal{S}$ implicitly stand for $\llbracket\mathcal{S}\rrbracket$. Thus, as examples, $t \xrightarrow{\mathcal{S}} t'$ stands for $t \xrightarrow{\llbracket\mathcal{S}\rrbracket} t'$ and an $\mathcal{S}$-normal form is simply a $\llbracket\mathcal{S}\rrbracket$-normal form. When used in a position *requiring* a subset of $\mathbb{S}$ or a binary relation on $\mathbb{T}$, let s stand for $\{s\}$ and let $\mathcal{S}, \mathcal{S}'$ stand for $\mathcal{S} \cap \mathcal{S}'$. Thus, as an example, $t \xrightarrow{\mathcal{S}, s} t'$ stands for $t \xrightarrow{\llbracket\mathcal{S} \cap \{s\}\rrbracket} t'$. When a binary relation on $\mathbb{T}$ is *required* and none is supplied, then let the relation $\llbracket\mathbb{S}\rrbracket$ be implicitly supplied. Thus, as examples, $t \longrightarrow t'$ stands for $t \xrightarrow{\llbracket\mathbb{S}\rrbracket} t'$ and $t \xrightarrow{k} t'$ stands for $t \xrightarrow{\llbracket\mathbb{S}\rrbracket, k} t'$.

Definition 3.2 (Rewrite Step Set Properties). *Define the following rewrite step sets and properties of rewrite step sets:*

Standardization:

$$\mathsf{Std}(\mathcal{S}, \mathcal{S}') \iff \begin{tikzcd} t_1 \arrow[rr, two heads, "\mathcal{S}"] \arrow[dr, dotted, "\mathbb{E},\mathcal{S}'"'] & & t_2 \\ & t_3 \arrow[ur, dotted, "\mathbb{N},\mathcal{S}'"'] & \end{tikzcd}$$

Confluence:

$$\mathsf{Conf}(\mathcal{S}) \iff \begin{tikzcd} t_1 \arrow[rr, leftrightarrow, "\mathcal{S}"] \arrow[dr, dotted, "\mathcal{S}"'] & & t_2 \arrow[dl, dotted, "\mathcal{S}"] \\ & t_3 & \end{tikzcd}$$

Local Confluence:

$$\mathsf{LConf}(\mathcal{S}) \iff \begin{tikzcd} t_1 \arrow[r, "\mathcal{S}"] \arrow[d, "\mathcal{S}"'] & t_4 \arrow[d, dotted, "\mathcal{S}"] \\ t_2 \arrow[r, two heads, dotted] & t_3 \end{tikzcd}$$

Meaning Preservation:

$$s \in \mathsf{MP} \iff \begin{tikzcd} t_1 \arrow[d, "s"'] \arrow[dr, dotted, "\mathsf{result}"] & \\ t_2 \arrow[r, dotted, "\mathsf{result}"'] & r \end{tikzcd}$$

Subcommutativity:

$$\mathsf{SubComm}(\mathcal{S}, i, j) \iff \begin{tikzcd} t_1 \arrow[r, "\mathcal{S},j"] \arrow[d, "\mathcal{S},i"'] & t_3 \arrow[d, dotted, "\mathcal{S},\leq i"] \\ t_2 \arrow[r, dotted, "\mathcal{S},\leq j"'] & t_4 \end{tikzcd}$$

Let $\mathsf{Std}(\mathcal{S})$ *abbreviate* $\mathsf{Std}(\mathcal{S}, \mathbb{S})$. *Let* $\mathsf{SubComm}(\mathcal{S})$ *abbreviate* $\mathsf{SubComm}(\mathcal{S}, 1, 1)$. *Traditionally, only* $\mathsf{Std}(\mathbb{S}) = \mathsf{Std}(\mathbb{S}, \mathbb{S})$ *is considered. The simple definition of* MP

is reasonable because axiom 3.4(1) (given below) means MP *implies preservation of the existence of* $\mathbb{E}$*-normal forms. See also warning 1.1 and do not confuse* MP *with observational equivalence.* □

Axiom 3.3 (Subcommutativity of Evaluation) $\mathsf{SubComm}(\mathbb{E})$. □

Non-deterministic evaluation is needed for rewriting systems with non-deterministic syntax, e.g., the system of [11] where the top syntax level is a set with unordered components. Often, it will be simpler to make evaluation deterministic so that $t_2 \xleftarrow{\mathbb{E},s_1} t_1 \xrightarrow{\mathbb{E},s_2} t_3$ implies that $t_2 = t_3$.

Axiom 3.3 does not ensure that any strategy for $\xrightarrow{\mathbb{E}}$ will find $\mathbb{E}$-normal forms when they exist. Strengthening axiom 3.3 so that the bottom and right diagram edges have the same length would ensure this, but is not needed otherwise.

Axiom 3.4 (Evaluation Sanity)

1. *"*$\mathsf{diverges}$*" **Means Evaluation Diverges:***
   $\mathsf{result}(t) = \mathsf{diverges} \Leftrightarrow \neg\mathsf{has\text{-}nf}(\mathbb{E}, t)$.
2. ***Evaluation Steps Preserve Meaning:***
   $\mathbb{E} \subseteq \mathsf{MP}$.
3. ***Non-Evaluation Steps Preserve Evaluation Steps:***

$$\begin{tikzcd} t_1 \arrow[r, "\mathbb{E}"] \arrow[d, leftrightarrow, "\mathbb{N}"'] & t_3 \\ t_2 \arrow[r, dotted, "\mathbb{E}"] & t_4 \end{tikzcd}$$

   Consequently, if $t_1 \xrightarrow{\mathbb{N}} t_2$*, then* $\mathsf{is\text{-}nf}(\mathbb{E}, t_1) \Leftrightarrow \mathsf{is\text{-}nf}(\mathbb{E}, t_2)$.
4. ***Non-Evaluation Steps on*** $\mathbb{E}$***-Normal Forms Preserve Meaning:***
   If $t \xrightarrow{\mathbb{N}} t'$ *and* $\mathsf{is\text{-}nf}(\mathbb{E}, t)$*, then* $\mathsf{result}(t) = \mathsf{result}(t')$. □

When defining an AES for a rewriting system, it is trivial to satisfy axioms 3.4(1) and 3.4(2) by using an auxiliary function result' which maps $\{\, t \mid \mathsf{is\text{-}nf}(\mathbb{E}, t) \,\}$ to $\mathbb{R} \setminus \{\mathsf{diverges}\}$ and defining result as follows:

$$\mathsf{result}(t) = \begin{cases} \mathsf{diverges} & \text{if } \neg\mathsf{has\text{-}nf}(\mathbb{E}, t), \\ \mathsf{result}'(t') & \text{if } t \xrightarrow{\mathbb{E},\mathsf{nf}}\!\!\!\!\rightarrow t'. \end{cases}$$

Indeed, the model of how evaluation should be computed expects to work this way. When $\neg\mathsf{is\text{-}nf}(\mathbb{E}, t)$, it is expected that computing $\mathsf{result}(t)$ involves first finding t' such that $t \xrightarrow{\mathbb{E},\mathsf{nf}}\!\!\!\!\rightarrow t'$, then computing $\mathsf{result}(t')$, and otherwise diverging if no such t' exists. Thus, the value of $\mathsf{result}(t)$ is unimportant if $\neg\mathsf{has\text{-}nf}(\mathbb{E}, t)$. Reserving the value $\mathsf{diverges}$ for this case simplifies things.

Satisfying axioms 3.4(3) and 3.4(4) requires more care in the design of the rewriting system and the AES, but is not hard. Anyway, axiom 3.4(3) is a consequence of the properties WL1 and WP1 or WLP1 (definition 5.1) which typically must also be proven. The condition of axiom 3.4(3) appears in other abstract frameworks as early as [9] and appears in non-abstract form in [18]. The first explicit statement of the condition of axiom 3.4(4) that we are aware of appears in [10], although the condition is partially present in [2].

Lemma 3.5 (Non-Evaluation Steps on Eval-Normal Forms). *If* $t_1 \overset{\mathbb{N}}{\twoheadleftarrow\!\twoheadrightarrow} t_2$ *and* $\mathsf{is\text{-}nf}(\mathbb{E}, t_1)$*, then* $t_1 \overset{\mathsf{MP}}{\twoheadleftarrow\!\twoheadrightarrow} t_2$*.* □

Proof. See appendix A. □

4 Lift/Project Diagrams for Meaning Preservation

This section presents properties of rewrite step sets in definition 4.1 and shows how to use them to prove *meaning preservation*, the important connection between arbitrary-strategy rewriting and evaluation. When evaluation is defined by a subset of the rewrite steps (specified in an AES by the set $\mathbb{E}$), it is necessary to show that arbitrary rewriting preserves the evaluation result in order to have confidence that the non-evaluation rewrite steps are at all meaningful. Traditionally, this has been done by proving confluence (Conf) and standardization (Std), the preconditions of Plotkin's approach [18] (presented in lemma 4.5(1,2)).

Needing confluence and standardization is a big weakness, as shown by the non-confluent system in [11] and the $\lambda^{:=,\mathsf{letrec}}$ calculus we mention in section 9 which has neither confluence nor standardization. In contrast, our new method in theorem 4.3 needs only the lift/project (LP) property. By lemma 4.2, lift/project can be obtained from the lift (Lift) and project (Proj) properties. Because lift and project do not imply confluence (lemma 4.5(2)), theorem 4.3 does not need confluence. Furthermore, because lift/project implies neither lift nor project (lemma 4.2(8)) and lift is equivalent to standardization (lemma 4.4(1)), theorem 4.3 does not need standardization when lift is not used.

Theorem 4.3 differs from earlier work of Machkasova and Turbak [11] in several important ways. First, it is abstract (syntax-free). Second, it makes it easy to treat rewrite step subsets independently. This vastly simplifies auxiliary termination proofs (e.g., for properties SN or BE as used in definition 5.1) and is vital when a single method fails to cover all $\mathbb{N}$ steps (e.g., section 9). Third, it needs only the weaker lift/project property rather than lift and project. This is vital because lift is equivalent to standardization so the Machkasova/Turbak method fails for systems without standardization (e.g., section 9).

