A Safety and Liveness Theory for Total Reversibility (Extended Abstract)

Claudio Antares Mezzina\textsuperscript{1} and Vasileios Koutavas\textsuperscript{2}

\textsuperscript{1} IMT School for Advanced Study Lucca, Lucca, Italy  
claudio.mezzina@imtlucca.it  
\textsuperscript{2} Trinity College Dublin, Dublin, Ireland  
avasileios.koutavas@scss.tcd.ie

Abstract

We study the theory of safety and liveness in a reversible calculus where reductions are totally ordered and rollback moves return systems to past states. Similar to previous work on communicating transactions, liveness and safety respectively correspond to the should-testing and inverse may-testing preorders. We develop fully abstract models for these preorders in terms of forward transitions in a labelled transition system, and give sufficient conditions for these models to apply to other calculi. Avoiding rollback transitions in the characterisations simplifies reasoning. We show that with respect to safety, total reversibility is a conservative extension to CCS. With respect to liveness, however, adding total reversibility to CCS distinguishes more systems. To our knowledge, this work provides the first characterisations of safety, liveness and testing for reversible calculi.

1 Introduction

A reversible system is a system that can execute forwards but also backwards, reversing the effects of its computations. Systems that can go back to past states appear in different disciplines, including fault-tolerant systems \cite{10}, reverse debugging \cite{11}, transactional systems \cite{23}, and computational biology \cite{4, 19}. Moreover, there is recent interest in low energy reversible computing following the experimental validation \cite{2} of Landauer’s principle \cite{14}, which states that only information loss in irreversible computation needs to consume energy. Reversibility unites these disciplines, aiming to develop theoretical foundations and relevant technology of their common concept.

A number of approaches model general reversible computation as an extension of standard process calculi. Depending on the flavour of reversibility these approaches can be divided into three categories: uncontrolled reversibility, where every forward reduction can be reversed non-deterministically by the operational semantics (e.g., \cite{5, 20}); reversibility with irreversible actions, where certain reductions are irreversible (e.g., \cite{6}); and controlled reversibility, where reverse reductions are explicitly programmed (e.g., \cite{15, 19}).

The behaviours encoded in these calculi can be understood by executing instructive examples, but more thoroughly by developing the theory of a contextual preorder (or equivalence). Such a preorder relates systems $M \leq N$ of a reversible calculus when, for a certain class of contexts $C$ and notion of observation, the observable behaviour of $C[M]$ is also observable in $C[N]$. In the presence of non-determinism there are numerous choices for such a preorder \cite{12}, with most common being the may- and must-testing preorder \cite{7}, should-testing preorder \cite{21}, and reduction barbed congruence \cite{18}. To our knowledge no theory of a contextual preorder or equivalence has been developed for a reversible calculus.
In this paper we study a simple form of controlled reversibility and develop the theory for two contextual preorders: the safety and liveness preservation preorders. As explained in previous work for communicating transactions [9] the former corresponds to the inverse may-testing preorder and the latter to the should-testing preorder.

To briefly explain this, consider a safety property \( \mathcal{P} \) which expresses that “something bad will not happen” [13]. Safety properties are exactly the properties enforced by monitors [22], which in our framework are test processes running in parallel with the system and report properties violations on a special channel \( \omega \). Thus, if \( M \subseteq_{\text{safe}} N \) and \( T \) is the monitor of \( \mathcal{P} \), if \( N \parallel T \) can output on \( \omega \) (i.e., one execution of \( N \) violates the property monitored by \( T \)) then \( M \parallel T \) can also output on \( \omega \). This means that \( N \subseteq_{\text{may}} M \) according to may-testing.

A liveness property \( \mathcal{P}' \) expresses that “something good will eventually happen” [13]. This property can also be encoded as test process \( T' \) which reports on \( \omega \) the good outcome. For a system \( M \) to pass the liveness test \( T' \), the parallel composition \( M \parallel T' \) should be able to eventually reach \( \omega \) from all of its reachable states. Thus, if \( M \subseteq_{\text{live}} N \) and \( M \parallel T' \) can always reach \( \omega \) then so does \( N \parallel T' \). Therefore \( M \subseteq_{\text{should}} N \) according to should-testing.

Note that for a system with an infinite execution to pass a test according to should-testing, merely requires \( \omega \) to be reachable from all (infinite) states of the execution. According to must-testing, however, the infinite execution must reach \( \omega \) at some finite prefix. This difference makes should-testing more suitable for systems with any form of reversibility where even a single backwards reduction creates potential infinite loops of abort-and-retry.

Our model of the safety preorder is (inverse) forward trace inclusion, identical to CCS proper. This means that reversibility is a conservative extension of CCS with respect to safety. Perhaps not surprisingly, this result relies on the property that any state reachable along the forward traces in \( M' \) can also output on \( \omega \) from all of its reachable states. Thus, if \( M \subseteq_{\text{live}} N \) and \( M \parallel T' \) can always reach \( \omega \) then so does \( N \parallel T' \). Therefore \( M \subseteq_{\text{should}} N \) according to should-testing.

The question now becomes: does a sound characterisation of the liveness preorder need to consider the full generality of the reversible transition system and explore all states reachable with forward and backward transitions from any other state? In this paper we answer this question to the negative, providing a simple model for the liveness preorder. This model is based on tree refusals of the form \( (t; V; W) \). If \( M \) has this refusal then it can perform the forward trace \( t \) and reach a state \( M' \), from which it cannot perform in full the forward traces in the set \( W \), and cannot roll back before \( M' \) along the forward traces in \( V \).

The benefit of our theory is that it only requires a compositional semantics for forward moves. This makes the observable behaviour easy to understand and reason about in proofs. There is however one caveat: we prove that this model is sound and complete for a simple reversible language \( \text{CCS}_{\text{roll}} \) with temporal reduction dependencies. This means that a system rolling back a past reduction \( r \) reverses all reductions that followed \( r \). This is in contrast to traditional causality in reversible calculi where rolling back \( r \) reverses only those reductions that occurred because of \( r \). However, we identify a key property which, if proven for a reversible calculus in addition to the one discussed above for safety, our liveness model will be sound and complete for that calculus. We were not able to prove or disprove this property for CCS extended with traditional controlled reversibility, which remains an open question.
A Safety and Liveness Theory for Total Reversibility

\[ \alpha ::= \tau \mid a \quad \iota ::= k \mid \gamma \quad a, b, \ldots \in \text{Name} \quad k, l \in \text{Key} \quad \gamma, \delta \in \text{KeyVar} \]

\[ P, Q ::= \sum_{i \in I} \alpha_i(\gamma).P_i \mid P \parallel Q \mid \nu a.P \mid \text{rec} X(\gamma).P \mid X \mid \text{rl}(\iota) \]  (Proc)

\[ M, N ::= \emptyset \mid \nu a.M \mid M \parallel N \mid k; P \mid [\mu; k] \]  (Sys)

\[ \mu ::= k; \sum_{i \in I} \alpha_i(\gamma).P_i \]  (Mem)

\[ \nu a.0 = 0 \]

\[ (\nu a.A) \parallel B = \nu a.(A \parallel B) \quad \text{(when } a \not\in B) \]

\[ \nu a.\nu b.A = \nu b.\nu a.A \]


\section*{Figure 1} CCS\textsubscript{roll} syntax.

In Sections 2 and 3 we respectively present CCS\textsubscript{roll} and the contextual preorders for safety and liveness. In Section 4 we describe the two crucial semantic properties for the soundness of our forward-transition models of the preorders. In Section 5 we give a compositional semantics in terms of a labelled transition system (LTS) for forward moves and develop necessary results for LTS traces which we use to give the models of the preorders in Sections 6 and 7. In Section 8 we return to the main semantic properties in the context of a traditional reversible language. Section 9 contains related work and conclusions.

\section{The Language CCS\textsubscript{roll}}

The language CCS\textsubscript{roll} extends CCS with a form of controlled reversibility, where reductions are totally ordered and running systems can be programmed to return to any previous state. As in CCS, synchronisation between processes occurs over channel names (Name) according to a total, irreflexive bijection (\(\tau\)) over Name. Unlike CCS, unique keys (Key) are used to identify and roll back synchronisation and internal (\(\tau\)) reductions.

The CCS\textsubscript{roll} syntax is shown in Figure 1 and is organised in two levels: processes and systems. Processes (Proc) include the potentially infinite choice operator \(\sum_{i \in I} \alpha_i(\gamma).P_i\), where \(I\) is an indexing set and \(\gamma\) a key variable. When the prefix is reduced, \(\gamma\) is replaced with a fresh key \(k\) in the continuation process. At any point, the system can roll back to the state before this reduction by executing \(\text{rl}(k)\). Unfolding a recursive process is an internal reduction in CCS\textsubscript{roll}, thus \(\text{rec} X(\gamma).P\) also includes a bound variable \(\gamma\) to facilitate the roll back of the unfolding. Processes also include \(P \parallel Q\), which can be thought of spawning \(P\) and \(Q\) in parallel, and \(\nu a.P\) which restricts the scope of \(a\) in \(P\).

Running systems (Sys) may contain zero or more parallel named processes \(k; P\). The key \(k\) identifies the reduction that produced \(P\), which may be shared with other processes produced by the same reduction. Systems may also contain memories \([\mu; k]\), each recording a named process that was involved in the \(k\) reduction. If a past \(k\)-reduction was a synchronisation, the current running system will contain two memories \([\mu_1; k]\) and \([\mu_2; k]\) recording the named processes that synchronised. Otherwise the \(k\)-reduction was internal and the running system will contain only one memory of the form \([\mu; k]\). We make use of two structural equivalences over \(A, \ldots \in \text{Sys} \cup \text{Proc}\), a limited and an extended one.

\begin{definition}[Structural Equivalences] Limited structural equivalence (\(\equiv\)) for systems is defined to be the least equivalence satisfying the axioms

\[ k; \nu a.P = \nu a.k; P \]

\[ k;(P \parallel Q) = k;P \parallel k;Q \]

and is closed under parallel (\(- \parallel -\)) and name restriction (\(\nu a.-\)). Structural equivalence (\(\equiv\)) is obtained by requiring the additional axioms

\[ \emptyset \parallel A = A \]

\[ A \parallel B = B \parallel A \]

\[ (A \parallel B) \parallel C = A \parallel (B \parallel C) \]

\(\nu a.\emptyset = \emptyset\)

\([\nu a.A] \parallel B = \nu a.(A \parallel B) \quad \text{(when } a \not\in B)\]

\([\nu a.\nu b.A = \nu b.\nu a.A\]

\[ \nu a.\nu b.A = \nu b.\nu a.A \]

\end{definition}
A Safety and Liveness Theory for Total Reversibility

\[
\begin{align*}
\text{Rα} & \quad j \in I \quad k \not\in (P_i)_{i \in I} \\
\sum_{i \in I} \alpha_i(\gamma).P_i & \xrightarrow{\alpha_j(k)} P_j\{k/\gamma\} \\
\text{REC} & \quad k \not\in P \\
\text{rec}\{\gamma\}.P & \xrightarrow{\gamma(k)} P\{\text{rec}\{\gamma\}.P/X\}\{k/\gamma\} \\
\text{RSYNC} & \quad P \xrightarrow{a(k)} P' \quad Q \xrightarrow{\pi(k)} Q' \\
& \quad k \not\in k_1, k_2, P, Q \\
\text{Rτ} & \quad P \xrightarrow{\tau(k)} P' \\
& \quad k_0.\{k\} \quad k_0.\{k\} \quad [k_1; P; k] \\
\text{RPAR} & \quad M \xrightarrow{(k)} M' \quad k \not\in N \\
& \quad M \parallel N \xrightarrow{(k)} M' \parallel N \\
\text{Rν} & \quad M \xrightarrow{(k)} M' \\
& \quad \nu a.M \xrightarrow{(k)} \nu a.M' \\
\text{REQV} & \quad M \equiv N \quad N \xrightarrow{(k)} N' \quad N' \equiv M' \\
\text{RSYS} & \quad M \xrightarrow{(k)} M' \quad k \not\in D \\
& \quad D \vdash M \xrightarrow{(k)} k \prec D \vdash M' \\
\text{FW} & \quad D \vdash M \xrightarrow{(k)} D' \vdash M' \\
& \quad D \vdash M \rightarrow D' \vdash M' \\
\end{align*}
\]

Figure 2 Reduction Semantics: Forward Rules.

Before a system starts running, it contains no memories and no keys corresponding to past reductions. We call these systems initial. To abide to the syntax for systems we consider one special key ε which is used to annotate processes in initial systems of the form \( M \equiv \varepsilon.P \). In the following we use standard abbreviations from CCS, such as \( \overline{o} \) for a sequence of syntactic objects \( a_1, \ldots, a_n \), where \( n \) is implicit. We write keys(\( o \)) for the set of keys in \( o \) except for \( \varepsilon \), and \( a_1 \not\prec a_2 \) when keys(\( a_1 \)) \( \cap \) keys(\( a_2 \)) = \( \emptyset \). We omit variables \( \gamma \) when not used.

The reduction semantics of CCS_{roll} is defined by transitions between systems of the form \( D \vdash M \), where \( D \) is a dependency history, recording the order of reductions and derived by

\[
D::= \varepsilon \mid k \prec D \quad \text{(if } k \not\in D) 
\]

The side-condition guarantees that each \( k \) in a dependency history is recorded at most once. We write \( D \prec D' \) for the concatenation of \( D \) and \( D' \), and \( k \prec D \) for \( k_1 \prec \ldots \prec k_n \prec D \). We also write \( D \prec \varepsilon \) when \( D = D_1 \prec k_1 \prec l \prec D_2 \), and \( \varepsilon^+ \) (and \( \varepsilon^- \)) for the transitive (resp. reflexive and transitive) closure of \( \prec \).