Definition 4.1 (Lift, Project, and Related Properties). *Define the following rewrite step sets and properties of rewrite step sets:*

Strong Lift:

$$\mathsf{SLift}(\mathcal{S}) \iff \begin{array}{ccc} t_1 & \overset{\mathbb{E}}{\cdots\!\twoheadrightarrow} & t_4 \\ {\scriptstyle \mathbb{N},\mathcal{S}}\downarrow & & \vdots\,{\scriptstyle \mathbb{N},\mathcal{S}} \\ t_2 & \overset{\mathbb{E}}{\twoheadrightarrow} & t_3 \end{array}$$

Lift:

$$s \in \mathsf{Lift} \iff \begin{array}{ccc} t_1 & \overset{\mathbb{E}}{\cdots\!\twoheadrightarrow} & t_4 \\ s\downarrow & & \vdots\,{\scriptstyle \mathbb{N}} \\ t_2 & \overset{\mathbb{E}}{\twoheadrightarrow} & t_3 \end{array}$$

Lift′:

$$s \in \mathsf{Lift}' \iff \begin{array}{ccccc} t_1 & & \overset{\mathbb{E}}{\cdots\cdots\twoheadrightarrow} & & t_4 \\ s\downarrow & & & & \vdots\,{\scriptstyle \mathbb{N}} \\ t_2 & \overset{\mathbb{E}}{\twoheadrightarrow} & t_3 & \overset{\mathbb{E}}{\cdots\!\twoheadrightarrow} & t_5 \end{array}$$

Strong Project:

$$\mathsf{SProj}(\mathcal{S}) \iff \begin{array}{ccccc} t_1 & \overset{\mathbb{E}}{\twoheadrightarrow} & t_2 & \overset{\mathbb{E}}{\cdots\!\twoheadrightarrow} & t_4 \\ {\scriptstyle \mathbb{N},\mathcal{S}}\downarrow & & & & \vdots\,{\scriptstyle \mathbb{N},\mathcal{S}} \\ t_3 & & \overset{\mathbb{E}}{\cdots\cdots\twoheadrightarrow} & & t_5 \end{array}$$

Project:

$$s \in \mathsf{Proj} \iff \begin{array}{ccccc} t_1 & \overset{\mathbb{E}}{\twoheadrightarrow} & t_2 & \overset{\mathbb{E}}{\cdots\!\twoheadrightarrow} & t_4 \\ s\downarrow & & & & \vdots\,{\scriptstyle \mathbb{N}} \\ t_3 & & \overset{\mathbb{E}}{\cdots\cdots\twoheadrightarrow} & & t_5 \end{array}$$

Strong Lift/Project:

$$\mathsf{SLP}(\mathcal{S}) \iff \begin{tikzcd} t_1 \arrow[r, two heads] \arrow[d, leftrightarrow, "\mathbb{N},\mathcal{S}"'] & t_3 \arrow[r, dotted, two heads, "\mathbb{E}"] & t_5 \arrow[d, leftrightarrow, dotted, "\mathbb{N},\mathcal{S}"] \\ t_2 \arrow[rr, dotted, two heads, "\mathbb{E}"] & & t_4 \end{tikzcd}$$

Lift/Project:

$$s \in \mathsf{LP} \iff \begin{tikzcd} t_1 \arrow[r, two heads] \arrow[d, leftrightarrow, "s"'] & t_3 \arrow[r, dotted, two heads, "\mathbb{E}"] & t_5 \arrow[d, leftrightarrow, dotted, "\mathbb{N}"] \\ t_2 \arrow[rr, dotted, two heads, "\mathbb{E}"] & & t_4 \end{tikzcd}$$

Lift/Project when Terminating:

$$s \in \mathsf{LPT} \iff \begin{tikzcd} t_1 \arrow[r, two heads, "\mathbb{E},\mathsf{nf}"] \arrow[d, leftrightarrow, "s"'] & t_3 \arrow[d, leftrightarrow, dotted, "\mathbb{N}"] \\ t_2 \arrow[r, dotted, two heads, "\mathbb{E}"] & t_4 \end{tikzcd}$$

□

The Lift and Proj properties given here match the properties by the names "Lift" and "Project" in [11], except that there both properties are defined on the entire rewriting system rather than on individual rewrite steps and both properties specify the step on the left diagram edge to be a $\mathbb{N}$ step (an inessential difference). Only the weaker Lift' which is symmetrical with Proj is actually needed together with Proj to obtain LP (lemma 4.2(7)). However, Lift' can not replace Lift in the statement of lemma 4.4(1).

Lemma 4.2 (Relationships between Lift and Project Properties).

1. $\mathbb{E} \subseteq \mathsf{Lift} \cap \mathsf{Proj}$.
2. *If* $\mathsf{SLift}(\mathcal{S})$, *then* $\mathcal{S} \subseteq \mathsf{Lift}$.
3. *If* $\mathsf{SProj}(\mathcal{S})$, *then* $\mathcal{S} \subseteq \mathsf{Proj}$.
4. *If* $\mathsf{SLP}(\mathcal{S})$, *then* $\mathcal{S} \subseteq \mathsf{LP}$.
5. $\mathsf{Lift} \subseteq \mathsf{Lift}'$.
6. $\mathsf{Lift}' \subseteq \mathsf{Lift}$ *need not be true.*
7. $\mathsf{Lift}' \cap \mathsf{Proj} \subseteq \mathsf{LP}$.
8. *None of* $\mathsf{LP} \subseteq \mathsf{Lift}' \cap \mathsf{Proj}$, $\mathsf{LP} \subseteq \mathsf{Lift}'$, *and* $\mathsf{LP} \subseteq \mathsf{Proj}$ *need to be true.*
9. $\mathsf{LP} \subseteq \mathsf{LPT}$.
10. $\mathsf{LPT} \subseteq \mathsf{LP}$ *need not be true.* □

Proof. See appendix B. □

Theorem 4.3 (Relationships between Lift, Project, and Meaning Preservation).

1. $\mathsf{LPT} \subseteq \mathsf{MP}$.
2. $\mathsf{MP} \subseteq \mathsf{LPT}$ *need not be true.* □

Proof.

1. Suppose $s \in \mathsf{LPT}$. Let $t_1 \xrightarrow{s} t_2$. Suppose neither $\mathsf{has\text{-}nf}(\mathbb{E}, t_1)$ nor $\mathsf{has\text{-}nf}(\mathbb{E}, t_2)$. By axiom 3.4(1), it holds that $\mathsf{result}(t_1) = \mathsf{diverges} = \mathsf{result}(t_2)$, so $s \in \mathsf{MP}$. Suppose instead that either $\mathsf{has\text{-}nf}(\mathbb{E}, t_1)$ or $\mathsf{has\text{-}nf}(\mathbb{E}, t_2)$. Suppose $\mathsf{has\text{-}nf}(\mathbb{E}, t_1)$ (w/o.l.o.g. because only $t_1 \xleftrightarrow{s} t_2$ is used). Then $t_1 \xrightarrow{\mathbb{E},\mathsf{nf}}\!\!\!\!\rightarrow t_3$ for some t_3. By $s \in \mathsf{LPT}$, it holds that $t_3 \xleftrightarrow{\mathbb{N}}\!\!\!\!\rightarrow t_4 \xleftarrow{\mathbb{E}} t_2$ for some t_4. Because $\mathsf{is\text{-}nf}(\mathbb{E}, t_3)$, by lemma 3.5 it holds that $t_3 \xleftrightarrow{\mathsf{MP}}\!\!\!\!\rightarrow t_4$. By axiom 3.4(2) and induction on the lengths of rewrite sequences, it holds that $t_1 \xrightarrow{\mathsf{MP}}\!\!\!\!\rightarrow t_3$ and $t_2 \xrightarrow{\mathsf{MP}}\!\!\!\!\rightarrow t_4$. Thus, $t_1 \xleftrightarrow{\mathsf{MP}}\!\!\!\!\rightarrow t_2$. Thus, $s \in \mathsf{MP}$.

2. Consider this 4-term 3-step AES where all results are the same:

$$\begin{tikzcd} t_1 \arrow[r, "\mathbb{E}, s_1"] \arrow[d, "\mathbb{N}, s_2"'] & t_2 \\ t_3 \arrow[r, "\mathbb{E}, s_3"] & t_4 \end{tikzcd}$$

Then $\mathsf{MP} = \mathbb{S}$, but $\mathsf{MP} \setminus \mathsf{LPT} = \{s_2\}$. □

4.1 Comparison with Traditional Approach

This subsection compares the lift & project and lift/project methods with the traditional confluence & standardization method. Plotkin's traditional approach was separated out and presented abstractly by Machkasova [10] in a form similar to the combination of the proofs of lemma 4.5(1), lemma 4.2(9), and theorem 4.3(1). We have done the work needed to allow the argument be easily applied to subsets of $\mathbb{S}$ and reformulated it for the AES framework. Furthermore, we have factored the argument to show it goes through LP (lemma 4.5(1)) and LPT before reaching MP. Thus, it appears that the main previously known rewriting-based methods of showing meaning preservation implicitly use the lift/project diagram. Interestingly, in lemma 4.5(3,4) it is shown that the confluence & standardization method and the lift & project method are incomparable in their power; each can address problems that the other can not. Section 5 will develop another method ($\mathsf{WB}\backslash\mathsf{Std}$ in definition 5.1) of proving lift/project which can address yet more problems, because it does not require standardization.

The following equivalence of Lift and standardization in lemma 4.4(1) appears in [10], although here it has been parameterized on rewrite step sets.

Lemma 4.4 (Lift Equivalent to Standardization).

1. *$\mathcal{S} \subseteq \mathsf{Lift}$ iff $\mathsf{Std}(\mathcal{S} \cup \mathbb{E})$. (Consequently, $\mathsf{Lift} = \mathbb{S}$ iff $\mathsf{Std}(\mathbb{S})$.)*
2. *The above statement need not be true with Lift replaced by Lift'.* □

Proof.

1. $\mathsf{Std}(\mathcal{S} \cup \mathbb{E}) \Rightarrow \mathcal{S} \subseteq \mathsf{Lift}$ is immediate. $\mathcal{S} \subseteq \mathsf{Lift} \Rightarrow \mathsf{Std}(\mathcal{S} \cup \mathbb{E})$ is proven by induction on the length of rewrite sequences. See appendix B for full details.
2. Consider this 6-term 6-step AES where all results are the same:

$$\begin{tikzcd} t_1 \arrow[r, "\mathbb{E}, s_1"] \arrow[d, "\mathbb{N}, s_2"'] & t_2 \arrow[dr, "\mathbb{N}, s_3"] & \\ t_3 \arrow[r, "\mathbb{E}, s_4"] & t_4 \arrow[r, "\mathbb{E}, s_5"'] \arrow[d, "\mathbb{N}, s_6"'] & t_5 \\ & t_6 & \end{tikzcd}$$

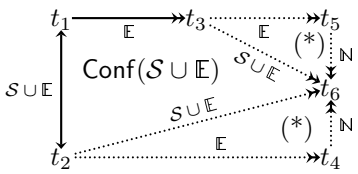
Note that $\mathsf{Lift}' = \mathbb{S}$, but $\mathsf{Lift}' \backslash \mathsf{Lift} = \{s_2\}$. The desired $\mathsf{Std}(\mathbb{S})$ is false, because $t_1 \twoheadrightarrow t_6$ but there is no t such that $t_1 \overset{\mathbb{E}}{\twoheadrightarrow} t \overset{\mathbb{N}}{\twoheadrightarrow} t_6$. □

Lemma 4.5 (Relationships between Confluence + Standardization and Lift + Project).