The reduction semantics of CCS_{roll} systems are divided between forward reductions (\( D \vdash M \rightarrow D' \vdash M' \)) and backward reductions, or rollbacks, (\( D \vdash M \rightarrow D' \vdash M' \)) shown in Figures 2 and 3, respectively. Together form the reduction relation (\( D \vdash M \rightarrow D' \vdash M' \)).

A forward reduction \( D \vdash M \rightarrow D' \vdash M' \) is derived from rule FW of Figure 2 only when \( D \vdash M \xrightarrow{(k)} D' \vdash M' \), which in turn is derived from rule Rsys when \( D' = k \prec D \) and \( M \xrightarrow{(k)} M' \). The key \( k \) is fresh (\( k \not\in D, M \)) and uniquely identifies the reduction. The rules deriving \( M \xrightarrow{(k)} M' \) are adapted from CCS and enable synchronisation (RSync), internal reduction (Rτ) and propagation (Rpar, Rν); they are also closed under structural equivalence (REQV). When a prefix is reduced (via rule Rα or REC) the bound variable \( \gamma \) is substituted with the key identifying the reduction in the continuation of the process, replacing any free occurrence of \( \gamma \) in terms \( r1(\gamma) \).

A backward reduction \( D \vdash M \rightarrow D' \vdash M' \) is derived from rule Bw of Figure 3 only when \( M \) can execute a \( k \)-rollback (\( M \equiv l.r1(k) \parallel N \)) and \( D \vdash M \xrightarrow{(k)} D' \vdash M' \). The latter is derived...
from rule RLsys if $M \xrightarrow{(k_1) \ldots (k_n)} M'$ and $k = k_n$. Each individual $M \xrightarrow{(k)} N$ reduction is derived from the rules in the first part of Figure 3 and broadcast the $k$-rollback throughout the system $M$. Processes and memories in $M$ that do not contain $k$ are left unaffected by this transition; $k$-processes disappear and $k$-memories reinstate their contents. In the following we write $(\bar{k})$ and $\langle \bar{k} \rangle$ for $(k_1) \ldots (k_n)$ and $\langle k_1 \rangle \ldots \langle k_n \rangle$, respectively.

In CCS_roll, rollbacks are deterministic. Moreover, any forward $k$-reduction can be rolled back, and when this happens the system returns to the state before the reduction, up to structural equivalence.

- **Lemma 2.2** (Deterministic Rollback). Let $D \vdash M \xrightarrow{(k)} D' \vdash M'$ and $D \vdash M \xrightarrow{(k)} D'' \vdash M''$. Then $D' = D''$ and $M' = M''$.

- **Lemma 2.3** (Rollback a Forward Reduction). If $D \vdash M \xrightarrow{(k)} D' \vdash M'$ then there exists $M''$ such that $D' \vdash M' \xrightarrow{(\bar{k})} D'' \vdash M''$ and $M \equiv M''$.

To establish more properties for systems we require them to be well-formed.

- **Definition 2.4** (Well-Formed System). $D \vdash M$ is well-formed, written $\text{wf}(D \vdash M)$, when
  1. Key Compatibility: $\text{keys}(M) \subseteq \text{keys}(D)$,
  2. Rollback Loop: if $D = k \prec D'$ then $M \xrightarrow{(k)} M' \xrightarrow{(k)} M$ and $\text{wf}(D' \vdash M')$, for some $M'$.

This definition guarantees that in a system $D \vdash M$, keys in $M$ were produced by a past reduction recorded in $D$, and any such reduction can be rolled back and repeated obtaining the same system. These properties are sufficient to describe well-behaved systems, simplifying the definitions of previous work [15].

Well-formedness is preserved by structural equivalence, name opening and reductions. This, together with the fact that initial systems are trivially well-formed, allows us to implicitly assume well-formed systems in the following sections.

### 3 Safety and Liveness Preorders

In this section we give the definitions of the safety and liveness preorders, and examples of their use. As we discussed in the introduction, the safety preorder corresponds to the inverse may-testing preorder [7] and the liveness preorder corresponds to should-testing.
preorder [21]. Here we use tests \( T \) derived from the grammar of processes, with the addition of a special name \( \omega \) used by the test to report an outcome.

- **Definition 3.1 (Basic Observable (barb)).** \( D \vdash M \) has a strong barb, written \( D \vdash M \downarrow \omega \), when \( M \equiv N \parallel k : \omega \) and a weak barb, written \( D \vdash M \downarrow^* \omega \), when \( D \vdash M \downarrow^* D' \vdash M' \downarrow \omega \).

  We are interested in testing initial systems (Definition A.1); the composition of a system \( \varepsilon : M \) with a test \( \varepsilon : T \) is \( \varepsilon : M \parallel \varepsilon : T \).

  We start with safety. A safety property of a system expresses that “something bad will not happen” [13]. A safety test \( T \) can be thought to be a monitor enforcing a safety policy. Indeed, safety policies are exactly the policies enforced by monitors [22]. When \( T \) reports an error on \( \omega \) then the enforced safety policy has been violated by the system. Thus, a system \( M \) passes a safety test \( T \) when their parallel composition cannot report a violation on \( \omega \). This means that all non-deterministic computations of \( \varepsilon : M \parallel \varepsilon : T \) do not contain a state \( D \vdash M \downarrow \omega \). In negative form, \( M \) fails \( T \) if \( \varepsilon : M \parallel \varepsilon : T \downarrow \omega \). System \( M \) is potentially “less safe” than \( N \) when any violation of \( N \) is also a violation of \( M \).

- **Definition 3.2 (Safety Preorder).** For two initial systems \( M \) and \( N \) we write \( M \sqsubseteq_{\text{safety}} N \) when for all tests \( T \), \( \varepsilon : T \parallel \varepsilon : \omega \) implies \( \varepsilon : M \parallel \varepsilon : T \downarrow \omega \).

  A liveness property of a system expresses a promise that “something good will happen” [13]. We can consider liveness tests that report on \( \omega \) when the “something good” of the liveness property happens. A system \( M \) passes a liveness test \( T \) when their parallel composition always succeeds. As we discussed in the introduction, we follow the definition of should testing [21] to express success for liveness tests. According to this definition, a system passes a test if at any state reachable from their parallel composition, success is possible.

- **Definition 3.3 (Passing a Liveness Test).** If \( M \) is an initial system, \( M \) passes the liveness test \( T \), written \( M \text{ shd } T \), when for all \( D \vdash N \) such that

\[
\varepsilon : M \parallel \varepsilon : T \rightsquigarrow^* D \vdash N \implies D \vdash N \downarrow \omega
\]

In other words, for \( M \) to fail a liveness test \( T \), it must be that \( \varepsilon : M \parallel \varepsilon : T \rightsquigarrow^* D \vdash N \downarrow \omega \). System \( M \) is “less live” than \( N \) if \( N \) passes every liveness test that \( M \) passes.

- **Definition 3.4 (Liveness Preorder).** For two initial systems \( M \) and \( N \) we write \( M \sqsubseteq_{\text{live}} N \) when for all liveness tests \( T \), \( M \text{ shd } T \) implies \( N \text{ shd } T \).

We consider the systems in Figure 4. Systems \( M_1 \) and \( M_2 \) are essentially plain CCS processes. In CCS these are safe-equivalent because they have the same traces. With respect to the liveness preorder, also in CCS, \( M_2 \sqsubseteq_{\text{live}} M_1 \) but \( M_1 \nsubseteq_{\text{live}} M_2 \) because \( M_1 \) passes the liveness test \( T = \omega + \pi . b . \omega \) but \( M_2 \) doesn’t: after a communication on \( a \), \( M_2 \parallel T \) can become \( c.0 \parallel \tau . \omega \), which cannot reach \( \omega \). The same is true in CCSroll. In fact the following conservative extension theorem holds for safety in CCSroll: it’s proof relies in the models for safety in the two languages.
Theorem 3.5 (Conservative Extension). Let $P$ and $Q$ be CCS processes with $P \subseteq_{\text{safe}} Q$ (in CCS); then $\varepsilon : P \subseteq_{\text{safe}} \varepsilon : Q$ (in CCSroll).

Terms $M_5$ and $M_4$ show why CCSroll is not a conservative extension of CCS with respect to liveness. In CCS $M_5$ is safe-equivalent to $M_4$ [21, Ex. 34], however these terms can be distinguished in CCSroll by test $T = \omega + \tau.\tau(\gamma).((\exists 1(\gamma)) || \overline{b}d.\omega)$. System $M_5$ passes this test because if, while communicating with $T$, it chooses the “wrong” $b$-branch $b.d.\emptyset$, the rollback in the test reverses this choice and $\omega$ is reachable. On the other hand, $M_4$ can communicate with $T$ using branch $a.b.d.\emptyset$ where $\omega$ will not be reachable; the rollback in the test cannot reverse this choice. For a similar reason, $M_5$ and $M_6$ are safe- but not live-equivalent. System $M_6$ passes the liveness test $T = \omega + a.b$ but $M_5$ does not; the rollback of the $a$-communication in $M_6$ makes $\omega$ to always be reachable.

### 4 Important Properties of Forward Reductions

Our safety and liveness models rely on two semantic properties. The first is that, starting from any system, any state reachable with arbitrary reductions can also be reached with only forward reductions after at most one backward reduction. Thus, all states of an initial system can be reached with only forward reductions. This essentially means that the forward traces over a compositional labelled transition system are sufficient to characterise safety.

Our safety characterisation relies only on this property. Moreover, this property holds for traditional controlled reversibility [15] but also for other forms of reversibility (e.g., [5, 6]). Therefore, our forward-only model of safety applies to other reversible systems.

Lemma 4.1. Let $D \vdash M \dashv^{*} D' \vdash M'$. Then one of the following holds:

1. $D \vdash M \dashv^{*} D' \vdash M'' \equiv M'$, for some $M''$, or
2. $D \vdash M \vdash (D_0 \vdash M_0 \dashv^{*} D' \vdash M'' \equiv M'$, and $D \vdash M \dashv^{*} D_0 \vdash M_0' \equiv M_0$, for some $l \in D$, $D_0$, $M''$, $M_0$, $M_0'$.

Corollary 4.2. Let $\varepsilon \vdash M \dashv^{*} D' \vdash M'$. Then $\varepsilon \vdash M \dashv^{*} D' \vdash M'$.

The above property is also necessary for the characterisation of the liveness preorder in terms of forward traces. However we also need to establish a result for tree failures: if an initial system can reach a state $D \vdash N$ from which it fails to reach an $\omega$-action, then the same original system should be able to reach a failure state $D \vdash N_1$ where all reachable states can be reached with forward reductions. This allows us to use a forward LTS to encode liveness.

Lemma 4.3. Let $\varepsilon \vdash M \dashv^{*} D \vdash N \not\not\not\not_{\omega}$ and $\text{keys}(M) = \emptyset$; there exist $D_1$ and $N_1$:

1. $\varepsilon \vdash M \dashv^{*} D_1 \vdash N_1 \not\not\not\not_{\omega}$
2. if $D_1 \vdash N_1 \dashv^{*} D' \vdash N'$ then there exists $N''$ such that $D_1 \vdash N_1 \dashv^{*} D' \vdash N'' \equiv N'$

In CCSroll, where reductions are temporally dependent, we can show that once we reach the state $D \vdash N$ we can explore all past states reachable with rollbacks from $D \vdash N$ and pick the oldest one. Because of the total temporal ordering of reductions we know that there is always a single oldest state (up to $\equiv$) and it can reach with only forward reductions all the states that $D \vdash N$ can reach with forward and backward reductions (up to $\equiv$). So in fact we prove instead the following lemma from which Lemma 4.3 follows.

Lemma 4.4. Let $\varepsilon \vdash M \dashv^{*} D \vdash N$ and $\text{keys}(M) = \emptyset$; there exist $D_1$ and $N_1$:

1. $\varepsilon \vdash M \dashv^{*} D_1 \vdash N_1$
2. if $D_1 \vdash N_1 \dashv^{*} D' \vdash N'$ then $D \vdash N \dashv^{*} D' \vdash N'$
3. if $D \vdash N \dashv^{*} D' \vdash N'$ then there exists $N''$ such that $D_1 \vdash N_1 \dashv^{*} D' \vdash N'' \equiv N''$
4. if $D_1 \vdash N_1 \dashv^{*} D' \vdash N'$ then there exists $N''$ such that $D_1 \vdash N_1 \dashv^{*} D' \vdash N'' \equiv N''$
Compositional Semantics

Our characterisation of the testing behaviour of $\text{CCS}_{\text{roll}}$ systems will be based solely on compositional forward transitions, simplifying reasoning. We construct a labelled transition system (LTS) of forward transitions between configurations of the form $D \models M$, ranged over by $\mathcal{C}$ shown in Figure 5. These transitions have labels $\alpha(k)$, ranged over by $\lambda$.

These transitions, besides internal reductions, can describe the interaction of a part of a system with its environment, which we call the observer. We assume the adaptation of the definition of well-formed systems to configurations and work with well-formed configurations.

Forward reductions correspond to $\tau$-transitions in the LTS.

\textbf{Theorem 5.1.} $D \vdash M \xrightarrow{(k)} D' \models M'$ iff $D \models M \xrightarrow{\tau(k)} D' \models M'' \equiv M'$.

Our theory is based on canonical traces; $t$ is canonical if each key in $t$ appears at most once in $t$. A trace is a dependency history transformer and can be typed as such.

\textbf{Definition 5.2 (Trace typing).} We write $(D \vdash t \triangleright D')$ for the predicate defined by the following rules:

\[
(D \vdash \epsilon \triangleright D) \quad \quad (D \vdash \alpha(k), t \triangleright D') \text{ if } (k \prec D \vdash t \triangleright D')
\]

We will treat $(D \vdash t \triangleright D')$ as a typed trace; this formalism helps us synchronise dependency histories with traces. Canonical traces are typable, provided they use new keys, and any typed trace is canonical.