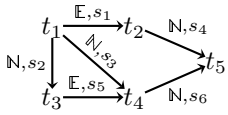
1. *If* $\mathsf{Conf}(\mathcal{S} \cup \mathbb{E})$ *and* $\mathsf{Std}(\mathcal{S} \cup \mathbb{E})$*, then* $\mathcal{S} \subseteq \mathsf{LP}$*.*
2. *Consequently,* $\mathsf{Conf}(\mathcal{S} \cup \mathbb{E})$ *and* $\mathcal{S} \subseteq \mathsf{Lift}$ *imply* $\mathcal{S} \subseteq \mathsf{MP}$*.*
3. *If* $\mathsf{Conf}(\mathcal{S} \cup \mathbb{E})$ *and* $\mathsf{Std}(\mathcal{S} \cup \mathbb{E})$*, then* $\mathcal{S} \subseteq \mathsf{Proj}$ *need not be true.*
4. $\mathsf{Conf}(\mathsf{Lift} \cap \mathsf{Proj})$ *need not be true.* □

Proof.

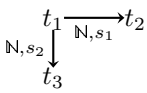
1. Suppose that $\mathsf{Conf}(\mathcal{S} \cup \mathbb{E})$ and (*) $\mathsf{Std}(\mathcal{S} \cup \mathbb{E})$ hold. Using the reason (*) as indicated, the following diagram proves $\mathcal{S} \cup \mathbb{E} \subseteq \mathsf{LP}$ and thus $\mathcal{S} \subseteq \mathsf{LP}$:



2. By lemmas 4.2(9), 4.4, and 4.5(1) and theorem 4.3(1).
3. Consider this 5-term 6-step AES where all results are the same:



   Then $\mathsf{Conf}(\mathbb{N} \cup \mathbb{E})$ and $\mathsf{Std}(\mathbb{N} \cup \mathbb{E})$, but $\mathbb{N} \setminus \mathsf{Proj} = \{s_2\}$.
4. Consider this 3-term 2-step AES where all results are the same:



   Then $\mathsf{Lift} = \mathsf{Proj} = \mathbb{S}$, but $\neg\mathsf{Conf}(\mathbb{S})$. □

5 Elementary Diagrams for Strong Lift/Project

According to section 4, one can prove rewrite step sets to have the lift/project property in order to prove meaning preservation. Furthermore, lift/project can be obtained via stronger properties such as the lift and project properties. However, proving these properties can be very difficult.

To help, this section provides abstract methods for proving strong lift, strong project, and/or strong lift/project for particular rewrite step sets. Definition 5.1 defines that a rewrite step set is *well behaved* when it satisfies either the $\mathsf{WB{+}Std}$ or $\mathsf{WB{\setminus}Std}$ properties. In turn, each of these are conjunctions of a small number of specific properties, one termination property and some *elementary diagrams*, i.e., diagrams where the given edges are two adjacent single rewrite steps. The $\mathsf{WB{+}Std}$ and $\mathsf{WB{\setminus}Std}$ properties are about rewrite step *sets* rather than individual steps because it is necessary to simultaneously treat all the steps in a set that is closed under (a very informal notion of) "residuals with respect to evaluation steps". This section's main result (theorem 5.4) is that a well behaved

rewrite step set $\mathcal{S}$ has either the strong lift and strong project properties or the strong lift/project property.

Each of $\mathsf{WB{+}Std}$ and $\mathsf{WB\backslash Std}$ has particular advantages. The termination property of $\mathsf{WB{+}Std}$ requires only a bound on the number of $\mathbb{E}$ steps in a rewrite sequence (BE), not full termination. When used together with the methods of section 6, this is significantly weaker than the *finite developments* property needed by some other proof methods, because it allows infinite developments (and there is no requirement that coinitial developments can be completed to be cofinal). In contrast, $\mathsf{WB\backslash Std}$ requires a strong termination property, but replaces the $\mathsf{WL1}$ and $\mathsf{WP1}$ elementary diagrams with the weaker diagram $\mathsf{WLP1}$. The big advantage of $\mathsf{WLP1}$ is that it does not require standardization. Although proving $\mathsf{WB\backslash Std}(\mathcal{S})$ does require showing strong normalization and local confluence for $\mathcal{S}$, if confluence can be shown for $\mathcal{S}$ by other means then strong normalization can be weakened to weak normalization and the local confluence requirement can be dropped (lemmas 5.2(3) and 5.3(3)).

Definition 5.1 (Well Behaved Rewrite Step Sets). *Let* $\mathsf{N}^*\mathsf{EN}^*(\mathcal{S})$ *be the relation* $\xrightarrow{\mathbb{N},\mathcal{S}}\!\!\!\!\rightarrow ; \xrightarrow{\mathbb{E},\mathcal{S}} ; \xrightarrow{\mathbb{N},\mathcal{S}}\!\!\!\!\rightarrow$. *Define the following rewrite step set properties:*

Bounded $\mathbb{E}$*-Steps:*

$$\mathsf{BE}(\mathcal{S}) \iff \mathsf{SN}(\mathsf{N}^*\mathsf{EN}^*(\mathcal{S}))$$

$\mathbb{N}$*-Steps Do Not Create* $\mathbb{E}$*-Steps:*

$$\mathsf{NE}(\mathcal{S}) \iff \begin{tikzcd} t_1 \arrow[r, dotted, "\mathbb{E},\mathcal{S}"] \arrow[d, "\mathbb{N},\mathcal{S}"'] & t_4 \\ t_2 \arrow[r, "\mathbb{E},\mathcal{S}"] & t_3 \end{tikzcd}$$

Weak Lift 1-Step:

$$\mathsf{WL1}(\mathcal{S},\mathcal{S}') \iff \begin{tikzcd} t_1 \arrow[r, dotted, "\mathbb{E},\mathcal{S}'"] \arrow[d, "\mathbb{N},\mathcal{S}"'] & t_4 \arrow[d, dotted, "\mathcal{S}"] \\ t_2 \arrow[r, "\mathbb{E},\mathcal{S}'"] & t_3 \end{tikzcd}$$

Weak Project 1-Step:

$$\mathsf{WP1}(\mathcal{S}) \iff \begin{tikzcd} t_1 \arrow[r, "\mathbb{E}"] \arrow[d, "\mathbb{N},\mathcal{S}"'] & t_2 \arrow[d, dotted, "\mathcal{S}"] \\ t_3 \arrow[r, dotted, "\mathbb{E}"] & t_4 \end{tikzcd}$$

Weak Lift/Project 1-Step:

$$\mathsf{WLP1}(\mathcal{S}) \iff \begin{tikzcd} t_1 \arrow[r, "\mathbb{E}"] \arrow[d, "\mathbb{N},\mathcal{S}"'] & t_4 \arrow[d, dotted, "\mathcal{S}"] \\ t_2 \arrow[r, dotted, "\mathbb{E}"] & t_3 \end{tikzcd}$$

Standardization to Normal Form:

$$\mathsf{Std\text{-}nf}(\mathcal{S}) \iff \begin{tikzcd} t_1 \arrow[rr, "\mathcal{S},\mathsf{nf}"] \arrow[dr, dotted, "\mathbb{E},\mathcal{S},\mathsf{nf}"'] & & t_2 \\ & t_3 \arrow[ur, dotted, "\mathbb{N},\mathcal{S},\mathsf{nf}"'] & \end{tikzcd}$$

Well Behaved with Standardization:

$$\mathsf{WB{+}Std}(\mathcal{S}) \iff \mathsf{BE}(\mathcal{S}) \wedge \mathsf{WL1}(\mathcal{S},\mathcal{S}) \wedge \mathsf{WL1}(\mathcal{S},\mathbb{S}) \wedge \mathsf{WP1}(\mathcal{S})$$

Well Behaved without Standardization:

$$\mathsf{WB\backslash Std}(\mathcal{S}) \iff \mathsf{SN}(\mathcal{S}) \wedge \mathsf{LConf}(\mathcal{S}) \wedge \mathsf{NE}(\mathcal{S}) \wedge \mathsf{WLP1}(\mathcal{S})$$

□

Lemma 5.2 (Confluence and Standardization-Like Properties).

1. *If* $\mathsf{BE}(\mathcal{S})$ *and* $\mathsf{WL1}(\mathcal{S},\mathcal{S})$*, then* $\mathsf{Std}(\mathcal{S},\mathcal{S})$*.*
2. *If* $\mathsf{LConf}(\mathcal{S})$ *and* $\mathsf{SN}(\mathcal{S})$*, then* $\mathsf{Conf}(\mathcal{S})$ *(Newman's Lemma).*
3. *If* $\mathsf{Conf}(\mathcal{S})$*,* $\mathsf{WN}(\mathcal{S})$*, and* $\mathsf{NE}(\mathcal{S})$*, then* $\mathsf{Std\text{-}nf}(\mathcal{S})$*.* □

Proof. By definition 5.1 and lemmas C.2, C.7, and C.8 in appendix C. □

Lemma 5.3 (Strong Lift and Project Properties).

1. *If* $\mathsf{WL1}(\mathcal{S},\mathbb{S})$ *and* $\mathsf{Std}(\mathcal{S},\mathcal{S})$*, then* $\mathsf{SLift}(\mathcal{S})$*.*

2. *If* $\mathsf{WP1}(\mathcal{S})$ *and* $\mathsf{Std}(\mathcal{S},\mathcal{S})$*, then* $\mathsf{SProj}(\mathcal{S})$*.*
3. *If* $\mathsf{Conf}(\mathcal{S})$*,* $\mathsf{WN}(\mathcal{S})$*,* $\mathsf{Std\text{-}nf}(\mathcal{S})$*, and* $\mathsf{WLP1}(\mathcal{S})$*, then* $\mathsf{SLP}(\mathcal{S})$*.* □

Proof. By definition 5.1 and lemmas C.4, C.6, and C.11 in appendix C. □

Theorem 5.4 (Well Behaved Rewrite Step Sets).

1. *If* $\mathsf{WB}{+}\mathsf{Std}(\mathcal{S})$*, then* $\mathsf{SLift}(\mathcal{S})$ *and* $\mathsf{SProj}(\mathcal{S})$*.*
2. *If* $\mathsf{WB}\backslash\mathsf{Std}(\mathcal{S})$*, then* $\mathsf{SLP}(\mathcal{S})$*.* □

Proof. By definition 5.1 and lemmas 5.2 and 5.3. □

6 Marked Rewriting and Developments

Sometimes, a desired termination property (e.g., BE from definition 5.1, SN, or WN) fails for a step set $\mathcal{S}$ generated by some rewrite rule(s), but holds for $\mathcal{S} \cap \mathbb{M}$ where $\mathbb{M}$ is a set of *marked* steps. The marks will typically force termination by forbidding contraction of unmarked redexes and ensuring that "created" redexes will be unmarked. To use this method, the desired rewriting system is embedded in a larger marked system with additional marked terms and rewrite steps, so proving the larger system correct also proves the desired system correct.