Traces encode both the observable and internal ($\tau$) actions of a system. Systems related by the safety and liveness preorders may of course have traces that differ in their internal actions. We write $\text{obs}(t)$ to denote the sub-trace of $t$ containing only non-$\tau$ actions.

\[
\text{obs}(\epsilon) = \epsilon \quad \text{obs}(\tau(k), t) = \text{obs}(t) \quad \text{obs}(a(k), t) = a(k), \text{obs}(t)
\]

We say that a trace $t$ is observable when it contains no $\tau$ actions, thus $\text{obs}(t) = t$. We write $\overline{t}$ to denote the same-length trace derived from $t$ by applying $(\overline{\tau})$ to all non-$\tau$ actions. If $t_1 = \overline{t_2}$ then we call these traces complementary. A single LTS transition of two parallel systems can be decomposed to either a transition of one of the systems, or a synchronisation between them. This leads to a general decomposition of the trace of two parallel systems.
A Safety and Liveness Theory for Total Reversibility

**Definition 6.3 (Trace Preorder).** For initial systems, \( M \sqsubseteq \sim tr N \) if \( Tr(\epsilon \triangleright M) \subseteq Tr(\epsilon \triangleright N) \).

\[
\begin{align*}
\varepsilon \parallel \varepsilon & \rightarrow \varepsilon \\
(k \prec D_1) \parallel D_2 & \rightarrow k \prec D \quad \text{if} \quad D_1 \parallel D_2 \rightarrow D \quad \text{and} \quad k \nmid D_1, D_2 \quad (Z \varepsilon) \\
D_1 \parallel (k \prec D_2) & \rightarrow k \prec D \quad \text{if} \quad D_1 \parallel D_2 \rightarrow D \quad \text{and} \quad k \nmid D_1, D_2 \quad (Z \varepsilon) \\
(k \prec D_1) \parallel (k \prec D_2) & \rightarrow k \prec D \quad \text{if} \quad D_1 \parallel D_2 \rightarrow D \quad \text{and} \quad k \nmid D_1, D_2 \quad (Z \varepsilon)
\end{align*}
\]

\( \text{ Definition 6.3(Trace Preorder)} \)

\[
\begin{align*}
D_1 \parallel D_2 & \rightarrow D \\
(D_1 \vdash \epsilon \triangleright D_1) \parallel (D_2 \vdash \epsilon \triangleright D_2) & \rightarrow (D \vdash \epsilon \triangleright D)
\end{align*}
\]

\( \text{ZTL} \)

\[
\begin{align*}
(D_1 \vdash a(k), t_1 \triangleright D_1') \parallel (D_2 \vdash t_2 \triangleright D_2') & \rightarrow (D \vdash \tau(k), t \triangleright D') \\
(D_1 \vdash a(k), t_1 \triangleright D_1') \parallel (D_2 \vdash t_2 \triangleright D_2') & \rightarrow (D \vdash \tau(k), t \triangleright D')
\end{align*}
\]

\( \text{ZTSYNC} \)

\( \text{Proposition 5.3 (Unzipping Trace).} \) Let \( D \models M \parallel N \triangleleft D' \models R' \) and \( \text{obs}(t) = \epsilon \) and \( D_1 \parallel D_2 \rightarrow D \). There exist \( M', N', D_1', D_2', t_1, t_2 \) such that

\[
\begin{align*}
D \models M \triangleleft D_1' \models M' & \quad (D_1 \vdash t_1 \triangleright D_1') \parallel (D_2 \vdash t_2 \triangleright D_2') \rightarrow (D \vdash t \triangleright D') \\
D \models N \triangleleft D_2' \models N' & \quad R' \models M' \parallel N' \quad \text{obs}(t_1) = \text{obs}(t_2)
\end{align*}
\]

Conversely, if two systems can perform typed traces that can be into a single zipped trace, then the parallel composition of these systems can perform this zipped trace.

\( \text{Proposition 5.4 (Zipping Traces).} \) Suppose \( D_1 \models M \triangleleft D_1' \models M' \) and \( D_2 \models M \triangleleft D_2' \models N' \) and \( (D_1 \vdash t_1 \triangleright D_1') \parallel (D_2 \vdash t_2 \triangleright D_2') \rightarrow (D \vdash t \triangleright D') \). Then \( D \models M \parallel N \triangleleft D' \models R' \) with \( R' \models M' \parallel N' \).

## 6 Model of the Safety Preorder

Here we show that the safety preorder coincides with the inverse observable trace inclusion.

\( \text{Definition 6.1 (Configuration Trace Set).} \) We write \( \text{Tr}(C) \) for the largest set of observable traces such that \( t \in \text{Tr}(C) \) when there exists \( t' \) and \( C' \) such that \( \text{obs}(t') = t \) and \( C \models t' \rightarrow C' \).

Observable traces correspond to the class of safety tests defined by the rules:

\[
\text{Test}^s(\epsilon) = \omega \quad \text{Test}^s(a(k), t) = \overline{a}.\text{Test}^s(t)
\]

\( \text{Lemma 6.2.} \) Let \( t \) be an observable trace; then

\( \text{1. there exist } D, T \text{ such that } \varepsilon \vdash \text{Test}^s(t) \rightarrow D \vdash T \downarrow \omega; \)

\( \text{2. if } \varepsilon \vdash \text{Test}^s(t) \rightarrow D \vdash T \downarrow \omega \text{ then there exists permutation } p \text{ such that } \overline{T} = pt'. \)

Our characterisation of the safety preorder is the inverse of the following trace preorder.

\( \text{Definition 6.3 (Trace Preorder).} \) For initial systems, \( M \sqsubseteq_{tr} N \) if \( \text{Tr}(\epsilon \models M) \subseteq \text{Tr}(\epsilon \models N) \).
A Safety and Liveness Theory for Total Reversibility

Theorem 6.4 (Soundness and Completeness). \( M \downarrow_{tr} N \) iff \( N \downarrow_{safe} M \).

It is now easy to check that systems \( M_1, M_2, M_3 \) and \( M_4 \) from Figure 4 are pairwise safe-equivalent because they have the same observable traces, and so are \( M_3 \) and \( M_4 \).

Model of the Liveness Preorder

Our model of the liveness preorder is based on forward traces but includes the following basic observable for rollback actions, determined entirely by the structure of terms.

Definition 7.1 (Rollback Barb). If \( \Delta \) is a set of keys, we write \( D \models M \downarrow_{tr}(\Delta) \) when \( \exists k \in \Delta \) such that \( M \equiv l.r1(k) \| N \), for some \( l, N \). We let \( D \models M \downarrow_{tr}''(D') \) mean \( D \models M \downarrow_{tr}(\text{keys}(D')) \).

Based on this basic observable for rollbacks, we define the set of observable traces of a configuration \( C \) which lead to a rollback of an action before \( C \).

Definition 7.2 (Rollback Traces). \( \text{Roll}(D \models M) \) is the set of observable traces for which \( t \in \text{Roll}(D \models M) \) iff there exists \( t' \) such that \( \text{obs}(t') = t \) and \( D \models M \xrightarrow{(C)} D' \models M \downarrow_{tr}(D) \).

Lemma 7.3. \( \text{Roll}(D \models M) \subseteq \text{Tr}(D \models M) \).

The main structure of our liveness model is a tree refusal [21] adapted for reversibility.

Definition 7.4 (Tree Refusal). A tree refusal is a tuple \((t; V; W)\), where:

1. \( t \) is an observable trace, and \( V \) and \( W \) are sets of such traces,
2. \( \epsilon \in V \) and \( V \) is prefix-closed,
3. \( \epsilon \notin W \) and \( W \subseteq V \).
4. \( V \) and \( W \) are closed under permutation of keys.

A tree refusal \((t; V; W)\) encodes how an initial system \( M \) can fail a liveness test \( T \):

- \( M \) communicates with \( T \) according to the actions in \( t \) and together reach the state \( D \models T \).
- From this state, an \( \omega \) output is not reachable and the liveness test fails.
- At this state the test is offering to communicate on the traces in \( W \) and then output on \( \omega \). Thus, the system cannot perform the traces in \( W \). Since any system can perform the empty trace, \( \epsilon \notin W \).
- If the state \( D \models T \) is rolled back the test will reach an \( \omega \) output. Moreover, at this state, all the traces that the test is offering to communicate are in \( V \), including those in \( W \); note that every test offers the empty trace \((\epsilon \in V \) \). Thus the system should not be able to roll back state \( D \models T \) while communicating with the test over \( V \). If the test can roll back \( D \models T \) over a trace \( t_0 \), we add \( t_0 \in W \).

Tree refusals correspond to the following class of characteristic liveness tests.

Definition 7.5 (Characteristic Liveness Tests).

\[
\begin{align*}
\text{Test}^t(\epsilon; V; W) & \overset{def}{=} \omega + \tau. \text{Test}^t(V; W) \\
\text{Test}^t(a(k); t; V; W) & \overset{def}{=} \omega + \pi. \text{Test}^t(t; V; W) \\
\text{Test}^t(V; W) & \overset{def}{=} \tau(\gamma). \left( \sum_{t_1 \in V \setminus W} \text{Test}^t(t_1; \emptyset) \right) \ + \left( \sum_{t_2 \in W} \text{Test}^t(t_2; \omega) \right) \parallel r1(\gamma) \\
\text{Test}^t(\epsilon; P) & \overset{def}{=} P \\
\text{Test}^t(a(k); t; P) & \overset{def}{=} \pi. \text{Test}^t(t; P)
\end{align*}
\]
As we discussed previously, an initial system fails $\text{Test}(t; V; W)$ only by communicating with the test along the trace $t$, thus reducing the test to $\text{Test}'(V; W)$. From this state, an $\omega$ is reachable only if the system communicates with the test along a trace $t_2 \in W$, or if the system rolls back to a previous state along any of the traces in $V$ which contains $W$. The $\text{rl}(\gamma)$ is used to non-deterministically bring the test to state $\text{Test}'(V; W)$, thus avoiding the deadlock of system and test while communicating along a strict prefix of a trace in $V$.

**Definition 7.6 (Refusal Set).** Let $M$ be an initial system; $\text{Ref}(M)$ is the largest set of tree refusals with the property that if $(t; V; W) \in \text{Ref}(M)$, there exist $t'$ and $C$ such that

$$\text{obs}(t') = t \quad \text{and} \quad \varepsilon \models M \xrightarrow{(t')C} C \quad \text{and} \quad V \cap \text{Roll}(C) = W \cap \text{Tr}(C) = \emptyset$$

**Lemma 7.7 (Refusals to Traces).** If $(t; V; W) \in \text{Ref}(M)$ then $t \in \text{Tr}(M)$.

**Definition 7.8 (Refusal Preorder).** For initial systems, $M \equiv_{\text{ref}} N$ if $\text{Ref}(M) \subseteq \text{Ref}(N)$.

**Theorem 7.9 (Soundness and Completeness).** $M \equiv_{\text{live}} N$ iff $N \equiv_{\text{ref}} M$.

Let us revisit Figure 4: $M_1 \equiv_{\text{live}} M_2$, $M_3 \equiv_{\text{live}} M_4$ and $M_5 \equiv_{\text{live}} M_6$ because

$$\begin{align*}
(a; \{\varepsilon, b\}; \{b\}) & \in \text{Ref}(M_2) & (a; \{\varepsilon, b\}; \{b\}) & \notin \text{Ref}(M_1) \\
(a; \{\varepsilon, b, c\}; \{b, c\}) & \in \text{Ref}(M_4) & (a; \{\varepsilon, b, c\}; \{b, c\}) & \notin \text{Ref}(M_3) \\
(a; \{\varepsilon, b\}; \emptyset) & \in \text{Ref}(M_6) & (a; \{\varepsilon, b\}; \emptyset) & \notin \text{Ref}(M_5)
\end{align*}$$

### 8 Causal Controlled Reversibility

We now consider a variation of $\text{CCS}_{\text{roll}}$ with causal action dependencies (similar to [15]). As we explained in the introduction and Section 4, our safety and liveness models depend on two properties, Corollary 4.2 and Lemma 4.3. It is easy to check that the former property holds for the causal calculus. However it is unclear to us if the latter property also holds.

Lemma 4.3 states that if a system reduces to a failing state $C$, from which a successful state is not reachable, then the system can reduce to another failing state $C'$ from which all future states are reachable with forward reductions. In $\text{CCS}_{\text{roll}}$, $C'$ is the oldest state reachable from $C$. In $\text{CCS}_{\text{roll}}$ with causal dependencies however, it is unclear that a single reachable state exists which causally precedes all other reachable states. The following example shows why.

**Example 8.1.** Consider the following processes $P$ and $Q$, where for convenience key variables are identified with the key they will be replaced at runtime.

$$\begin{align*}
P &= \nu a.(a(k_1).\pi \parallel a(k_2).\text{rl}(k_2) \parallel a(k_3).\text{rl}(k_3) \parallel \overline{\pi} \parallel \overline{\pi}) \\
Q &= \nu a, b, (b(k_1).\pi \parallel a(k_2).\text{rl}(k_2) \parallel a(k_3).\text{rl}(k_3) \parallel \pi \parallel \pi)
\end{align*}$$

The $\tau$ and rollback transitions of these processes include:

$$\begin{align*}
\text{C}_1 \xrightarrow{\tau(k_2)} \text{C}_1' \\
\xrightarrow{\varepsilon} P \\
\xrightarrow{\tau(k_3)} \text{C}_2 \\
\xrightarrow{\tau(k_2)} \text{C}_3 \\
\xrightarrow{\tau(k_3)} \text{C}_3'
\end{align*}$$

$$\begin{align*}
\text{C}_1' \xrightarrow{\tau(k_2)} \text{C}_1 \\
\xrightarrow{\varepsilon} Q \\
\xrightarrow{\tau(k_3)} \text{C}_2' \\
\xrightarrow{\tau(k_2)} \text{C}_3' \\
\xrightarrow{\tau(k_3)} \text{C}_3
\end{align*}$$
Consider the liveness test $T = c.\omega$. Process $Q$ fails the test by reducing to any state $C'_i$ from which a communication on $c$ is not reachable. Process $P$ cannot fail the test because even at $C_3$ it can reverse its $\tau$ moves and communicate on $c$ with the test from states $\epsilon:P$, $C_1$ or $C_2$. Thus $C'_3$ is a failing state but $C_3$ is not.

From $C_3$ (and $C'_3$) both $C_1$ and $C_2$ (resp. $C'_1$ and $C'_2$) are reachable with rollbacks and these two states are not causally related. In this example it is obvious that we can also reach $\epsilon:P$ (resp. $\epsilon:Q$) which is the causal predecessor of all reachable states and is also a successful (resp. failing) state. Crucially, here all reachable states from $\epsilon:P$ (resp. $\epsilon:Q$) are reachable using forward transitions. However it is unclear to us that in complicated examples such a common ancestor state would always be reachable from any successful (resp. failing) state.

9 Related Work and Conclusions

Causal reversibility was introduced in CCS [5], relating reversibility in a concurrent system with causality. In [20] a general approach to reverse CCS-like calculi given in particular SOS format is provided. Combinations of the two approaches lead to reversibility in higher-order pi [16], which can be encoded into higher order pi proper. With a weaker notion of causality, reversibility can model massive biological systems where different molecules of the same chemical species are indistinguishable [3]. Controlled reversibility is possible via explicit programmable commands [15], irreversible actions [6], or external controllers [19].

The theory of behavioural equivalences in reversibility has not been previously studied. However, back a forth barbed congruence for reversible higher order pi-calculus is introduced in [17] but its characterisation in terms of bisimulation is still open. Moreover the same work shows that classical notions of behavioural equivalences (bisimilarity or barbed congruence) are too discriminating in the reversible setting, while their reversible variants are too discriminating in the strong case since they distinguish the directions of reductions.

Strong back and forth barbed congruence for RCCS has been shown to coincide with strong back and forth bisimulation and that the latter corresponds to hereditary history-preserving bisimulation for a restricted calculus [1]. Strong relations, however, are not suitable notions of contextual equivalence since they observe internal moves.

In this paper we studied the theory of safety and liveness for a CCS extended with controlled reversibility named $\text{CCS}_{\text{roll}}$. In $\text{CCS}_{\text{roll}}$ reductions are temporally ordered, forming a total order, and rollback moves return systems to past states. We have opted for this form of reversibility instead of the more classical causally consistent one because it makes possible to develop a theory of liveness in terms of just forward transitions (as in [8]). The adoption of our theory to an uncontrolled reversible calculus is immediate since in such a setting all the states are reachable and may testing implies should testing.

We have identified two properties which are sufficient for the adoption of our theory. The first easily holds for other reversible calculi. The other is less clear. However, if a reversible calculus enjoys it, then we believe that our theory applies. We leave as future work to verify whether this property holds in a calculus with controlled causal reversibility such as [15].

We have showed that with respect to safety, total reversibility is a conservative extension of CCS. With respect to liveness, however, adding total reversibility to CCS distinguishes more systems. The characterisations we have developed for reversibility are fundamentally different than those for communicating transactions [9, 8], illuminating the difference between the two constructs. To our knowledge, this work provides the first characterisations of safety, liveness and testing for reversible calculi.

Acknowledgements: We thank Matthew Hennessy for illuminating discussions about
this work. This research is supported, in part, by the EU COST Action IC1405. The second author was supported by Science Foundation Ireland grant 13/RC/2094.

References

1. C. Aubert and I. Cristescu. Reversible barbed congruence on configuration structures. In ICE 2015, pages 68–85, 2015.
2. A. Berut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider, and E. Lutz. Experimental verification of Landauer’ s principle linking information and thermodynamics. Nature, 483(7388):187–189, 03 2012.
3. L. Cardelli and C. Laneve. Reversibility in massive concurrent systems. Sci. Ann. Comp. Sci., 21(2):175–198, 2011.
4. L. Cardelli and C. Laneve. Reversible structures. pages 131–140. ACM, 2011.
5. V. Danos and J. Krivine. Reversible communicating systems. In CONCUR, volume 3170 of LNCS, pages 292–307. Springer, 2004.
6. V. Danos and J. Krivine. Transactions in RCCS. In CONCUR 2005, pages 398–412, 2005.
7. R. De Nicola and M. C. B. Hennessy. Testing equivalences for processes. Theoretical Computer Science, 34(1-2):83–133, 1984.
8. E. de Vries, V. Kontavas, and M. Hennessy. Communicating transactions - (extended abstract). In CONCUR 2010, pages 569–583, 2010.
9. E. de Vries, V. Kontavas, and M. Hennessy. Liveness of communicating transactions (extended abstract). In APLAS 2010, pages 392–407, 2010.
10. E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson. A survey of rollback-recovery protocols in message-passing systems. ACM Comp. Surv., 34(3):375–408, 2002.
11. J. Engblom. A review of reverse debugging. In System, Software, SoC and Silicon Debug Conference (S4D), 2012, pages 1–6, Sept 2012.
12. R. J. Glabbeek. CONCUR ’90, chapter The linear time - branching time spectrum, pages 278–297. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990.
13. L. Lamport. Proving the correctness of multiprocess programs. IEEE Trans. Softw. Eng., 3(2):125–143, 1977.
14. R. Landauer. Irreversibility and heat generation in the computing process. IBM J. Res. Dek., 5:183–191, 1961.
15. I. Lanese, C. A. Mezzina, A. Schmitt, and J. Stefani. Controlling reversibility in higher-order pi. In CONCUR 2011, pages 297–311, 2011.
16. I. Lanese, C. A. Mezzina, and J. Stefani. Reversing higher-order pi. In CONCUR 2010, pages 478–493, 2010.
17. I. Lanese, C. A. Mezzina, and J. Stefani. Reversibility in the higher-order π-calculus. Theor. Comput. Sci., 625:25–84, 2016.
18. R. Milner and D. Sangiorgi. Barbed bisimulation. In Proceedings of the 19th International Colloquium on Automata, Languages and Programming, ICALP ’92, pages 685–695, London, UK, UK, 1992. Springer-Verlag.
19. I. Phillips, I. Ulidowski, and S. Yuen. A reversible process calculus and the modelling of the ERK signalling pathway. In RC, volume 7581 of LNCS, pages 218–232. Springer, 2012.
20. I. C. C. Phillips and I. Ulidowski. Reversing algebraic process calculi. J. Log. Algebr. Program., 73(1-2):70–96, 2007.
21. A. Rensink and W. Vogler. Fair testing. Inf. and Comp., 205(2):125–198, 2007.
22. F. B. Schneider. Enforceable security policies. ACM Trans. Inf. Syst. Secur., 3(1):30–50, Feb. 2000.
23. G. Weikum and G. Vossen. Transactional information systems. Morgan Kaufmann Publishers Inc., 2001.
This appendix will not appear in the final version. The full paper with the appendix will be uploaded after submission to arXiv.

## A Omitted Definitions and Lemmas

### Definition A.1. $M$ is initial if $M \equiv \varepsilon : P$.

### Definition A.2 (Configuration free keys). The set of keys of a configuration $M$, written $\text{keys}(M)$, is inductively defined as follows:

\[
\begin{align*}
\text{keys}(\emptyset) &= \emptyset \\
\text{keys}(M || N) &= \text{keys}(M) \cup \text{keys}(N) \\
\text{keys}([k_0 ; P ; k]) &= \{ k_0 , k \} \cup \text{keys}(P) \\
\text{keys}(\sum_{i \in I} \alpha_i \langle \gamma_i \rangle . P_i) &= \bigcup_{i \in I} \text{keys}(P_i) \\
\text{keys}(\nu a . k) &= \{ k \} \\
\text{keys}(\text{rec} X (\gamma) . P) &= \text{keys}(P) \\
\text{keys}(X) &= \emptyset
\end{align*}
\]

### Lemma A.3. If $M \xrightarrow{(k)} M'$ and $M' \xrightarrow{(k)} M''$ then $M' = M''$.

**Proof.** By induction on $M$. \hfill ◀

### Lemma A.4.

1. If $M_1 \equiv N_1 \xrightarrow{(k)} N_2$ then there exists $M_2$ such that $M_1 \xrightarrow{(k)} M_2 \equiv N_2$.

2. If $M_1 \equiv N_1 \xrightarrow{(k)} N_2$ then there exists $M_2$ such that $M_1 \xrightarrow{(k)} M_2 \equiv N_2$.

**Proof.** By induction on the derivation of $M_1 \equiv N_1$, with case analysis on the last applied axiom. We consider some key cases.

1. $M_1 = N_1' || N_1 = N_1' || \emptyset$. By hypothesis we have that $N_1' || \emptyset \xrightarrow{(k)} N_2'' || \emptyset$, which by rule RLPAR implies that $N_1' \xrightarrow{(k)} N_2''$, and we can conclude by noticing that $N_2'' \equiv N_2' || \emptyset$.

2. $M_1 = M' || N'$ and $N_1 = N' || M'$. By hypothesis we have that $N' || M' \xrightarrow{(k)} N_1' || M_1$, with $M' \xrightarrow{(k)} M_1'$ and $N' \xrightarrow{(k)} N_1'$. Hence also $M' || N' \xrightarrow{(k)} M_1' || N_1'$ with $M_1' || N_1' \equiv N_1' || M_1'$, as desired.

3. $M_1 = k' . \nu a . P$ and $N_1 = \nu a . k' : P$. By hypothesis, we have that $N_1 \xrightarrow{(k)} N_2$. We have to distinguish two cases, either $k = k'$ or $k \neq k'$. In the first case we have that $\nu a . k' : P \xrightarrow{(k)} \nu a . \emptyset$ and $\nu a . \emptyset \xrightarrow{(k)} \emptyset$ and we can conclude by noticing that $\nu a . \emptyset \equiv \emptyset$.

The second case trivially holds.

4. $M_1 = k : (P || Q)$ and $N_1 = k : P || k : Q$. By hypothesis, we have that $N_1 \xrightarrow{(k)} N_2$.

We have to distinguish two cases, either $k = k'$ or $k \neq k'$. In the first case we have that $k : P || k : Q \xrightarrow{(k)} \emptyset || \emptyset$ since $k : P \xrightarrow{(k)} \emptyset$ and $k : Q \xrightarrow{(k)} \emptyset$. But also $k : (P || Q) \xrightarrow{(k)} \emptyset$ and we can conclude by noticing that $\emptyset \equiv \emptyset || \emptyset$.

2. By induction on the length of the derivation of $M_1 \equiv N_1$. The proof is similar to the previous case. \hfill ◀

### Lemma A.5. For any key $k$ and term $M$ such that $k \equiv M$, $M \xrightarrow{(k)} M$.

**Proof.** By structural induction on $M$. \hfill ◀
Lemma A.6 (Forward-Backward). If $D \vdash M \xrightarrow{(k)} D' \vdash M'$ then there exists $M''$ such that $D' \vdash M' \xleftarrow{(kn)} D \vdash M''$ and $M \equiv M''$.

Proof. By induction on $M \xrightarrow{(k)} M'$, using Lemma A.4 in the case of $\text{Reqv}$ and Lemma A.5 in the case of $\text{Rpar}$.

Lemma A.7. If $M \xrightarrow{(k)} M'$ then either

\begin{align*}
M \equiv & \nu_\tilde{a}. \langle (k_0: P \parallel N) \rangle & P \xrightarrow{\tau(k)} P' \\
M' \equiv & \nu_\tilde{a}. \langle (k: P' \parallel [k_0: P; k] \parallel N) \rangle & k \notin M, k_0, P, N
\end{align*}

or

\begin{align*}
M \equiv & \nu_\tilde{a}. \langle (k_1: P \parallel k_2: Q \parallel N) \rangle & P \xrightarrow{\tau(k)} P' \\
M' \equiv & \nu_\tilde{a}. \langle (k: P' \parallel k: Q' \parallel [k_1: P; k] \parallel [k_2: Q; k] \parallel N) \rangle & Q \xrightarrow{\tau(k)} Q' & k \notin M, k_1, P, k_2, Q, N
\end{align*}

for some $\tilde{a}, \tilde{k}, P, Q, P', Q', N$.

Proof. By induction on the transition $M \xrightarrow{(k)} M'$.

Lemma A.8. If $M \xrightarrow{(k)} M'$ then $\text{keys}(M') \subseteq \text{keys}(M) \cup \{k\}$.

Proof. By induction on the transition.

Lemma A.9. If $M \xrightarrow{(k)} M'$ then $\text{keys}(M') \subseteq \text{keys}(M) \setminus \{k\}$.

Proof. By induction on the transition.

Lemma A.10. If $M \equiv N$ then $\text{keys}(M) = \text{keys}(N)$.

Proof. By induction on the derivation $M \equiv N$.

Lemma A.11. Let $M \xrightarrow{(k)} M'$ and $N \in M'$ with $N = k: P$ or $[\mu; k]$, for some $k$, $\mu$, and $P$. Then it must be $N \in M$.

Proof. By induction on the transition.

Lemma A.12. Let $M \xrightarrow{(k)} M'$ and $M_1 \in M$ and $k \notin M_1$. Then $M_1 \in M'$.

Proof. By induction on the rollback transition.

Lemma A.13. If $\text{wf}(D \vdash M)$ and $D \vdash M \xrightarrow{(k)} D' \vdash M'$ then $D' \vdash M' \Rightarrow^* D \vdash M$.

Proof. By $\text{RLsys}$ from Figure 3 we have

$M \xrightarrow{(k_1)} M_1 \cdots \xrightarrow{(k_n)} M_n = M' \quad D = (k_1 \prec \ldots \prec k_n \prec D')$

By Definition 2.4:

$M \xrightarrow{(k_1)} M'_1 \cdots \xrightarrow{(k_n)} M'_n$

$M \xleftarrow{(k_1)} M'_1 \cdots \xleftarrow{(k_n)} M'_n$

By Lemma A.3, $M_1 = M'_1, \ldots, M_n = M'_n$, from which the result follows.
Corollary A.14. If \( \text{wf}(D \vdash M) \) and \( D \vdash M \leadsto D' \vdash M' \) then \( D' \vdash M' \).  