This section defines conditions on marking and theorem 6.4 proves that when these conditions hold, proving LPT for $\mathcal{S} \cap \mathbb{M}$ (i.e., the marked fragment of the larger marked system) is sufficient to prove LPT for $\mathcal{S}$ (i.e., both the marked and unmarked steps in the larger system). Thus, when any of this paper's methods for proving meaning preservation work for $\mathcal{S} \cap \mathbb{M}$, the methods also work for $\mathcal{S}$. It is worth observing that the style of proof of theorem 6.4 can be repeated for many properties other than LPT, e.g., for Lift (and therefore for standardization).

This section's methods are related to *developments*. A development is a rewrite step sequence starting from a term t where each step contracts a redex which represents work that was already in t and "created" redexes are not contracted. Usually, the notions of "work already present" and "created" are defined using *residuals* of redexes across rewrite steps, sometimes defining residuals using marks. This section's methods do not need any notion of residual. This is important because there do not seem to be good ways to define residuals for many rewriting systems, e.g., those with highly overlapping rewrite rules.

A *mark structure* for an AES is a tuple

$$(\mathsf{Marks}, \mathsf{markOf}, \mathsf{noMark}, \mathsf{rename})$$

satisfying axiom 6.1 below and the following conditions. The set Marks is non-empty and does not contain $\star$. The function markOf maps $\mathbb{S}$ to $\mathsf{Marks} \cup \{\star\}$. The mark noMark is a member of Marks. The function rename is of type $(\mathsf{Marks} \times \mathsf{Marks}) \to \mathbb{T} \to \mathbb{T}$. Let m range over Marks. Let $\mathbb{M} = \{\, s \in \mathbb{S} \mid \mathsf{markOf}(s) \neq \mathsf{noMark} \,\}$. Let the statement $\mathsf{markOccurs}(m,t)$ hold iff there exist s and t' such that $t \xrightarrow{s} t'$ and $\mathsf{markOf}(s) = m$.

The intended meaning is as follows. The set Marks should contain marks used to track redexes. Each rewrite step s should be marked by the mark $\mathsf{markOf}(s)$. The special mark noMark means "no mark at all". The symbol $\star$ means "can be considered to be any mark because we do not track this kind of rewrite step with marks"; this is a convenience for systems where only some steps have marked versions. The operation $\mathsf{rename}(m_1, m_2)(t)$ should produce a new term t' resulting from renaming all occurrences of the mark m_1 in t to m_2.

Axiom 6.1 (Marking Sanity)

1. ***Marked Erasure:***
   For $\mathcal{S} \in \{\mathbb{E}, \mathbb{N}\}$,
   $$\begin{tikzcd} t_1 \arrow[r, "\mathcal{S}"] \arrow[d, "\mathsf{rename}(m,m')"'] & t_2 \arrow[d, "\mathsf{rename}(m,m')"] \\ t_3 \arrow[r, dotted, "\mathcal{S}"] & t_4 \end{tikzcd}$$
2. $\mathbb{E}$ ***Marked Unerasure:***
   $$\begin{tikzcd} t_1 \arrow[r, "\mathbb{E}"] & t_2 \\ t_3 \arrow[u, "\mathsf{rename}(m,m')"] \arrow[r, dotted, "\mathbb{E}"] & t_4 \arrow[u, dotted, "\mathsf{rename}(m,m')"'] \end{tikzcd}$$
3. ***Erasing Nonexistent Mark:***
   If $\neg\mathsf{markOccurs}(m, t)$*, then* $\mathsf{rename}(m, m')(t) = t$.
4. ***Marks Not Introduced by Rewriting:***
   If $\neg\mathsf{markOccurs}(m, t)$*,* $m \neq \mathsf{noMark}$*, and* $t \longrightarrow t'$*, then* $\neg\mathsf{markOccurs}(m, t')$.
5. ***Fresh Marks:***
   For any term t*, there exists a mark* $m \neq \mathsf{noMark}$ *such that* $\neg\mathsf{markOccurs}(m, t)$.

□

Convention 6.2 *In this paper, wherever no specific mark structure is being considered, statements are about every possible mark structure.* □

Definition 6.3 (Rewrite Step Set Property for Marks).
$\mathbb{N}$ *Step Can Be Marked:*

$$\mathsf{NM}(\mathcal{S}) \iff \left(\begin{pmatrix} m \neq \mathsf{noMark} \\ \wedge \neg\mathsf{markOccurs}(m, t_1) \end{pmatrix} \Rightarrow \begin{tikzcd} t_1 \arrow[rr, "\mathbb{N},\mathcal{S}"] & & t_2 \\ & t_3 \arrow[ul, dotted, "\mathsf{rename}(m,m')"] \arrow[ur, dotted, "\mathbb{N},\mathbb{M},\mathcal{S}"'] & \end{tikzcd} \right)$$

□

Theorem 6.4 (Lift/Project when Terminating via Marks). *If* $\mathcal{S} \cap \mathbb{M} \subseteq \mathsf{LPT}$ *and* $\mathsf{NM}(\mathcal{S})$*, then* $\mathcal{S} \subseteq \mathsf{LPT}$. □

Proof. Using axiom 6.1, lemma 4.2(1), and definitions 6.3 and 4.1. For full proof details, see appendix D. □

7 Example: The Call-by-Name λ-Calculus

This section gives an example of the use of our AES framework and our diagram-based methods for proving meaning preservation. We choose the call-by-name λ-calculus with left-most outermost evaluation to weak head normal forms because

it is a small system, needs the mark structure features of section 6, and will already be familiar to most readers. The AES and a mark structure will be defined and then the top-level proof strategy will be presented.

Define the AES for the call-by-name λ-calculus as follows. First, define the AES carrier sets $\mathbb{T}$, $\mathbb{S}$, and $\mathbb{R}$ as well as the evaluation step subset $\mathbb{E}$.

$$
\begin{array}{lll}
x, y, z & \in \mathsf{Variable} & \\
\overline{t} & \in \mathsf{Context} & ::= \Box \mid x \mid \lambda x\, \overline{t} \mid \overline{t_1}\, \overline{t_2} \mid \mathsf{let}^n x = \overline{t_2}\, \mathsf{in}\, \overline{t_1} \quad (n \geq 1) \\
t & \in \mathbb{T} & = \{\, \overline{t} \mid \overline{t} \text{ has no hole } \Box \,\} \\
\overline{E} & \in \mathsf{EvalContext} & ::= \Box \mid \overline{E}\, t \\
R & \in \mathsf{Redex} & ::= \mathsf{let}^n x = t_2\, \mathsf{in}\, t_1 \mid (\lambda x\, t_1)\, t_2 \\
s & \in \mathbb{S} & ::= (\overline{t}, R) \\
 & \quad \mathbb{E} & ::= (\overline{E}, R) \\
r & \in \mathbb{R} & = \{\mathsf{diverges}, \mathsf{stuck}, \mathsf{halt}\}
\end{array}
$$

In the term syntax, $(\mathsf{let}^n x = t_2\, \mathsf{in}\, t_1)$ is used to indicate a *marked* β-redex. Terms, contexts, and redexes are identified modulo α-conversion as usual. For contexts, α-conversion can not rename bound variables whose scope includes a hole. Substitution of t for x in t', written $t'[x := t]$, is defined as usual. Placing a term or context X in the hole of a one-hole context $\overline{t}$, written $\overline{t}[X]$, is defined as usual.

Now, finish defining the AES by supplying the functions.

$$
\begin{array}{ll}
\mathsf{endpoints}(\overline{t}, (\mathsf{let}^n x = t_2\, \mathsf{in}\, t_1)) & = (\overline{t}[\mathsf{let}^n x = t_2\, \mathsf{in}\, t_1], \overline{t}[t_1[x := t_2]]) \\
\mathsf{endpoints}(\overline{t}, (\lambda x\, t_1)\, t_2) & = (\overline{t}[(\lambda x\, t_1)\, t_2], \overline{t}[t_1[x := t_2]]) \\
\mathsf{result}(t) & = \begin{cases} \mathsf{diverges} & \text{if } \neg\mathsf{has\text{-}nf}(\mathbb{E}, t) \\ \mathsf{halt} & \text{if } t \xrightarrow{\mathbb{E},\mathsf{nf}}\!\!\!\!\rightarrow \lambda x\, t' \\ \mathsf{stuck} & \text{if } t \xrightarrow{\mathbb{E},\mathsf{nf}}\!\!\!\!\rightarrow t' \neq \lambda x\, t'' \end{cases}
\end{array}
$$

Define an accompanying mark structure as follows.

$$
\begin{array}{ll}
\mathsf{Marks} & = \{0, 1, 2, \ldots\} \\
\mathsf{noMark} & = 0 \\
\mathsf{markOf}(\overline{t}, (\mathsf{let}^n x = t_2\, \mathsf{in}\, t_1)) & = n \\
\mathsf{markOf}(\overline{t}, (\lambda x\, t_1) t_2) & = 0 \\
\mathsf{rename}(m_1, m_2) & = \theta \\
\quad \text{where} & \begin{cases} \theta(x) & = x \\ \theta(\lambda x\, t) & = \lambda x\, \theta(t) \\ \theta(t_1\, t_2) & = \theta(t_1)\, \theta(t_2) \\ \theta(\mathsf{let}^{m_1} x = t_2\, \mathsf{in}\, t_1) & = (\mathsf{let}^{m_2} x = \theta(t_2)\, \mathsf{in}\, \theta(t_1)) \quad \text{if } m_2 \neq 0 \\ \theta(\mathsf{let}^{m_1} x = t_2\, \mathsf{in}\, t_1) & = (\lambda x\, \theta(t_1))\, \theta(t_2) \quad \text{if } m_2 = 0 \\ \theta(\mathsf{let}^{m} x = t_2\, \mathsf{in}\, t_1) & = (\mathsf{let}^{m} x = \theta(t_2)\, \mathsf{in}\, \theta(t_1)) \quad \text{if } m \neq m_1 \end{cases}
\end{array}
$$

Lemma 7.1 (The Framework User's Proof Burden).