Keys in well-formed systems are nested according to the causality relation.

Lemma A.15 (Syntactic Causality). If \( \text{wf}(D \vdash M) \) then

1. Memory Causality: If \([\mu; l]\) \( \in M \) and \( k \in \text{keys}(\mu) \) then \( D \vdash l \bowtie^+ k \).
2. Process Causality: If \( l : P \in M \) and \( k \in \text{keys}(P) \) then \( D \vdash l \bowtie^+ k \).

Moreover, well-formed systems can roll back any memory, and the memories they contain are products of binary communication.

Lemma A.16 (Roll Back Any Memory). If \( \text{wf}(D \vdash M) \) and \([\mu; k]\) \( \in M \) then \( D \vdash M \leadsto_{(k)} D' \vdash M' \); for some \( D' \) and \( M' \).

Lemma A.17 (Binary Communication). If \( \text{wf}(D \vdash M) \) and \( M \equiv \nu a.(\mu_1; k) \parallel \mu_2; k \parallel N \) then for all \([\mu; l]\) \( \in N \), \( k \neq l \).

Lemma A.18 (Preservation of Well-Formedness). Let \( \text{wf}(D \vdash M) \); then:

1. If \( M \equiv N \) then \( \text{wf}(D \vdash N) \).
2. If \( M = \nu a.N \) then \( \text{wf}(D \vdash N) \).
3. If \( D \vdash M \leadsto_{(k)} D' \vdash M' \) then \( \text{wf}(D' \vdash M') \).
4. If \( D \vdash M \leadsto_{(k)} D' \vdash M' \) then \( \text{wf}(D' \vdash M') \).
5. If \( D \vdash M \leadsto D' \vdash M' \) then \( \text{wf}(D' \vdash M') \).

Proof. The first property follows from Lemmas A.4 and A.10. The second property follows by the definition of \text{keys} and rule RL\nu of Figure 3.

To prove the third property we derive \( D' = k \bowtie D \) by Rsys of Figure 2. We need to show that for some \( M'' \): \( \text{keys}(M'') \subseteq \text{keys}(D') \) and \( M' \leadsto_{(k)} M'' \) and \( M'' \leadsto_{(k)} M' \) and \( \text{wf}(D \vdash M'') \). The first follows from Lemma A.8. The next two follow from Lemma A.6 and rule Reqv of Figure 2; the last follows from the first property of the lemma.

The fourth property follows by definition of well-formedness and rule Bw of Figure 3, using an induction on the number of individual rollback transitions. The last property follows from the previous two.

Lemma A.19. Let \( D \vdash M \leadsto_{(k)} M' \). Then \( D = D' \) and \( M \equiv M' \).

Proof. By Lemmas 2.2 and A.6.

Lemma A.20. Let \( D \vdash M \leadsto_{(k)} D' \vdash M' \) and \( k \ngeq l \). Then there exists \( M'' \equiv M' \) such that \( D \vdash M \leadsto_{(l)} D' \vdash M'' \).

Proof. We have \( D \vdash M \leadsto_{(k)} D_1 \vdash M_1 \leadsto_{(k)} D' \vdash M' \). By rule Rsys of Figure 2 we have \( D_1 = k \bowtie D \). By Lemma A.6, there exists \( M_2 \equiv M \) such that \( D_1 \vdash M_2 \leadsto_{(k)} D \vdash M_2 \).

By rule RLsys of Figure 3 and \( D_1 \vdash M_1 \leadsto_{(l)} D' \vdash M' \) we get a sequence of keys \( \tilde{k} \) such that \( D_1 = \tilde{k} \bowtie l \bowtie D' \) and \( M_1 \leadsto_{(k)} D \). Because \( D_1 = k \bowtie D \) and \( k \ngeq l \), the sequence \( \tilde{k} \) must have at least one element, \( k \). Thus there exists \( M_2' \) and \( \tilde{k}' \) such that \( D_1 = k \bowtie \tilde{k}' \bowtie l \bowtie D' \) and \( M_1 \vdash_{(k)} M_2' \leadsto_{(l)} D' \vdash M' \). From Lemma 2.2, \( M_2' = M_2 \equiv M \). By repeated applications of Lemma A.4 (\( M_2' \equiv M \)) we get \( M \leadsto_{(k)} M'' \), for some \( M'' \equiv M' \). The result follows from rule RLsys.
Corollary A.21. Let $D ⊢ M \xrightarrow{\langle \tilde{k} \rangle} D' \vdash M'$ and $\tilde{k} \not\approx l$. Then there exists $M'' \equiv M'$ such that $D \vdash M \xrightarrow{\langle \tilde{k} \rangle} D' \vdash M''$.

Lemma A.22. Let $D \vdash M \xrightarrow{\langle \tilde{k} \rangle} D' \vdash M'$. Then one of the following holds:

1. $D \vdash M \xrightarrow{\langle \tilde{k} \rangle} D' \vdash M'' \equiv M'$ and $\tilde{k} \not\approx l$, for some $M''$, or
2. $D \vdash M \xrightarrow{\langle \tilde{k} \rangle} D' \vdash M'' \equiv M'$ and $\tilde{k} = \tilde{k}', \tilde{k}''$, for some $M''$, $\tilde{k}'$, $\tilde{k}''$.

Proof. By cases on $\tilde{k}$.

If $\tilde{k} \not\approx l$ then the first property follows from Corollary A.21.

Otherwise there exist $\tilde{k}', \tilde{k}''$ such that $k = \tilde{k}', l, \tilde{k}''$ and $\tilde{k}'' \not\approx l$. In this case the result follows from Corollary A.21 and Lemma A.19.

Lemma A.23. Let $D \vdash M \xrightarrow{\langle \tilde{k} \rangle} D' \vdash M'$. Then $D \vdash M \xrightarrow{\langle \tilde{k} \rangle} D' \vdash M'$.

Proof. From rule RLsys of Figure 3 we get $\tilde{k}, \tilde{l}$ such that $D = \tilde{k} \not\approx l \not\approx D'$ and $M \xrightarrow{\langle \tilde{k} \rangle} D' \vdash M'$. The lemma is proven by an application of RLsys.

Lemma A.24. Let $\text{wf}(\tilde{k} \not\approx D \vdash M_1)$ and $M_1 \xleftarrow{\tilde{k}} M_2$; then $D \vdash M_2 \rightarrow^* \tilde{k} \not\approx D \vdash M_1$.

Proof. By induction on the number of $\tilde{k}$. The base case, $\tilde{k} = \emptyset$ banally holds. In the inductive case we have

$$k_n \prec \ldots \prec k_2 \prec k_1 \prec D \vdash M_1 \xleftarrow{\langle k_n, \ldots, k_2 \rangle} k_1 \prec D \vdash M_2 \xleftarrow{\langle k_1 \rangle} D \vdash M_1$$

By applying inductive hypothesis on the reduction $\langle k_n, \ldots, k_2 \rangle$, we obtain

$$k_n \prec \ldots \prec k_2 \prec k_1 \prec D \vdash M_1 \xleftarrow{\langle k_n, \ldots, k_2 \rangle} k_1 \prec D \vdash M_2 \rightarrow^* \tilde{k} \not\approx D \vdash M$$

By Lemma A.18 we have that $\text{wf}(k_1 \not\approx D \vdash M_2)$, and by definition of $\text{wf}(\cdot)$ we have that $k_1 \not\approx D \vdash M_2 \xleftarrow{\langle k_1 \rangle} D \vdash M_1 \rightarrow k_1 \prec D \vdash M_2$ and we can conclude by noticing that

$$k_n \prec \ldots \prec k_2 \prec k_1 \prec D \vdash M_1 \xleftarrow{\langle k_n, \ldots, k_2 \rangle} k_1 \prec D \vdash M_2 \xleftarrow{\langle k_1 \rangle} D \vdash M \rightarrow k_1 \prec D \vdash M_2 \rightarrow^* \tilde{k} \not\approx D \vdash M$$

Corollary A.25. Let $\text{wf}(D_1 \vdash M_1)$ and $D_1 \vdash M_1 \xleftarrow{\tilde{k}} D_2 \vdash M_2$; then $D_2 \vdash M_2 \rightarrow^* D_1 \vdash M_1$.

Lemma A.26. If $D_1 \vdash M_1 \rightarrow^* D_2 \prec D_1 \vdash M_2 \xleftarrow{\tilde{k}} D_3 \vdash M_3$ and $l \in D_2$ then $D_1 \vdash M_1 \rightarrow^* D_3 \vdash M_3$.

Proof. Let $D_2 = \tilde{k_1} \prec l \prec \tilde{k_2}$. We proceed by induction on the size of $\tilde{k_2}$. In the base case, $\tilde{k_2} = \emptyset$ we have that $D_1 \vdash M_1 \xleftarrow{\tilde{k_1}} \tilde{k_1} \prec l \vdash M_2$. By Lemma 2.3 we have that there exists $M'_1 \equiv M_1$ such that $\tilde{k_1} \prec l \vdash M_2 \xleftarrow{\tilde{k_1}} D_1 \vdash M'_1$. Moreover since rollback is deterministic, and since $\tilde{k_1} \prec l \vdash M_2 \xleftarrow{\tilde{k_1}} D_3 \vdash M_3$ by Lemma 2.2 we have that $D_1 = D_3$ and $M'_1 = M_3$. We can conclude since $\rightarrow^*$ is closed under $\equiv$.

In the inductive case, let $\tilde{k_2} = \tilde{k_1} \prec \tilde{k_2'}$, we have that

$$D_1 \vdash M_1 \xleftarrow{\tilde{k_2'}} \tilde{k'_1} \prec D_1 \vdash M' \xleftarrow{\tilde{k_2}} \tilde{k_3} \prec \tilde{k_2'} \prec D_1 \vdash M'' \xleftarrow{\tilde{k_1}} D_2 \prec D_1 \vdash M_2 \xleftarrow{\tilde{k_1}} D_3 \vdash M_3$$

By inductive hypothesis we have that $\tilde{k'_2} \prec D_1 \vdash M' \xleftarrow{\tilde{k'_2}} \tilde{k'_2} \prec D_1 \vdash M' \rightarrow^* D_3 \vdash M_3$, and since $D_1 \vdash M_1 \xleftarrow{\tilde{k_2'}} \tilde{k_2'} \prec D_1 \vdash M'$ we have that $D_1 \vdash M_1 \rightarrow^* D_3 \vdash M_3$, as desired.
Lemma A.27. Let $D_1 \vdash M_1 \rightarrow^* D_2 \vdash M_2 \leftrightarrow D_3 \vdash M_3$. One of the following is true:
1. $D_3$ is smaller than $D_1$: there exists non-empty $D_4$ such that $D_1 = D_4 \prec D_3$, or
2. $D_3$ is larger than or equal to $D_1$: $D_1 \vdash M_1 \rightarrow^* D_3 \vdash M_3$.

Proof. By induction on the number $n$ of steps $D_1 \vdash M_1 \rightarrow^* D_2 \vdash M_2$. In the base case we have that $D_1 = D_2$ and $M_1 = M_2$. Moreover, since $D_2 \vdash M_2 \leftrightarrow D_3 \vdash M_3$, this implies that there exists a key $l \in D_2$ such that $D_2 = D'_l \prec l \prec D_3$ such that $D_2 \vdash M_2 \overset{(l)}{\rightarrow} D_3 \vdash M_3$.

And we have that 1 holds by setting $D_4 = D'_l \prec l$.

In the inductive case, we have that:

$$D_1 \vdash M_1 \overset{(k_1)}{\rightarrow} D'_1 \vdash M'_1 \overset{(k)}{\rightarrow} D_2 \vdash M_2 \leftrightarrow D_3 \vdash M_3$$

with $D_2 = \tilde{k} \prec l \prec D_1$ and $D_1 = l_1 \prec D'_1$. By inductive hypothesis we have that if 1 holds, then $D'_1 = D'_l \prec D_3$ with $D'_l$ non empty. And the property still holds by setting $D_4 = k_1 \prec D'_l$, as desired. By inductive hypothesis we have that if 2 holds, then $D_3$ is larger than or equal to $D'_1$: $D'_1 \vdash M'_1 \rightarrow^* D_3 \vdash M_3$. And the property still holds since $D_1 \vdash M_1 \overset{(k)}{\rightarrow} D'_1 \vdash M'_1 \rightarrow^* D_3 \vdash M_3$.

Lemma A.28. If $D \vdash M \rightarrow^* D \vdash N$ then $N = M$.

Definition A.29 (Well-Formed Configuration). $D \vdash M$ is well-formed ($\text{wf}(D \vdash M)$) when
1. $\text{keys}(M) \subseteq \text{keys}(D)$ and
2. if $D = k \prec D'$ then there exist $\alpha$ and $M'$ such that $\alpha(k) \cap M' \rightarrow \alpha(k) \vdash M \vdash M'$ and $D' \vdash M' \rightarrow \alpha(k) \vdash M \vdash \text{wf}(D' \vdash M')$.

Well-formedness in the reduction semantics implies well-formedness in the LTS, but not the converse.

Lemma A.30. If $\text{wf}(D \vdash M)$ then $\text{wf}(D \vdash M)$.

Lemma A.31. Let $\text{wf}(D \vdash M)$ and $\text{wf}(D \vdash N)$; then:
1. $\text{wf}(D \vdash \nu a.M)$ and $\text{wf}(D \vdash M \parallel N)$.
2. If $M = \nu a. M'$ then $\text{wf}(D \vdash M')$.
3. If $M \equiv M_1 \parallel M_2$ then $\text{wf}(D \vdash M_1)$ and $\text{wf}(D \vdash M_2)$.
4. If $M \equiv M'$ then $\text{wf}(D \vdash M')$.
5. If $D \vdash M \rightarrow \alpha(k) \vdash M'$ then $\text{wf}(D' \vdash M')$.

Lemma A.32 (($\equiv$) is a Strong Bisimulation). If $M \equiv N$ and $D \vdash M \overset{\lambda}{\rightarrow} D' \vdash M'$ then there exists $N' \equiv M'$ such that $D \vdash N \overset{\lambda}{\rightarrow} D' \vdash N'$.

Lemma A.33. Let $D \vdash M \overset{\alpha(k)}{\rightarrow} D' \vdash M'$; then either
1. $\alpha = \tau$ and
   $$M \equiv \nu \alpha. (\nu l : P \parallel N) \quad M' \equiv \nu \alpha. ([\nu l : P \parallel k : P' \parallel N]$$
   with $P \overset{\tau(k)}{\rightarrow} P'$ and $D' = k \prec D$ otherwise;
2. $\alpha = \tau$ and
   $$M \equiv \nu \alpha. (\nu l_1 : P \parallel l_2 : Q \parallel N) \quad M' \equiv \nu \alpha. ([\nu l_1 : P \parallel k : l_2 : Q \parallel k : P' \parallel Q']$$
   with $P \overset{\alpha(k)}{\rightarrow} P'$, $Q \overset{\pi(k)}{\rightarrow} Q'$ and $D' = k \prec D$ otherwise;
3. \( \alpha \neq \tau \) and
\[
M \equiv \nu \tilde{a}. (l : P \parallel N) \quad M' \equiv \nu \tilde{a}. ([l : P; k] \parallel k : P' \parallel N)
\]
with \( P \xleftarrow{\alpha(k)} P' \) and \( D' = k < D \).

\begin{itemize}
  \item \textbf{Corollary A.34.} \( D \vdash M \rightarrow^* D' \vdash M' \) iff \( D \models M \xleftarrow{\alpha(k)} D' \models M'' \) with \( M'' \equiv M' \).
  \item \textbf{Lemma A.35.} If \( \mathcal{C} \xrightarrow{\lambda} \mathcal{C}' \) then \( \lambda \notin \mathcal{C} \).
  \item \textbf{Lemma A.36.} If \( \mathcal{C} \xrightarrow{\lambda} \mathcal{C} ' \) and \( p \) is a name permutation then \( p \mathcal{C} \xrightarrow{p} p \mathcal{C}' \).
  \item \textbf{Lemma A.37 (Transition Weakening/Strengthening).} Let \( D \models M \xleftarrow{\alpha(k)} D' \models N \).
    1. If \( D = D_1 < l < D_2 \) then \( D_1 < D_2 \models M \xleftarrow{\alpha(k)} k < D_1 < D_2 \models N \).
    2. If \( D = D_1 < D_2 \) and \( l \notin D, k \) then \( D_1 < D_2 \models M \xleftarrow{\alpha(k)} k < D_1 < l \models N \).
  \item \textbf{Lemma A.38 (Canonical Traces are Typed).} Let \( t \) be a canonical trace with \( t \notin D \). There exists \( D' \) such that \( (D \vdash t \vdash D') \).
  \item \textbf{Lemma A.39 (Typed Traces are Canonical).} If \( (D \vdash t \vdash D') \) then \( t \) is canonical.
  \item \textbf{Lemma A.40.} If \( (D \vdash t \vdash D') \) then \( t \notin D \) and \( \text{keys}(D') = \text{keys}(D) \cup \text{keys}(t) \).
\end{itemize}

Our characterisations of the safety and liveness preorders are based on zipping and un-zipping typed traces. The definition of zipping, shown in Figure 6, is straightforward. Moreover, trace typing is invariant to key permutation. LTS traces are canonical by construction and thus they are typable.

\begin{itemize}
  \item \textbf{Lemma A.41.} If \( D_1 \parallel D_2 \rightarrow D \), then \( \text{keys}(D_1) \subseteq \text{keys}(D) \) and \( \text{keys}(D_2) \subseteq \text{keys}(D) \).
  \item \textbf{Lemma A.42 (Zipping Inversion).} If \( D_1 \parallel D_2 \rightarrow (k < D) \) then one of the following holds:
    1. \( (k < D_1) \parallel D_2 \rightarrow k < D \),
    2. \( D_1 \parallel (k < D_2) \rightarrow k < D \), or
    3. \( (k < D_1) \parallel (k < D_2) \rightarrow k < D \).
  \item \textbf{Lemma A.43.} If \( (D \vdash t_1 \vdash D'_1) \parallel (D_2 \vdash t_2 \vdash D'_2) \rightarrow (D \vdash t \vdash D'_1) \) then \( D_1 \parallel D_2 \rightarrow D \).
  \item \textbf{Lemma A.44 (Key Permutation).} Let \( p \) be a key permutation. Then
    1. If \( (D \vdash t \vdash D') \) then \( (pD \vdash pt \vdash pD') \).
    2. If \( \mathcal{C} \xrightarrow{\lambda} \mathcal{C}' \) then \( p \mathcal{C} \xrightarrow{p} p \mathcal{C}' \).
  \item \textbf{Lemma A.45.} Let \( D \models M \xrightarrow{l} D' \models M' \). Then:
    1. \( (D \vdash t \vdash D') \).
    2. If \( (D \vdash t \vdash D'' \) then \( D' = D'' \).
\end{itemize}

\textbf{Proof.} We prove the two properties by induction on the length \( n \) of the trace \( t \).

1. The base case \( n = 0 \) and \( t = \epsilon \) trivially holds, since \( D = D' \) and \( (D \vdash \epsilon \vdash D) \). In the inductive case, we have that \( t = \alpha(k), t' \), with \( D \models M \xleftarrow{\alpha(k)} D_1 \models M_1 \xrightarrow{l'} D' \models M' \).

According to Definition 5.2, in order to show \( (D \vdash \alpha(k), t' \vdash D') \) we have to prove \( (k < D \vdash t' \vdash D') \). By inductive hypothesis on the derivation \( D_1 \models M_1 \xrightarrow{l'} D' \models M' \) we have that \( (D_1 \vdash t' \vdash D') \) and since \( D \models M \xleftarrow{\alpha(k)} D_1 \models M_1 \), by Lemma A.33 we have that \( D_1 = k < D \) and we can conclude \( (k < D \vdash t' \vdash D') \).
2. The base case $n = 0$ and $t = \epsilon$ trivially holds. In the inductive case, we have $t = \alpha(k), t'$ and $D \models M \xrightarrow{\alpha(k)} D_1 \models M_1 \xrightarrow{t'} D'' \models M'$. By hypothesis we have that $(D \models \alpha(k), t' \triangleright D')$ and we can derive $(k \vartriangleleft D \triangleright t' \triangleright D')$. By Lemma A.33 we have that $D \models M \xrightarrow{\alpha(k)} k \vartriangleleft D \models M_1$, and since $k \vartriangleleft D \models M_1 \xrightarrow{t'} D'' \models M'$, we can apply the inductive hypothesis and obtain that $D' = D''$, as desired.

Lemma A.46 (Unzipping Transition). Let $D \models M \parallel N \xrightarrow{\alpha(k)} D' \models R' \parallel D_2 \rightarrow D$ and $wf(D_1 \models M)$ and $wf(D_2 \models N)$. Then one of the following is true
1. $D_1 \models M \xrightarrow{\alpha(k)} D'_1 \models M' \parallel D_2 \rightarrow D'$ and $R' = M' \parallel N$.
2. $D_2 \models N \xrightarrow{\alpha(k)} D'_2 \models N' \parallel D_1 \rightarrow D'$ and $R' = M \parallel N'$.
3. $D_1 \models M \xrightarrow{\alpha(k)} D'_1 \models M' \parallel D_2 \rightarrow D'$ and $R' = M' \parallel N'$ and $\alpha = \tau$.

Two typed complementary traces can be zipped provided the keys annotating their $\tau$ actions do not overlap. Their zipped trace will contain only $\tau$ actions.

Lemma A.47. Let $(D_1 \vdash t_1 \triangleright D'_1)$ and $(D_2 \vdash t_2 \triangleright D'_2)$ and $D_1 \parallel D_2 \rightarrow D_3$ and $obs(t_1) = t'_1$ and $obs(t_2) = t'_2$ and $t'_1 = t'_2$ and keys($t_1$) \ keys($t'_1$) \ keys($t'_2$) \ keys($t_2$). Then there exists $t_3, D'_3$ such that $(D_1 \vdash t_1 \triangleright D'_1) \parallel (D_2 \vdash t_2 \triangleright D'_2) \rightarrow (D_3 \vdash t_3 \triangleright D'_3)$ and $D'_1 \parallel D'_2 \rightarrow D'_3$ and $obs(t_3) = \epsilon$.

The set $Tr(C)$ is closed under the permutation of keys fresh from $C$.

Lemma A.48. If $t \in Tr(C)$ and $p$ is a permutation such that $p \not\sqsubseteq C$ then $pt \in Tr(C)$.

Theorem A.49 (Soundness). For any initial systems $M$ and $N$, $M \sqsubseteq_{tr} \sqsubseteq_{tr} N$ implies $N \sqsubseteq_{safe} M$.

Proof. We assume $M \sqsubseteq_{tr} N$, for initial systems $M$ and $N$. According to Definition 3.2, we need to show that for all tests $T$, $\epsilon \vdash M \parallel \epsilon:T\downarrow_{\omega}$ implies $\epsilon \vdash N \parallel \epsilon:T\downarrow_{\omega}$.

Let $\epsilon \vdash M \parallel \epsilon:T\downarrow_{\omega}$, for test $T$. From Definition 3.1, let $\epsilon \vdash M \parallel \epsilon:T \rightarrow^* D \downarrow_{\omega}$, for some system $O$. From Corollary 4.2, the same system is reached with only forward reductions: $\epsilon \vdash M \parallel \epsilon:T \rightarrow^* D \downarrow_{\omega}$. From Corollary A.34 we get $\epsilon \vdash M \parallel \epsilon:T \downarrow_{\omega} D \models O' \equiv O$ for $t = k(\tau)$.

Since $\epsilon \vdash \epsilon \rightarrow \epsilon$ we unzip the trace $t$ by applying Proposition 5.3 and obtain:

\[\epsilon \vdash M \Downarrow_{t_1} D_1 \models M_1 \quad (\epsilon \vdash t_1 \triangleright D_1) \parallel (\epsilon \vdash t_2 \triangleright D_2) \rightarrow (\epsilon \vdash t \triangleright D)\]

\[\epsilon \vdash \epsilon:T \downarrow_{t_2} D_2 \models T_2 \quad O' \downarrow_{\omega} \equiv M_1 \parallel T_2 \quad obs(t_1) = \overline{obs(t_2)}\]

Moreover, since $O'\downarrow_{\omega}$ and $\omega$ is only present in $T$, $T_2\downarrow_{\omega}$. By Definition 6.1, $obs(t_1) \in Tr(\epsilon \models M)$, and by $M \sqsubseteq_{tr} N$, $obs(t_1) \in Tr(\epsilon \models N)$. Therefore, there exists trace $t_3$ with $obs(t_1) = obs(t_3)$ such that $\epsilon \models N \Downarrow_{t_3} D_3 \models N_3$ and, by Lemma A.45, $(\epsilon \vdash t_3 \triangleright D_3)$. The keys in the observable part of $t_3$ are the same as in those of $t_1$. However, $t_3$ may contain $\tau$-actions whose keys overlap with the keys of $\tau$-actions in $t_2$. For this reason we invent a permutation $p$ which maps all $\tau$ keys of $t_3$ to fresh keys. Because $t_3 \not\sqsubseteq (\epsilon \models N)$ (from Lemma A.40, $wf(\epsilon \models N)$) we get $p \not\sqsubseteq (\epsilon \models N)$. Thus from Lemma A.44, $\epsilon \models N \Downarrow_{t_3} D_3 \models N_3$, where $t_3 = p t_3, D_3 = p D_3$ and $N_3 = p N_3$. From Lemma A.45, $(\epsilon \vdash t_4 \triangleright D_4)$. Moreover, because $p \not\sqsubseteq obs(t_3)$, we have $obs(t_1) = obs(t_4) = obs(t_2)$, and by construction of $p$ and Lemma A.40: keys($t_4$) \ keys(obs($t_4$)) \ keys($t_2$) = keys($D_2$).
keys(t_2) \setminus \text{keys(obs(t_2))} \uplus \text{keys(t_4)} = \text{keys(D_4)}. Thus we can apply Lemma A.47 and obtain (\epsilon \vdash t_4 \triangleright D_4) \parallel (\epsilon \vdash t_2 \triangleright D_2) \rightarrow (\epsilon \vdash t' \triangleright D') with \text{obs}(t') = \epsilon. By the Zipping proposition (Proposition 5.4) we get that for some O', \epsilon \vdash N \parallel \epsilon : T \xrightarrow{T'} D' \models O' with O' \models N \parallel T_2.

Therefore, by Corollary A.34, \epsilon \vdash N \parallel \epsilon : T \rightarrow^* D' \models O'' \models N \parallel T_2 \downarrow_w. Thus 
\epsilon \vdash N \parallel \epsilon : T \downarrow_w.

\textbf{Theorem A.50 (Completeness).} For any initial systems M and N, M \subseteq_{\text{safe}} N implies N \subseteq_{\text{live}} M.

\textbf{Proof.} Let M \subseteq_{\text{safe}} N for initial systems M and N. We must show that Tr(N) \subseteq Tr(M).