1. *Axioms 3.3, 6.1, and 3.4 hold.*
2. $\mathsf{WB{+}Std}(\mathbb{M})$.
3. $\mathsf{NM}(\mathbb{S})$.
4. *If $t_1 \longrightarrow t_2$, then $\overline{t}[t_1] \longrightarrow \overline{t}[t_2]$ for any context $\overline{t}$.* $\Box$

Proof. Many standard proofs by induction which are left to the reader. The only difficult bit is $\mathsf{BE}(\mathbb{M})$ (part of $\mathsf{WB}{+}\mathsf{Std}(\mathbb{M})$). The stronger statement $\mathsf{SN}(\mathbb{M})$ is proven by a known argument (e.g., see [3]) of rearranging the mark values so that rewriting decreases the multiset of all marks in the term in the multiset extension of $<$. □

Theorem 7.2 (Meaning Preservation). $\mathbb{S} \subseteq \mathsf{MP}$. □

Proof. Everything implicitly relies on lemma 7.1(1). By lemma 7.1(2) and theorem 5.4(1), $\mathsf{SLift}(\mathbb{M})$ and $\mathsf{SProj}(\mathbb{M})$. By lemma 4.2(2,3,5,7,9), $\mathbb{S} \cap \mathbb{M} = \mathbb{M} \subseteq \mathsf{LPT}$. By lemma 7.1(3) and theorem 6.4, $\mathbb{S} \subseteq \mathsf{LPT}$. By theorem 4.3(1), $\mathbb{S} \subseteq \mathsf{MP}$. □

Corollary 7.3 (Observational Equivalence). *If* $t_1 \longrightarrow t_2$, *then* $\mathsf{result}(\overline{t}[t_1]) = \mathsf{result}(\overline{t}[t_2])$. □

Proof. Suppose $t_1 \longrightarrow t_2$. By lemma 7.1(4), $\overline{t}[t_1] \longrightarrow \overline{t}[t_2]$. By theorem 7.2 and the definition of MP, $\mathsf{result}(\overline{t}[t_1]) = \mathsf{result}(\overline{t}[t_2])$. □

8 Related Work

The most closely related work is that by Machkasova and Turbak [10, 11, 12]. Their work is discussed throughout this paper, so only a few points will be made here. First, our BE property corresponds to their complicated notion of γ-development [12, sec. 4.5]. The γ-development idea may be implicitly the same as BE [12, p. 193], but the exact relationship is unclear due to the complexity. Second, Machkasova's requirement of γ-confluence on evaluation is incomparable with our requirement of evaluation subcommutativity (axiom 3.3). Because γ-confluence involves the complicated γ-development machinery, we prefer our simpler requirement. Third, our proof diagrams for lemmas C.1, C.3, C.4, C.5, and C.6 are similar to some in [12] but were developed completely independently. Our proof diagrams are simpler because we do not use γ-developments and we have completely separated the treatment of marks for developments (section 6).

The next most closely related work is an unpublished technical report by Odersky [17] that presents conditions for proving that a proposed transformation $\sim$ is an observational equivalence. This assumes $\sim$ is contextually closed. One of Odersky's conditions is that $\sim$ is *locally stable*, defined by diagram (2) in [17, p. 2]:

$$\begin{array}{ccc} t_1 & \longrightarrow & t_2 \\ {\scriptstyle \sim_1}\Big| & \simeq & \Big|{\scriptstyle \sim_1} \\ t_3 & \cdots & t_4 \end{array}$$

The relation $\sim_1$ is *parallel similarity*, the use of $\sim$ simultaneously at many different (presumably non-overlapping) positions. Another of Odersky's conditions is that $\sim$ *preserves answers*, i.e., $t_1 \sim t_2 \Rightarrow (\mathsf{is\text{-}nf}(\mathbb{E}, t_2) \Rightarrow t_1 \twoheadrightarrow t_2)$. There are additional definitions and conditions that we do not have space to discuss.

Odersky's approach is related to ours as follows. Where Odersky uses $\longrightarrow$ (normal rewriting) and $\simeq$ (observational equivalence), we would use $\xrightarrow{\mathbb{E}}$ and $\xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow$.

Odersky's gives two versions of his approach. In the one in the diagram he takes convertibility with a set of answers ($\mathbb{E}$-normal forms in our setting) in the entire rewriting system to be the definition of meaning preservation and then asks whether additional rewrite rules can be safely added. In this case, the diagram must be proven for all rewrite steps. In the other, he takes an evaluation strategy like we do. In this case, using $\simeq$ on the bottom edge seems more general, but it also seems that in practice this diagram edge would be completed with $\mathbb{E}$ steps. Where Odersky uses $\sim_1$, we would typically use $\xrightarrow{\mathcal{S},\mathbb{M}}\!\!\!\!\rightarrow$ and a combination of one of the well-behavedness conditions of section 5 and the marks for developments of section 6. Odersky's use of parallel (simultaneous) rewriting corresponds to our use of a termination property.

The key distinctions between Odersky's approach and ours are as follows. Many details of Odersky's approach are entangled in syntactic notions for extensions of the λ-calculus while our approach is abstract. Odersky does not provide elementary diagrams where each given edge is a single use of a rewrite rule; it seems that one must work with full parallel similarity. Odersky appears to assume that standardization is already proven while our approach proves whatever standardization is needed and can also work without it. Odersky's approach requires a notion of "preserving evaluation contexts" which we do not fully understand but which we are fairly sure one of our intended applications does not satisfy. Odersky can not distinguish terms that go wrong from those that either diverge or halt normally; thus his framework does not verify that rewriting can not switch between non-wrong and wrong.

9 Future Work

The generalizations of our AES framework and lift/project diagrams were developed to handle $\lambda^{:=,\mathsf{letrec}}$, a calculus we are developing for reasoning about call-by-value higher-order programs with mutable reference cells and mutually recursive definitions (i.e., letrec). Evaluation of assignment statements can introduce cycles in the store, so evaluation results may need letrec even if the initial program was letrec-free. A specific evaluation strategy is given for the $\lambda^{:=,\mathsf{letrec}}$ calculus to define the meaning of programs. Calculi for assignments have been done before (e.g., [7]), but $\lambda^{:=,\mathsf{letrec}}$ also includes improvements like very simple evaluation contexts as well as rules for letrec in the style of the work of Ariola and Blom ([1]). The only previously known methods for reasoning about the correctness of Ariola/Blom style letrec rules use very difficult methods.

The development of $\lambda^{:=,\mathsf{letrec}}$ is nearing completion. Because $\lambda^{:=,\mathsf{letrec}}$ is non-confluent (due to using rules for letrec that Ariola and Klop [3] proved non-confluent), we were using the lift & project method to prove meaning preservation. It does not have finite developments, but has a number of rule subsets whose associated rewrite step sets satisfy the BE property. The last barrier to completing the proof of meaning preservation was several critical pairs of a rule named [lift] (name unrelated to the Lift diagram from definition 4.1). One par-

ticularly irritating critical pair is only completable as follows:

$$\begin{tikzcd}
t_1 \arrow[r, "\mathbb{E},[\mathrm{lift}]"] \arrow[dd, "\mathbb{N},[\mathrm{lift}]"'] & t_3 \arrow[d, dotted, "\mathbb{E},[\mathrm{lift}]"] \\
 & t_5 \\
t_2 \arrow[r, "\mathbb{E},[\mathrm{lift}]"] & t_4 \arrow[u, dotted, "\mathbb{N},[\mathrm{lift}]"']
\end{tikzcd}$$

Unfortunately, this breaks standardization, so the lift & project proof method can not be used. We considered changing the definition of $\lambda^{:=,\mathsf{letrec}}$ (which would be for probably the 1000th time), but felt that the changes to "fix" this critical pair would probably break something else. Also, the rules of $\lambda^{:=,\mathsf{letrec}}$ are *obviously* meaning preserving, so we felt that rather than forcing $\lambda^{:=,\mathsf{letrec}}$ through awkward contortions to fit some weak proof method, it was the proof method that should be fixed. Fortunately, the $\mathsf{WB}\backslash\mathsf{Std}$ property can be proven for the steps of the [lift] rule, so we expect to complete the development of $\lambda^{:=,\mathsf{letrec}}$ soon.

After $\lambda^{:=,\mathsf{letrec}}$ is completed, we want to apply our proof methods to equational reasoning for assembly language and maybe also to explicit substitutions.

References

[1] Z. M. Ariola, S. Blom. Skew confluence and the lambda calculus with letrec. *Ann. Pure Appl. Logic*, 117(1–3), 2002.

[2] Z. M. Ariola, M. Felleisen. The call-by-need lambda calculus. *J. Funct. Programming*, 3(7), 1997.

[3] Z. M. Ariola, J. W. Klop. Lambda calculus with explicit recursion. *Inform. & Comput.*, 139, 1997.

[4] *Programming Languages & Systems, 9th European Symp. Programming*, vol. 1782 of *LNCS*. Springer-Verlag, 2000.

[5] M. Felleisen, D. Friedman. Control operators, the SECD-machine, and the λ-calculus. In M. Wirsing, ed., *Formal Description of Programming Concepts — III*. North-Holland, 1986.

[6] M. Felleisen, D. P. Friedman. A syntactic theory of sequential state. *Theoret. Comput. Sci.*, 69(3), 1989.

[7] M. Felleisen, R. Hieb. The revised report on the syntactic theories of sequential control and state. *Theoret. Comput. Sci.*, 102, 1992.

[8] K. Fisher, J. Reppy, J. G. Riecke. A calculus for compiling and linking classes. In ESOP '00 [4].

[9] G. Gonthier, J.-J. Lévy, P.-A. Melliès. An abstract standardisation theorem. In *Proc. 7th Ann. IEEE Symp. Logic in Comput. Sci.*, 1992.

[10] E. Machkasova. Techniques for proving observational equivalence. ASCII notes that later turned into [11]. Not sure about year, 1998.

[11] E. Machkasova, F. A. Turbak. A calculus for link-time compilation. In ESOP '00 [4].

[12] E. L. Machkasova. *Computational Soundness of Non-Confluent Calculi with Applications to Modules and Linking*. PhD thesis, Boston Univ., 2002.

[13] J. Maraist, M. Odersky, P. Wadler. The call-by-need lambda calculus. *J. Funct. Programming*, 8(3), 1998.

[14] P.-A. Melliès. Axiomatic Rewriting Theory IV: A diagrammatic standardization theorem. Submitted, 2001.

[15] R. Muller. M-LISP: A representation-independent dialect of LISP with reduction semantics. *ACM Trans. on Prog. Langs. & Systs.*, 14(4), 1992.

[16] M. H. A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Math.*, 43(2), 1942.

[17] M. Odersky. A syntactic method for proving observational equivalences. Research Report YALEU/DCS/RR-964, Yale Univ., Dept. of Comp. Science, 1993.

[18] G. D. Plotkin. Call-by-name, call-by-value and the lambda calculus. *Theoret. Comput. Sci.*, 1, 1975.

[19] W. Taha. A sound reduction semantics for untyped CBN multi-stage computation: Or, the theory of MetaML is non-trivial. In *Proceedings of the 2000 ACM SIGPLAN Workshop on Evaluation and Semantics-Based Program Manipulation (PEPM-00)*, N.Y., 2000. ACM Press.