We take observable t \in Tr(N). By definition, \epsilon \vdash N \overset{\omega}{\rightarrow} D_1 \models N_1, for some D_1, T_1, t_1 with \text{obs}(t_1) = t. From Lemma 6.2, \epsilon \vdash \text{Test}^k(t) \xrightarrow{T} D_2 \models T_2 \downarrow_w, for some D_2, T_2. By Lemma A.45, (\epsilon \vdash t_1 \triangleright D_1) and (\epsilon \vdash T \triangleright D_2). Moreover, keys(t_1) \setminus \text{keys(obs(t_1))} \uplus \text{keys(obs(t_1))} = \text{keys(t)} = \text{keys(T)} and by Lemma A.40, keys(T) = keys(D_2). Because t is observable, keys(T) \setminus \text{keys(obs(T))} = \emptyset \parallel D_1. Thus we can apply Lemma A.47 and obtain (\epsilon \vdash t_1 \triangleright D_1) \parallel (\epsilon \vdash T \triangleright D_2) \rightarrow (\epsilon \vdash t_0 \triangleright D) for some t_0, D with \text{obs}(t_0) = \epsilon. By the Zipping proposition (Proposition 5.4) we get that for some O, \epsilon \vdash N \parallel \epsilon : \text{Test}^k(t) \xrightarrow{T} D \models O \models N \parallel T_2. Therefore, by Corollary A.34, \epsilon \vdash N \parallel \epsilon : \text{Test}^k(t) \rightarrow^* D' \equiv N \parallel T_2 \downarrow_w, and \epsilon \vdash \epsilon : \text{Test}^k(t) \downarrow_w.

By M \subseteq_{\text{safe}} N, we get \epsilon \vdash M \parallel \epsilon : \text{Test}^k(t) \downarrow_w, thus \epsilon \vdash M \parallel \epsilon : \text{Test}^k(t) \rightarrow^* D' \models O'' \downarrow_w. From Corollary 4.2, \epsilon \vdash M \parallel \epsilon : \text{Test}^k(t) \rightarrow^* D' \models O'' \downarrow_w. Therefore, by Corollary A.34, \epsilon \vdash M \parallel \epsilon : \text{Test}^k(t) \xrightarrow{T} D' \equiv O'' \downarrow_w, for some t_0 and O'' such that \text{obs}(t_0) = \epsilon. By the Unzipping Proposition (Proposition 5.3) we obtain:

\[\epsilon \vdash \epsilon : \text{Test}^k(t) \xrightarrow{T} D_4 \models M_4 \parallel (\epsilon \vdash t_3 \triangleright D_3) \parallel (\epsilon \vdash t_4 \triangleright D_4) \rightarrow (\epsilon \vdash t_0 \triangleright D')\]

\[\epsilon \vdash \epsilon : \text{Test}^k(t) \xrightarrow{T} D_4 \models T_4 \parallel O'' \equiv M_4 \parallel T_4 \parallel \text{obs}(t_3) = \text{obs}(t_4)\]

Therefore \text{obs}(t_4) \in Tr(M). From Lemma 6.2, there exists permutation p such that \text{pt} = t_4.

Because t is observable, \text{obs}(t_4) = pt, and by Lemma A.48, t \in Tr(M) (note \text{ppt} = t).

\textbf{Lemma A.51.} Let (t; V; W) be a refusal; then:

1. For all k \not\parallel t, there exists D such that \epsilon \vdash \text{Test}^k(t; V; W) \xrightarrow{T,\tau(k)} D \models \text{Test}^k(t; V; W).

2. For all t \in V and (k \not\parallel t), there exists T such that D \models \text{Test}^k(t; V; W) \xrightarrow{T,\tau(k)} D' \models T.

3. If D \models \text{Test}^k(t; V; W) \xrightarrow{\alpha(k);t} D' \models T then \alpha = \tau and \text{V} \in V and D' \models T \downarrow_{\text{ref}(\alpha)}.

4. \text{Test}^k(t; V; W) \downarrow_w.

5. For all t \in V and D \not\parallel t, there exists T such that D \models \text{Test}^k(t; V; W) \xrightarrow{T} D' \models T \downarrow_w.

6. If t \in W and D \models \text{Test}^k(t; V; W) \xrightarrow{T} D' \models T then T \downarrow_w.

7. If D \models \text{Test}^k(t; V; W) \rightarrow D' \models T \downarrow_w then T \in W.

Refusal sets are closed under subset of their second and third elements.

\textbf{Lemma A.52 (Subset and Key Permutation Closure of Ref).} Let (t; V; W) \in \text{Ref}(M) and (t'; V'; W') is a refusal such that V' \subseteq V and W' \subseteq W and t = pt' (p a key permutation). Then (t'; V'; W') \in \text{Ref}(M).

\textbf{Theorem A.53 (Soundness).} For initial systems, M \subseteq_{\text{ref}} N implies N \subseteq_{\text{live}} M.
**Proof.** Let $M \subseteq_{\text{ref}} N$ and $T$ be a liveness test. We need to show that $N \not\subseteq_{\text{ref}} M$ implies $M \not\subseteq_{\text{ref}} T$ (Definition 3.4). We prove the contra-positive: $M \not\subseteq_{\text{ref}} T$ implies $N \not\subseteq_{\text{ref}} T$. Let $M \not\subseteq_{\text{ref}} T$. By Definition 3.3, there exists $D_0 \vdash O_0$ such that

$$\varepsilon \vdash M \parallel \varepsilon;T \Rightarrow^* D_0 \vdash O_0 \quad \text{and} \quad \forall(D' \Rightarrow O'). \ D_0 \vdash O_0 \Rightarrow^* D' \Rightarrow O'$$

implies $D' \Rightarrow O'$. From Corollary 4.2, the same system is reached with only forward reductions:

$$\varepsilon \vdash M \parallel \varepsilon;T \Rightarrow^* D_0 \vdash O_0 \quad \text{and} \quad \forall(D' \Rightarrow O'). \ D_0 \vdash O_0 \Rightarrow^* D' \Rightarrow O'$$

implies $D' \Rightarrow O'$. Note that from $D_0 \vdash O_0$ we can reach systems with forward or rollback transitions. However, we apply Lemma 4.4 and get $D \vdash O$ such that

$$\varepsilon \vdash M \parallel \varepsilon;T \Rightarrow D \vdash O' \Rightarrow O \quad \forall(D' \Rightarrow O'). \ D \vdash O \Rightarrow D' \Rightarrow O'$$

implies $D' \Rightarrow O'$. Using Corollary A.34 we derive

$$\forall't, (D' \Rightarrow O'). \ obs(t') = \varepsilon \quad \text{and} \quad D \vdash O \Rightarrow D' \Rightarrow O'$$

implies $D' \Rightarrow O'$. Note that (3) follows from (1). Since $\varepsilon \parallel \varepsilon = \varepsilon$ we unzip the trace $t$ by applying Proposition 5.3 and obtain:

$$\varepsilon \vdash M \Rightarrow D_1 \vdash M_1 \quad \text{(with) } t \vdash t_1 \Rightarrow D_1 \Rightarrow (\varepsilon \vdash t_2 \Rightarrow D_2)$$

implies $\varepsilon \vdash T \Rightarrow D_2 \vdash T_2$.

We take

$$V = \{ \text{obs}(t) \mid \exists C. \ D_2 \vdash T_2 \Rightarrow C \} \quad W = \{ \text{obs}(t) \mid \exists C. \ D_2 \vdash T_2 \Rightarrow C \}$$

and argue that $(\text{obs}(t_1); V; W) \in \text{Ref}(M)$ according to Definition 7.6. Because of (4) and

$$V \cap \text{Roll}(D_1 \vdash M_1) = W \cap \text{Tr}(D_1 \vdash M_1) = \emptyset$$

and (5), it suffices to show that $V \cap \text{Roll}(D_1 \vdash M_1) = W \cap \text{Tr}(D_1 \vdash M_1) = \emptyset$. 

By $M \subseteq_{\text{ref}} N$, we get $(\text{obs}(t_1); V; W) \in \text{Ref}(N)$. Using Lemmas A.40, A.40, A.44, A.45 and A.47 and Proposition 5.4 we derive that there exist $t_3, D_3, T_3, t', D', O''$ and $O'''$ such that $V \cap \text{Roll}(D_3 \vdash N_3)$ and $\varepsilon \vdash N \Rightarrow D_3 \vdash N_3$ and $\text{obs}(t_3) = \text{obs}(t_1)$ and $(\varepsilon \vdash t_3 \Rightarrow D_3) \Rightarrow (\varepsilon \vdash t_2 \Rightarrow D_2)$ and $\text{obs}(t') = \varepsilon$ and $\varepsilon \vdash N \parallel \varepsilon;T \Rightarrow D' \Rightarrow O'' \Rightarrow O''' \Rightarrow O'$. This contradicts (3). Thus it is necessary $V \cap \text{Roll}(D_1 \vdash M_1) = \emptyset$. 

We take $W \cap \text{Tr}(D_1 \vdash T_1) = \emptyset$. With a similar argument, this is necessary for (2) and (3) to hold.

By $M \subseteq_{\text{ref}} N$, we get $(\text{obs}(t_1); V; W) \in \text{Ref}(N)$. Using Lemmas A.40, A.40, A.44, A.45 and A.47 and Proposition 5.4 we derive that there exist $t_3, D_3, N_3, t', D', O''$ and $O'''$ such that $V \cap \text{Roll}(D_3 \vdash N_3)$ and $\varepsilon \vdash N \Rightarrow D_3 \vdash N_3$ and $\text{obs}(t_3) = \text{obs}(t_1)$ and $(\varepsilon \vdash t_3 \Rightarrow D_3) \Rightarrow (\varepsilon \vdash t_2 \Rightarrow D_2)$ and $\text{obs}(t') = \varepsilon$ and $\varepsilon \vdash N \parallel \varepsilon;T \Rightarrow D' \Rightarrow O'' \Rightarrow O''' \Rightarrow O'$.
It remains to establish that $D' \vdash N_3 \parallel T_2 \neq \omega$.

We first show by contradiction that

$$D' \vdash N_3 \parallel T_2 \sim^* D'' \vdash O(4)$$ (6)

Suppose that $D' \vdash N_3 \parallel T_2 \sim^* D' \vdash O(4) \sim^* D'' \vdash O(5)$ is the smallest trace for which (6) does not hold. Then $D' \vdash N_3 \parallel T_2 \vdash^* D' \vdash O(4) \vdash (l) D'' \vdash O(5)$ and $l \in D'$. Thus $D' \vdash N_3 \parallel T_2 \vdash^* D' \vdash O(4) \downarrow_{\operatorname{rl}(1)}$.

By Corollary A.34, $D' \models N_3 \parallel T_2 \vdash (\tau(k)) D' \vdash O(6) \downarrow_{\operatorname{rl}(1)}$, for some $O(6) \equiv O(4)$. From Proposition 5.3 we obtain:

$$D_3 \models N_3 \vdash l \to C_4 \quad D_2 \models T_2 \vdash l \to C_5 \quad \text{obs}(t_4) = \text{obs}(t_5)$$

and either $C_{4 \downarrow_{\operatorname{rl}(1)(D_4)}}$ or $C_{5 \downarrow_{\operatorname{rl}(1)(D_2)}}$. If it is the former, we observe that $\text{obs}_4 \in V$ by construction of $V$ and thus $V \cap \text{Roll}(D_3 \models N_3) \neq \emptyset$, which is a contradiction. If it is the latter, we observe that in this case $\text{obs}_4 \in W$ by construction of $W$ and thus $Q \cap \text{Tr}(D_3 \models N_3) \neq \emptyset$, which is again a contradiction. Therefore (6) must be true.

Due to (6) it remains to show that for all $D(7)$ and $O(7)$:

$$D' \vdash N_3 \parallel T_2 \vdash^* D(7) \vdash O(7) \quad (7)$$

Again we prove this by contradiction. Assume $D' \vdash N_3 \parallel T_2 \vdash^* D(7) \vdash O(7) \downarrow_{\omega}$. By Corollary A.34, $D' \models N_3 \parallel T_2 \vdash (\tau(k)) D(7) \vdash O(8) \downarrow_{\omega}$, for some $O(8) \equiv O(6)$. From Proposition 5.3:

$$D_3 \models N_3 \vdash l \to C_6 \quad D_2 \models T_2 \vdash l \to C_7 \quad \text{obs}(t_6) = \text{obs}(t_7)$$

and $C_{7 \downarrow_{\omega}}$. By construction of $W$, $\text{obs}(t_6) \in W$. Therefore $W \cap \text{Tr}D_3 \models N_3 \neq \emptyset$ which contradicts the definition of $(\text{obs}(t_1):V:W) \in \text{Ref}(N)$. Therefore it must be that $D' \vdash N_3 \parallel T_2 \neq \omega$ and thus $N \subseteq_{\text{live}} M$. ▲

**B**

**Omitted Proofs**

**Proof of Lemma 2.2 on p. 5.** By RLsys of Figure 3 we get

$$D = (k_1 \ldots \times k_m \times k \times D') \quad M \vdash (k_1) M_1 \ldots \vdash (k_m) M_m \vdash (k) M'$$

because of $D \vdash M \vdash (k) D' \vdash M'$, and

$$D = (k'_1 \ldots \times k'_n \times k \times D'') \quad M \vdash (k'_1) M'_1 \ldots \vdash (k'_n) M'_n \vdash (k) M''$$

because of $D \vdash M \vdash (k) D'' \vdash M''$. By definition of dependency histories, $k$ appears only once in $D$, therefore $m = n$, $k_i = k'_i$ and $D' = D''$. We show $M' = M''$ by induction on $n$ and Lemma A.3. ▲

**Proof of Lemma 2.3 on p. 5.** By Lemmas A.4 and A.6 and induction on $n$, we get $D' \vdash M' \vdash (k_i) \ldots \vdash (k_1) (k) M' \vdash M$. Thus $M' \vdash (k_i) \ldots \vdash (k_1) (k) M'' \equiv M$ and $D' = l_n \times \ldots \times l_1 \times k \times D$, and we derive $D' \vdash M' \vdash (k) D \vdash M'' \equiv M$ as needed. ▲
Proof of Lemma 4.1 on p. 7. By induction on the number of rollback transitions in $D \vdash M \rightsquigarrow D' \vdash M'$.

Base case: the reduction has no rollback transitions. The first property trivially holds.