A Proofs for Section 3

Lemma A.1 (Multi-Step Subcommutativity).
$\mathsf{SubComm}(\mathcal{S}) \Leftrightarrow \forall i, j \geq 0.\ \mathsf{SubComm}(\mathcal{S}, i, j)$. □

Proof. The $\Leftarrow$ direction is immediate, so $\mathsf{SubComm}(\mathcal{S}) \Rightarrow \mathsf{SubComm}(\mathcal{S}, i, j)$ is proven here for $i, j \geq 0$ by induction on $i \times j$. Suppose $\mathsf{SubComm}(\mathcal{S})$ and $t_2 \xleftarrow{\mathcal{S},i} t_1 \xrightarrow{\mathcal{S},j} t_3$ and it will be proven that there exists t_4 such that $t_2 \xrightarrow{\mathcal{S},\leq j} t_4 \xleftarrow{\mathcal{S},\leq i} t_3$. By cases on i and j.

1. Suppose $i = 0$ or $j = 0$. W/o.l.o.g., consider only the case $i = 0$. So $t_1 = t_2$. Let $t_4 = t_3$. Done.
2. Suppose $i = 1$ and $j = 1$. By $\mathsf{SubComm}(\mathcal{S})$.
3. Suppose $i > 1$ or $j > 1$. W/o.l.o.g., consider only the case $j > 1$. By diagram:

$$\begin{tikzcd}
t_1 \arrow[r, "\mathcal{S}"] \arrow[d, "\mathcal{S},i"'] \arrow[dr, phantom, "\text{IH}"] & t_5 \arrow[r, "\mathcal{S},j-1"] \arrow[d, dotted, "\mathcal{S},i',\leq i"] \arrow[dr, phantom, "\text{IH}"] & t_3 \arrow[d, dotted, "\mathcal{S},\leq i'"] \\
t_2 \arrow[r, dotted, "\mathcal{S},\leq 1"'] & t_6 \arrow[r, dotted, "\mathcal{S},\leq j-1"'] & t_4
\end{tikzcd}$$

□

Lemma A.2 (Multi-Step Evaluation Subcommutativity).
$\mathsf{SubComm}(\mathbb{E}, i, j)$ *for all* $i, j \geq 0$. □

Proof. By axiom 3.3 and lemma A.1. □

Lemma A.3 (Confluence and Unique Normal Forms for Evaluation).

1. $\mathsf{Conf}(\mathbb{E})$.
2. *If* $t_1 \xleftarrow{\mathbb{E},\mathsf{nf}} t \xrightarrow{\mathbb{E},\mathsf{nf}} t_1$, *then* $t_1 = t_2$. □

Proof.

1. Immediate from lemma A.2.
2. Suppose $t_1 \xleftarrow{\mathbb{E},\mathsf{nf}} t \xrightarrow{\mathbb{E},\mathsf{nf}} t_2$. By lemma A.3(1), there exists t_3 such that $t_1 \xrightarrow{\mathbb{E}} t_3 \xleftarrow{\mathbb{E}} t_2$. Because $\mathsf{is\text{-}nf}(\mathbb{E}, t_1)$ and $\mathsf{is\text{-}nf}(\mathbb{E}, t_2)$, it holds that $t_1 = t_3 = t_2$. Done. (This is a standard result.) □

Proof of Lemma 3.5 (Non-Evaluation Steps on Eval-Normal Forms). First, it is proven that $t \xleftrightarrow{\mathbb{N},s} t'$ and $\mathsf{is\text{-}nf}(\mathbb{E}, t)$ imply $t \xleftrightarrow{\mathsf{MP},s} t'$ and $\mathsf{is\text{-}nf}(\mathbb{E}, t')$. Suppose $t \xleftrightarrow{\mathbb{N},s} t'$ and $\mathsf{is\text{-}nf}(\mathbb{E}, t)$. By Axiom 3.4(3), it holds that $\mathsf{is\text{-}nf}(\mathbb{E}, t')$. By Axiom 3.4(4), it holds that $\mathsf{result}(t) = \mathsf{result}(t')$. Hence, $s \in \mathsf{MP}$ and thus $t \xleftrightarrow{\mathsf{MP},s} t'$. The lemma claim then follows by induction on k such that $t_1 \xleftrightarrow{\mathbb{N},k} t_2$. □

B Proofs for Section 4

Proof of Lemma 4.2 (Relationships between Lift and Project Properties).

1. Suppose $s \in \mathbb{E}$. It will now be proven that $s \in \mathsf{Lift}$ and $s \in \mathsf{Proj}$.
   Suppose $t_1 \xrightarrow{s} t_2 \xrightarrow{\mathbb{E}} t_3$. Then for $t_4 = t_3$ it holds that $t_1 \xrightarrow{\mathbb{E}} t_4 \xrightarrow{\mathbb{N}} t_3$. Thus, $s \in \mathsf{Lift}$.
   Suppose $t_3 \xleftarrow{s} t_1 \xrightarrow{\mathbb{E}} t_2$. By lemma A.2, there exists t_4 such that $t_3 \xrightarrow{\mathbb{E}} t_4 \xleftarrow{\mathbb{E}} t_2$. Then for $t_5 = t_4$ it holds that $t_3 \xrightarrow{\mathbb{E}} t_4 \xleftarrow{\mathbb{N}} t_5 \xleftarrow{\mathbb{E}} t_2$. Thus, $s \in \mathsf{Proj}$.
2. Suppose $\mathsf{SLift}(\mathcal{S})$ and $s \in \mathcal{S}$. If $s \in \mathbb{E}$, then $s \in \mathsf{Lift}$ by lemma 4.2(1). Suppose $s \in \mathbb{N}$. Suppose $t_1 \xrightarrow{s} t_2 \xrightarrow{\mathbb{E}} t_3$. By $\mathsf{SLift}(\mathcal{S})$ and $s \in \mathcal{S}$, there exists t_4 such that $t_1 \xrightarrow{\mathbb{E}} t_4 \xrightarrow{\mathbb{N},\mathcal{S}} t_3$. This implies $t_1 \xrightarrow{\mathbb{E}} t_4 \xrightarrow{\mathbb{N}} t_3$. Thus, $s \in \mathsf{Lift}$.
3. Similar to the proof of lemma 4.2(2).
4. Similar to the proof of lemma 4.2(2).
5. Suppose $s \in \mathsf{Lift}$. Suppose $t_1 \xrightarrow{s} t_2 \xrightarrow{\mathbb{E}} t_3$. Because $s \in \mathsf{Lift}$, there exists t_4 such that $t_1 \xrightarrow{\mathbb{E}} t_4 \xrightarrow{\mathbb{N}} t_3$. Then for $t_5 = t_3$ it holds that $t_1 \xrightarrow{\mathbb{E}} t_4 \xrightarrow{\mathbb{N}} t_5 \xleftarrow{\mathbb{E}} t_3$. Thus, $s \in \mathsf{Lift}'$.
6. Consider this 5-term 5-step AES where all results are the same:

$$\begin{tikzcd}
t_1 \arrow[r, "\mathbb{E},s_1"'] \arrow[d, "\mathbb{N},s_2"'] & t_2 \arrow[dr, "\mathbb{N},s_3"] & \\
t_3 \arrow[r, "\mathbb{E},s_4"'] & t_4 \arrow[r, "\mathbb{E},s_5"'] & t_5
\end{tikzcd}$$

   Then $\mathsf{Lift}' = \mathbb{S}$ and $\mathsf{Lift}' \setminus \mathsf{Lift} = \{s_2\}$.
7. Suppose $s \in \mathsf{Lift}' \cap \mathsf{Proj}$. Suppose $t_2 \xleftrightarrow{s} t_1 \xrightarrow{\mathbb{E}} t_3$. By cases on how $t_1 \xleftrightarrow{s} t_2$ is true.
   (a) Suppose $t_1 \xrightarrow{s} t_2$. By $s \in \mathsf{Proj}$ there exist t_4 and t_5 such that $t_2 \xrightarrow{\mathbb{E}} t_4 \xleftarrow{\mathbb{N}} t_5 \xleftarrow{\mathbb{E}} t_3$. So $t_4 \xleftrightarrow{\mathbb{N}} t_5$. Thus, $s \in \mathsf{LP}$.
   (b) Suppose $t_1 \xleftarrow{s} t_2$. By $s \in \mathsf{Lift}'$ there exist t_4 and t_5 such that $t_2 \xrightarrow{\mathbb{E}} t_4 \xrightarrow{\mathbb{N}} t_5 \xleftarrow{\mathbb{E}} t_3$. So $t_4 \xleftrightarrow{\mathbb{N}} t_5$. Thus, $s \in \mathsf{LP}$.
8. Consider this 4-term 4-step AES where all results are the same:

$$\begin{tikzcd}
t_1 \arrow[r, "\mathbb{E},s_1"'] \arrow[d, "\mathbb{N},s_2"'] & t_2 \\
t_3 \arrow[r, "\mathbb{E},s_4"'] & t_4 \arrow[u, "\mathbb{N},s_3"']
\end{tikzcd}$$

   Then $\mathsf{LP} = \mathbb{S}$ and $\mathsf{LP} \setminus \mathsf{Lift}' = \mathsf{LP} \setminus \mathsf{Proj} = \{s_2\}$.
9. Suppose $s \in \mathsf{LP}$. Suppose $t_2 \xleftrightarrow{s} t_1 \xrightarrow{\mathbb{E},\mathsf{nf}} t_3$. By $s \in \mathsf{LP}$ there exist t_4 and t_5 such that $t_2 \xrightarrow{\mathbb{E}} t_4 \xleftrightarrow{\mathbb{N}} t_5 \xleftarrow{\mathbb{E}} t_3$. Because $\mathsf{is\text{-}nf}(\mathbb{E}, t_3)$, it holds that $t_3 = t_5$. So $t_2 \xrightarrow{\mathbb{E}} t_4 \xleftrightarrow{\mathbb{N}} t_3$. Thus, $s \in \mathsf{LPT}$.
10. Consider this 3-term 4-step AES:

$$\begin{tikzcd}
t_1 \arrow[loop right, "\mathbb{E},s_3"] \arrow[d, "\mathbb{N},s_1"'] & & \\
t_2 \arrow[r, "\mathbb{E},s_2"'] & t_3 \arrow[loop right, "\mathbb{E},s_4"] &
\end{tikzcd}$$

   Then $\mathsf{LPT} = \mathbb{S}$, and $\mathsf{LPT} \setminus \mathsf{LP} = \{s_1\}$. $\square$

Proof of Lemma 4.4 (Lift Equivalent to Standardization) part 1.

(⇒): Suppose $\mathcal{S} \subseteq \mathsf{Lift}$ and $\mathsf{Std}(\mathcal{S} \cup \mathbb{E})$ will be proven. Suppose $t_1 \xrightarrow{\mathcal{S} \cup \mathbb{E}}\!\!\!\!\rightarrow t_2$ and it will be proven that there exists t_3 such that $t_1 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_3 \xrightarrow{\mathbb{N}}\!\!\!\!\rightarrow t_2$. By induction on k such that $t_1 \xrightarrow{\mathcal{S} \cup \mathbb{E}, k} t_2$. By cases on k.

1. Case $k = 0$: Let $t_3 = t_1 = t_2$. Then $t_1 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_3 \xrightarrow{\mathbb{N}}\!\!\!\!\rightarrow t_2$. Done.
2. Case $k \geq 1$: By lemma 4.2(1), $\mathbb{E} \subseteq \mathsf{Lift}$, so $\mathcal{S} \cup \mathbb{E} \subseteq \mathsf{Lift}$. By diagram:

$$\begin{tikzcd}
t_1 \arrow[r, "\mathcal{S} \cup \mathbb{E}"] \arrow[d, two heads, dotted, "\mathbb{E}"'] \arrow[dr, phantom, "\mathcal{S} \cup \mathbb{E} \subseteq \mathsf{Lift}"] & t' \arrow[r, "\mathcal{S} \cup \mathbb{E}, k-1"] \arrow[d, two heads, dotted, "\mathbb{E}"] \arrow[dr, phantom, "\mathrm{IH}"] & t_2 \\
t_3 \arrow[r, two heads, dotted, "\mathbb{N}"] & t'' \arrow[ur, two heads, dotted, "\mathbb{N}"'] &
\end{tikzcd}$$

(⇐): Suppose $\mathsf{Std}(\mathcal{S} \cup \mathbb{E})$. Suppose $t_1 \xrightarrow{\mathcal{S}} t_2 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_3$. Thus, $t_1 \xrightarrow{\mathcal{S} \cup \mathbb{E}}\!\!\!\!\rightarrow t_3$. By $\mathsf{Std}(\mathcal{S} \cup \mathbb{E})$, there exists t_4 such that $t_1 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_4 \xrightarrow{\mathbb{N}}\!\!\!\!\rightarrow t_3$. Thus, $\mathcal{S} \subseteq \mathsf{Lift}$.

□

C Proofs for Section 5

Lemma C.1 (Weak Lift Many-to-1). *If* $\mathsf{WL1}(\mathcal{S}, \mathcal{S}')$*, then:*

$$\begin{tikzcd} t_1 \arrow[r, dotted, "\mathbb{E},\mathcal{S}'"] \arrow[d, two heads, "\mathbb{N},\mathcal{S}"'] & t_4 \arrow[d, dotted, two heads, "\mathcal{S}"] \\ t_2 \arrow[r, "\mathbb{E},\mathcal{S}'"] & t_3 \end{tikzcd}$$

□

Proof. By induction on k such that $t_1 \xrightarrow{\mathbb{N},\mathcal{S},k} t_2$. By cases on k.

- Case $k = 0$. Then $t_1 = t_2$. Let $t_4 = t_3$. Done.
- Case $k > 0$. By diagram:

$$\begin{tikzcd} t_1 \arrow[r, dotted, "\mathbb{E},\mathcal{S}'"] \arrow[d, two heads, "\mathbb{N},\mathcal{S}"'] \arrow[dr, phantom, "\mathsf{WL1}(\mathcal{S},\mathcal{S}')"] & t_4 \arrow[d, dotted, two heads, "\mathcal{S}"] \\ t \arrow[r, dotted, "\mathbb{E},\mathcal{S}'"] \arrow[d, two heads, "\mathbb{N},\mathcal{S}"'] \arrow[dr, phantom, "\text{IH}"] & t' \arrow[d, dotted, two heads, "\mathcal{S}"] \\ t_2 \arrow[r, "\mathbb{E},\mathcal{S}'"] & t_3 \end{tikzcd}$$

□

Lemma C.2 ($\mathcal{S}$-Standardization). *If* $\mathsf{WL1}(\mathcal{S}, \mathcal{S})$ *and* $\mathsf{BE}(\mathcal{S})$*, then* $\mathsf{Std}(\mathcal{S}, \mathcal{S})$*.*

□

Proof. Let $\mathsf{evalStepBound}(t, \mathcal{S}) = \mathsf{max}\{\, k \mid \exists t'.\ t \xrightarrow{\mathsf{N}^*\mathsf{EN}^*(\mathcal{S}),k} t' \,\}$. Note that $\mathsf{evalStepBound}(t, \mathcal{S})$ is only defined when a bound exists. If $\mathsf{BE}(\mathcal{S})$, then $\mathsf{evalStepBound}(t, \mathcal{S})$ is defined for every term t.

Suppose $t_1 \overset{\mathcal{S}}{\twoheadrightarrow} t_2$ and it will be proven that there exists t_3 such that $t_1 \overset{\mathbb{E},\mathcal{S}}{\twoheadrightarrow} t_3 \overset{\mathbb{N},\mathcal{S}}{\twoheadrightarrow} t_2$. Let $k = \mathsf{evalStepBound}(t_1, \mathcal{S})$ (this is well defined because $\mathsf{BE}(\mathcal{S})$). By induction on k. By cases on whether $t_1 \overset{\mathbb{N},\mathcal{S}}{\twoheadrightarrow} t_2$.

1. Suppose $t_1 \overset{\mathbb{N},\mathcal{S}}{\twoheadrightarrow} t_2$. Then let $t_3 = t_1$. Done.
2. Suppose $t_1 \overset{\mathbb{N},\mathcal{S}}{\twoheadrightarrow} t_2$ does not hold. The fact that $t_1 \overset{\mathcal{S}}{\twoheadrightarrow} t_2$ holds must depend on using at least one $\mathbb{E}$ step. By diagram, observing that $\mathsf{evalStepBound}(t_6, \mathcal{S}) < k$:

$$\begin{tikzcd} t_1 \arrow[r, dotted, "\mathbb{E},\mathcal{S}"] \arrow[d, two heads, "\mathbb{N},\mathcal{S}"'] \arrow[dr, phantom, "\mathsf{WL1}(\mathcal{S},\mathcal{S}),\ \text{lem. C.1}"] & t_6 \arrow[r, dotted, two heads, "\mathbb{E},\mathcal{S}"] \arrow[d, dotted, two heads, "\mathcal{S}"] \arrow[dr, phantom, "\text{IH}"] & t_3 \arrow[d, dotted, two heads, "\mathbb{N},\mathcal{S}"] \\ t_4 \arrow[r, "\mathbb{E},\mathcal{S}"] & t_5 \arrow[r, two heads, "\mathcal{S}"] & t_2 \end{tikzcd}$$

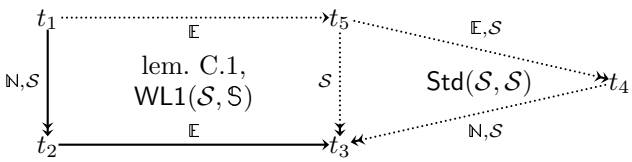
□

Lemma C.3 (Strong Lift Many-to-1). *If* $\mathsf{WL1}(\mathcal{S}, \mathbb{S})$ *and* $\mathsf{Std}(\mathcal{S}, \mathcal{S})$*, then*

$$\begin{tikzcd} t_1 \arrow[r, dotted, "\mathbb{E},\geq 1"] \arrow[d, two heads, "\mathbb{N},\mathcal{S}"'] & t_4 \arrow[d, dotted, two heads, "\mathbb{N},\mathcal{S}"] \\ t_2 \arrow[r, "\mathbb{E}"] & t_3 \end{tikzcd}$$

□

Proof. By diagram:

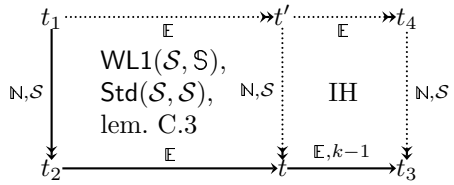


□

Lemma C.4 (Strong Lift). *If* $\mathsf{WL1}(\mathcal{S},\mathbb{S})$ *and* $\mathsf{Std}(\mathcal{S},\mathcal{S})$*, then* $\mathsf{SLift}(\mathcal{S})$. □

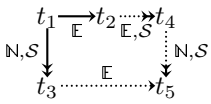
Proof. Suppose $t_1 \xrightarrow{\mathbb{N},\mathcal{S}}\!\!\!\!\rightarrow t_2 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_3$ and prove there exists t_4 such that $t_1 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_4 \xrightarrow{\mathbb{N},\mathcal{S}}\!\!\!\!\rightarrow t_3$. By induction on k such that $t_2 \xrightarrow{\mathbb{E},k} t_3$. By cases on k.

- Suppose $k = 0$. Let $t_4 = t_1$. Done.
- Suppose $k \geq 1$. By diagram:



□

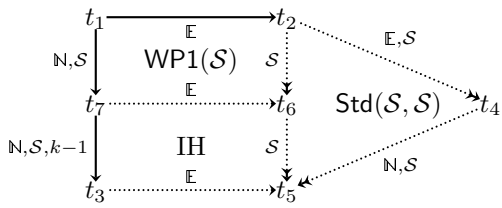
Lemma C.5 (Strong Project Many-to-1). *If* $\mathsf{WP1}(\mathcal{S})$ *and* $\mathsf{Std}(\mathcal{S},\mathcal{S})$*, then*



□

Proof. By induction on k where $t_1 \xrightarrow{\mathbb{N},\mathcal{S},k} t_3$. By cases on k.

1. Suppose $k = 0$. Then $t_1 = t_3$. Let $t_2 = t_4 = t_5$. Done.
2. Suppose $k \geq 1$. By diagram:



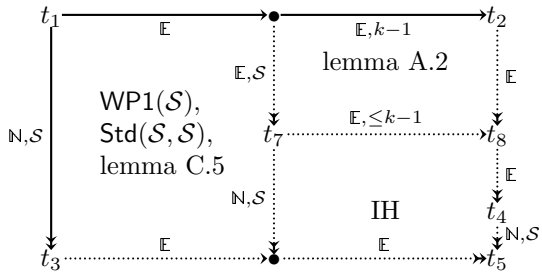
□

Lemma C.6 (Strong Project). *If* $\mathsf{WP1}(\mathcal{S})$ *and* $\mathsf{Std}(\mathcal{S},\mathcal{S})$*, then* $\mathsf{SProj}(\mathcal{S})$. □

Proof. Suppose $t_3 \xleftarrow{\mathbb{N},\mathcal{S}}\!\!\!\!\leftarrow t_1 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_2$ and prove there exist t_4 and t_5 such that $t_3 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_5 \xleftarrow{\mathbb{N},\mathcal{S}}\!\!\!\!\leftarrow t_4 \xleftarrow{\mathbb{E}}\!\!\!\!\leftarrow t_2$. By induction on k where $t_1 \xrightarrow{\mathbb{E},k} t_2$. By cases on k.

1. Suppose $k = 0$. Then $t_2 = t_1$. Let $t_4 = t_2$ and let $t_5 = t_3$. Thus, $t_3 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_5 \xleftarrow{\mathbb{N},\mathcal{S}}\!\!\!\!\leftarrow t_4 \xleftarrow{\mathbb{E}}\!\!\!\!\leftarrow t_2$. Done.

2. Suppose $k \geq 1$. By diagram:



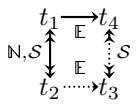
□

Lemma C.7 (Local Confluence & SN Imply Confluence (Newman [16])). *If* $\mathsf{LConf}(\mathcal{S})$ *and* $\mathsf{SN}(\mathcal{S})$*, then* $\mathsf{Conf}(\mathcal{S})$*.* □

Lemma C.8 ($\mathcal{S}$-Standardization to Normal Form). *If* $\mathsf{Conf}(\mathcal{S})$*,* $\mathsf{WN}(\mathcal{S})$*, and* $\mathsf{NE}(\mathcal{S})$*, then* $\mathsf{Std\text{-}nf}(\mathcal{S})$*.* □

Proof. Suppose $t_1 \xrightarrow{\mathcal{S},\mathsf{nf}}\!\!\!\!\rightarrow t_2$. Because $\mathsf{WN}(\mathcal{S})$, it holds that $\mathsf{WN}(\mathbb{E} \cap \mathcal{S})$. So there exists t_3 such that $t_1 \xrightarrow{\mathbb{E},\mathcal{S},\mathsf{nf}}\!\!\!\!\rightarrow t_3$. Because $t_3 \xleftarrow{\mathcal{S}}\!\!\!\!\leftarrow t_2$, by $\mathsf{Conf}(\mathcal{S})$ there exists t_4 such that $t_3 \xrightarrow{\mathcal{S}}\!\!\!\!\rightarrow t_4 \xleftarrow{\mathcal{S}}\!\!\!\!\leftarrow t_2$. Because $\mathsf{is\text{-}nf}(\mathcal{S}, t_2)$, it holds that $t_4 = t_2$. So $t_3 \xrightarrow{\mathcal{S},\mathsf{nf}}\!\!\!\!\rightarrow t_2$. Because $\mathsf{is\text{-}nf}(\mathbb{E} \cap \mathcal{S}, t_3)$ and $\mathsf{is\text{-}nf}(\mathcal{S}, t_2)$, by induction on k such that $t_3 \xrightarrow{\mathcal{S},k}\!\!\!\!\rightarrow t_2$ and using $\mathsf{NE}(\mathcal{S})$, it holds that $t_3 \xrightarrow{\mathbb{N},\mathcal{S},\mathsf{nf}}\!\!\!\!\rightarrow t_2$. Thus, there exists t_3 such that $t_1 \xrightarrow{\mathbb{E},\mathcal{S},\mathsf{nf}}\!\!\!\!\rightarrow t_3 \xrightarrow{\mathbb{N},\mathcal{S},\mathsf{nf}}\!\!\!\!\rightarrow t_2$. □

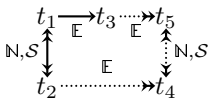
Lemma C.9 (Weak Lift/Project Many-to-1). *If* $\mathsf{WLP1}(\mathcal{S})$*, then:*



□

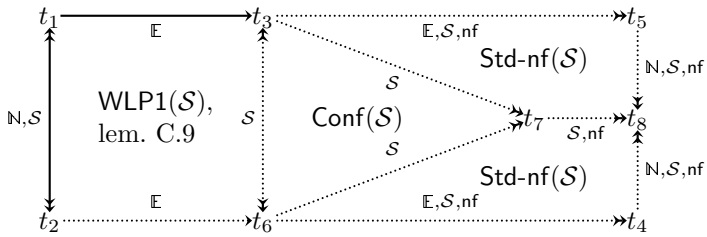
Proof. By induction using a proof resembling the proofs of lemmas C.1 and C.4. □

Lemma C.10 (Strong Lift/Project Many-to-1). *If* $\mathsf{Conf}(\mathcal{S})$*,* $\mathsf{WN}(\mathcal{S})$*,* $\mathsf{Std\text{-}nf}(\mathcal{S})$*, and* $\mathsf{WLP1}(\mathcal{S})$*, then:*



□

Proof. By the following diagram, where $t_7 \xrightarrow{\mathcal{S},\mathsf{nf}}\!\!\!\!\rightarrow t_8$ because $\mathsf{WN}(\mathcal{S})$:



□

Lemma C.11 (Strong Lift/Project). *If* $\mathsf{Conf}(\mathcal{S})$*,* $\mathsf{WN}(\mathcal{S})$*,* $\mathsf{Std\text{-}nf}(\mathcal{S})$*, and* $\mathsf{WLP1}(\mathcal{S})$*, then* $\mathsf{SLP}(\mathcal{S})$*.* □

Proof. By induction using lemma C.10. The proof is nearly the same as that for lemma C.6. □

D Proofs for Section 6

Lemma D.1 (Marking Properties).

1.

$$\begin{tikzcd} t_1 \arrow[r, two heads, "\mathbb{E}"] \arrow[d, "\mathsf{rename}(m,m')"'] & t_2 \arrow[d, "\mathsf{rename}(m,m')"] \\ t_3 \arrow[r, dotted, two heads, "\mathbb{E}"] & t_4 \end{tikzcd}$$

2.

$$\begin{tikzcd} t_1 \arrow[r, leftrightarrow, two heads, "\mathbb{N}"] \arrow[d, "\mathsf{rename}(m,m')"'] & t_2 \arrow[d, "\mathsf{rename}(m,m')"] \\ t_3 \arrow[r, dotted, leftrightarrow, two heads, "\mathbb{N}"] & t_4 \end{tikzcd}$$

3.

$$\begin{tikzcd} t_1 \arrow[r, two heads, "\mathbb{E}"] & t_2 \\ t_3 \arrow[u, "\mathsf{rename}(m,m')"] \arrow[r, dotted, two heads, "\mathbb{E}"] & t_4 \arrow[u, dotted, "\mathsf{rename}(m,m')"'] \end{tikzcd}$$

4. If $\neg\mathsf{markOccurs}(m,t)$*,* $m \neq \mathsf{noMark}$*, and* $t \twoheadrightarrow t'$*, then* $\neg\mathsf{markOccurs}(m,t')$*.*

5. If $\mathsf{rename}(m,m')(t_1) = t_2$*, then* $\mathsf{is\text{-}nf}(\mathbb{E},t_1)$ *iff* $\mathsf{is\text{-}nf}(\mathbb{E},t_2)$*.* □

Proof. Using axiom 6.1, by induction on the length of rewrite sequences where needed. □

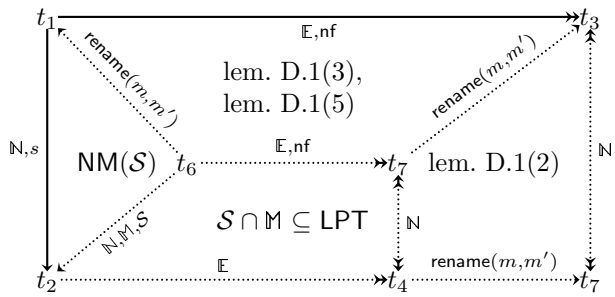
Proof of Theorem 6.4 (Lift/Project when Terminating via Marks). Suppose that $\mathcal{S}\cap\mathbb{M} \subseteq \mathsf{LPT}$ and $\mathsf{NM}(\mathcal{S})$ and $\mathcal{S} \subseteq \mathsf{LPT}$ will be proven. Suppose that $s \in \mathcal{S}$. If $s \in \mathbb{E}$, then $s \in \mathsf{LPT}$ by lemma 4.2. Suppose $s \in \mathbb{N}$. Suppose that $t_2 \xleftrightarrow{s} t_1 \xrightarrow{\mathbb{E},\mathsf{nf}}\!\!\!\!\rightarrow t_3$ and it will be proven that there exists t_4 such that $t_2 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_4 \xleftrightarrow{\mathbb{N}}\!\!\!\!\rightarrow t_3$. By cases on whether $t_1 \xrightarrow{s} t_2$ or $t_2 \xrightarrow{s} t_1$.

1. Case $t_2 \xrightarrow{s} t_1$. Let $m \neq \mathsf{noMark}$ be a mark such that $\neg\mathsf{markOccurs}(m,t_2)$ (this exists by axiom 6.1(5)). By diagram:

$$\begin{tikzcd} t_2 \arrow[rrr, dotted, two heads, "\mathbb{E}"] \arrow[dr, dotted, "\mathsf{rename}(m,m')"] \arrow[ddd, "\mathbb{N},s"'] & & & t_4 \\ & \mathsf{NM}(\mathcal{S}) \quad \text{lem. D.1(1)} & t_6 \arrow[r, dotted, two heads, "\mathbb{E}"] \arrow[dl, dotted, "\mathbb{N},\mathbb{M},\mathcal{S}"] & t_5 \arrow[ur, dotted, "\mathsf{rename}(m,m')"] \arrow[u, dotted, two heads, "\mathbb{N}"] \arrow[d, dotted, two heads, "\mathbb{N}"] \\ & & \mathcal{S}\cap\mathbb{M} \subseteq \mathsf{LPT} & \text{lem. D.1(2)} \\ t_1 \arrow[rrr, two heads, "\mathbb{E},\mathsf{nf}"] & & & t_3 \arrow[r, dotted, "\mathsf{rename}(m,m')"] & t_7 \end{tikzcd}$$

   Because $t_2 \twoheadrightarrow t_3$, $\neg\mathsf{markOccurs}(m,t_2)$, and $m \neq \mathsf{noMark}$, by lemma D.1(4) it holds that $\neg\mathsf{markOccurs}(m,t_3)$. Then $\mathsf{rename}(m,m')(t_3) = t_7 = t_3$ by axiom 6.1(3). Thus, $t_2 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_4 \xleftrightarrow{\mathbb{N}}\!\!\!\!\rightarrow t_3$. Thus, $s \in \mathsf{LPT}$.

2. Case $t_1 \xrightarrow{s} t_2$. Let $m \neq \mathsf{noMark}$ be a mark such that $\neg\mathsf{markOccurs}(m, t_1)$ (this exists by axiom 6.1(5)). By diagram:



Because $t_1 \twoheadrightarrow t_4$, $\neg\mathsf{markOccurs}(m, t_1)$, and $m \neq \mathsf{noMark}$, by lemma D.1(4) it holds that $\neg\mathsf{markOccurs}(m, t_4)$. Then $(\mathsf{rename}(m, m'))(t_4) = t_7 = t_4$ by axiom 6.1(3). Thus, $t_2 \xrightarrow{\mathbb{E}}\!\!\!\!\rightarrow t_4 \xleftarrow{\mathbb{N}}\!\!\!\!\rightarrow t_3$. Thus, $s \in \mathsf{LPT}$. $\square$