Inductive case: the reduction can be decomposed (via the rules FW and Bw of Figures 2 and 3, respectively) to:

$$D \vdash M \rightsquigarrow D_1 \vdash M_1 \rightsquigarrow D' \vdash M'$$

because

$$D \vdash M \xrightarrow{(\kappa_1)} D_1 \vdash M_1$$

The reduction $D_1 \vdash M_1 \rightsquigarrow D' \vdash M'$ is strictly smaller than the original one, thus we can apply the induction hypothesis and get that one of the following holds:

1. $D_1 \vdash M_1 \rightsquigarrow D' \vdash M'' \equiv M'$

From Lemma A.22 and (8) we have the sub-cases:

- a. $D \vdash M \xrightarrow{(\kappa_1)} D_1 \vdash M_1 \equiv M_1$, for some $l \in D$. From rule Reqv of Figure 2, $D_1 \vdash M_1' \rightsquigarrow D' \vdash M'' \equiv M'' \equiv M'$, for some $M''$. The proof is completed in this case by (8): $D \vdash M \rightsquigarrow D_1 \vdash M_1 \equiv M_1'$.

- b. $D \vdash M \xrightarrow{(\kappa_1)} D_1 \vdash M_1 \equiv M_1$. The first property follows using again rule Reqv.

2. $D_1 \vdash M_1 \rightsquigarrow D_0 \vdash M_0 \rightsquigarrow D' \vdash M'' \equiv M'$ and $D_1 \vdash M_1 \rightsquigarrow D_0 \vdash M_0 \equiv M_0$, for some $l_0 \in D_1$, $D_0$, $M''$, $M', M_0$.

By Lemma A.19, $D \vdash M \xrightarrow{(\kappa_1)} D_0 \vdash M_0$.

From Lemma A.22 we have the sub-cases:

- a. $D \vdash M \xrightarrow{(l_0)} D_0 \vdash M_0 \equiv M_0$ and $l_0 \in D$.

From rule Reqv of Figure 2, $D_0 \vdash M_0'' \rightsquigarrow D' \vdash M''' \equiv M' \equiv M'$, for some $M'''$. Moreover from (8), $D \vdash M \rightsquigarrow D_1 \vdash M_1 \rightsquigarrow D_0 \vdash M_0'' \equiv M_0 \equiv M_0'''$, establishing the second conclusion of the lemma.

- b. $D \vdash M \xrightarrow{(\kappa)} D_0 \vdash M_0 \equiv M_0$.

From rule Reqv of Figure 2, $D_0 \vdash M_0'' \rightsquigarrow D' \vdash M''' \equiv M' \equiv M'$, for some $M'''$, establishing the first conclusion of the lemma. ▲

Proof of Lemma 4.4 on p. 7. We expand the forward reduction in the assumption using rule FW of Figure 2.

$$\varepsilon \vdash M \xrightarrow{(\kappa)} D \vdash N$$

The following set contains the states reachable from $D \vdash N$ with arbitrary transitions:

$$S = \{D' \vdash N' \mid D' \vdash N \rightsquigarrow D' \vdash N'\}$$

We take the earliest such state: let $(D_1 \vdash N_1) \in S$ such that $D_1$ is a prefix of all states in $S$. If there are multiple states with the same $D_1$ then we pick one of them.

We know that $\varepsilon \vdash M \rightsquigarrow D \vdash N \rightsquigarrow D_1 \vdash N_1$. Therefore, by Corollary 4.2: $\varepsilon \vdash M \rightsquigarrow D_1 \vdash N_1$, which proves the first conclusion of the lemma.

Because $D \vdash N \rightsquigarrow D_1 \vdash N_1$, it follows that if $D_1 \vdash N_1 \rightsquigarrow D_2 \vdash N_2$ then also $D \vdash N \rightsquigarrow D_2 \vdash N_2$, which proves the second conclusion. For the same reason, the fourth conclusion follows from the third.

It remains to show the third conclusion.

We first establish that $D_1 \vdash N_1 \rightsquigarrow D \vdash N' \equiv N$, for some $N'$. From Lemma 4.1 one of the following holds:
1. $D \vdash N \rightarrow^* D_1 \vdash N_1' \equiv N_1$, for some $N_1'$.

In this case, because $D_1$ is the smallest prefix of all $D'$s in $\mathcal{S}$, it must be that $D = D_1$ and the proof is completed by Lemma A.28 which gives us $N = N_1'$ and thus $D_1 \vdash N \rightarrow^* D_1 \vdash N_1' \equiv N_1$.

2. $D \vdash N \rightarrow^* D_0 \vdash N_0 \rightarrow^* D_1 \vdash N_1' \equiv N_1$, and $D \vdash N \rightarrow^* D_0 \vdash N_0' \equiv N_0$, for some $l \in D$, $D_0$, $N_1'$, $M_0$, $M_0'$.

Because $(D_0 \vdash N_0' \equiv N_0)$ is not in $\mathcal{S}$ and we took $D_1$ to be the smallest prefix of all states in $\mathcal{S}$, it must be that $D_0 = D_1$. Thus, from Lemma A.28 $N_0 = N_1'$. From Corollary A.25, $D_3 \vdash N_1' \rightarrow^* D \vdash N$. Note that $\text{wf}(\varepsilon \vdash M)$ and thus $\text{wf}(D \vdash N)$ by Lemma A.18.

Therefore, using rule ReqV, we derive that $D_1 \vdash N_1 \rightarrow^* D \vdash N' \equiv N$, for some $N'$. Hence, $D_1 \vdash N_1 \rightarrow^* D \vdash N' \equiv N$, for some $N'$.

It now suffices to show that if $D_1 \vdash N_1 \rightarrow^* D' \vdash N'$ then $D_1 \vdash N_1 \rightarrow^* D' \vdash N'' \equiv N'$, for some $N''$. We prove this by induction on the number of reductions. The base case is trivial. In the inductive case we have $D_1 \vdash N_1 \rightarrow^* D_2 \vdash N_2' \equiv N_2$, for some $N_2'$. Therefore, $D_2 \vdash N_2' \rightarrow^* D' \vdash N'' \equiv N'$, for some $N''$. If this ($\rightarrow^*$) reduction is a forward transition then the proof is completed. Otherwise $D_1 \vdash N_1 \rightarrow^* D_2 \vdash N_2' \rightarrow^* D' \vdash N'' \equiv N'$. From Lemma A.27 we get that either:

- there exists non-empty $D_3$ such that $D_1 = D_3 \times D'$. In this case we have that $D'$ is a strict prefix of $D_1$. Moreover ($D' \vdash N'' \equiv N'$).
- $D_3$ is larger than or equal to $D_1$: $D_1 \vdash M_1 \rightarrow^* D' \vdash N'' \equiv N'$, as needed.

\textbf{Proof of Theorem 5.1 on p. 8.} In the forward direction, from hypothesis and rules $\text{FW}$ and $\text{Rsv}$ of Figure 2, we get $M \stackrel{\tau(k)}{\rightarrow} M'$. The result follows by induction on this derivation. The case $\text{ReqV}$ follows by the induction hypothesis and Lemma A.32. The reverse direction follows from Lemma A.33.

\textbf{Proof of Proposition 5.3 on p. 9.} By induction on the length of the trace $t$ and then using the preceding lemma to take cases on the transition. The base case, $t = \varepsilon$ banally holds. In the inductive case we have that $t = \alpha(k), s$ with:

$$D \models M \parallel N \stackrel{\tau(k)}{\rightarrow} D_s \models M_s \parallel N_s \stackrel{\alpha}{\rightarrow} D' \models M' \parallel N'$$

We have three cases depending on who generated the action $\tau(k)$, either $M$, or $N$ or both.

In the first case we have that $D \models M \parallel N \stackrel{\tau(k)}{\rightarrow} D_s \models M_s \parallel N_s$, $D \models M \stackrel{\tau(k)}{\rightarrow} D_s \models M_s$ and $N_s \equiv N'$. By Lemma A.46 we have that $D_1 \models M \stackrel{\tau(k)}{\rightarrow} D'$ $D_1 \models M$ and that $D_1 \parallel D_2 \rightarrow D_s$. Since $\text{obs}(t) = \varepsilon$ and $t = \tau(k), s$ we have that $\text{obs}(s) = \varepsilon$. By inductive hypothesis on $D_s \models M_s \parallel N_n \rightarrow^* D' \models R'$ and $D_1' \parallel D_2 \rightarrow D_s$ we have that there exist $M', N', D_1', D_2', s_1$ and $s_2$ such that

$$D_s \models M \stackrel{s_2}{\rightarrow} D_2' \models M', \quad (D_1' \parallel s_1 \triangleright D_2') \parallel (D_2 \parallel s_2 \triangleright D_2') \rightarrow (D_s \parallel s \triangleright D')$$

$$D_s \models N \stackrel{s_2}{\rightarrow} D_2' \models N', \quad R' \equiv M' \parallel N' \quad \text{obs}(s_1) = \text{obs}(l_2)$$

by Lemma A.33 we have that $D_s = k \prec D$ and $D_1' = k \prec D_1$. We can conclude by noticing that:

$$D \models M \stackrel{\tau(k)}{\rightarrow} D_s \models M \stackrel{s_1}{\rightarrow} D_2' \models M' \quad \text{implies} \quad D \models M \stackrel{\tau(k), s_1}{\rightarrow} D_2' \models M'$$

$$D \models N \stackrel{s_2}{\rightarrow} D_s \models N \stackrel{s_2}{\rightarrow} D_2' \models N' \quad \text{implies} \quad D \models N \stackrel{s_2}{\rightarrow} D_2' \models N'$$
By applying rule ZTL of Figure 6 we have that

\[
(k \triangleleft D_1 \vdash s_1 \triangleright D'_1) \parallel (D_2 \vdash s_2 \triangleright D''_2) \implies (k \triangleleft D \triangleright s \triangleright D')
\]

implies

\[
(D_1 \vdash \alpha(k), s_1 \triangleright D'_1) \parallel (D_2 \vdash s_2 \triangleright D'_2) \implies (D \vdash \alpha(k), s \triangleright D')
\]

The other two cases are similar.

Proof of Proposition 5.4 on p. 9. By induction on

\[
(D_1 \vdash t_1 \triangleright D'_1) \parallel (D_2 \vdash t_2 \triangleright D''_2) \implies (D \vdash t \triangleright D').
\]

We have four cases, corresponding to the rules ZTe, ZTL, ZTR and ZSYNC of Figure 6. We will consider just the first two cases, the remaining ones are similar to the second case.

ZTe In this case we have that \( t_1 = t_2 = t = \epsilon \), \( D_1 = D'_1 \) and \( D_2 = D''_2 \). Then the proposition banally holds.

ZTL In this case we have that \( t_1 = \alpha(k), s_1 : D_1 \models M \xrightarrow{\alpha(k)} D_s \models M_s \xrightarrow{s_1} D'_1 \models M' \quad D_2 \models N \xrightarrow{t_2} D'_2 \models N' \)

By Lemma A.41 we have that \( D_s = k \triangleleft D_1 \). By initial hypothesis we have that

\[
(D_1 \vdash \alpha(k), s_1 \triangleright D'_1) \parallel (D_2 \vdash t_2 \triangleright D'_2) \implies (D \vdash \alpha(k), s \triangleright D')
\]

and by applying rule ZTL of Figure 6 we can derive

\[
(k \triangleleft D_1 \vdash s_1 \triangleright D'_1) \parallel (D_2 \vdash t_2 \triangleright D'_2) \implies (k \triangleleft D \triangleright s \triangleright D')
\]

We can now apply the inductive hypothesis and derive that \( k \triangleleft D \models M_s \parallel N \xrightarrow{\epsilon} D' \models R' \).

We have now to show that \( D \models M \parallel N \xrightarrow{\alpha(k)} k \triangleleft D \models M_s \parallel N \). By applying Lemma A.43 on the derivation \((k \triangleleft D_1 \vdash s_1 \triangleright D'_1) \parallel (D_2 \vdash t_2 \triangleright D'_2) \implies (k \triangleleft D \triangleright s \triangleright D')\), we have that \( k \triangleleft D_1 \parallel D_2 \implies k \triangleleft D \), and by Lemma A.42 this implies that \( D_1 \parallel D_2 \rightarrow D \). Since we are considering well formed system, \( D_1 \models M \xrightarrow{\alpha(k)} D_s \models M_s \) implies that \( \text{keys}(M) \subseteq \text{keys}(D_1) \) and by Lemma A.41 we have that \( \text{keys}(D_1) \subseteq \text{keys}(D) \) and hence \( \text{keys}(M) \subseteq \text{keys}(D) \). We can now apply Lemma A.37 and obtain \( D \models M \xrightarrow{\alpha(k)} k \triangleleft D \models M_s \) and by applying rule TRar of Figure 5 we can derive \( D \models M \parallel N \xrightarrow{\alpha(k)} k \triangleleft D \models M_s \parallel N \). We can conclude by noticing that:

\[
D \models M \parallel N \xrightarrow{\alpha(k)} k \triangleleft D \models M_s \parallel N \xrightarrow{\epsilon} D' \models R'
\]
Proof of Lemma A.17 on p. 16. By contradiction. Assume \( [\mu; k] \in N \). By well-formedness (Definition 2.4), \( D = k_1 \prec \ldots \prec k_n \prec k \prec D' \) and \( M \overset{(k_1)}{\Rightarrow} M_1 \ldots \overset{(k_n)}{\Rightarrow} M_n \overset{(k)}{\Rightarrow} M' \) and \( \mathsf{wf}(M_n) \). By Lemma A.15, \( \tilde{k}, k \notin \mu, \mu_2, \mu \). Therefore, by Lemma A.12, \( [\mu_1; k] \in M_n, [\mu_2; k] \in M_n, \) and \( [\mu; k] \in M_n \). Moreover, by Lemma A.9, \( k \notin M' \). Then it is not possible to derive \( M' \overset{(k)}{\Rightarrow} M_n \), because forward reductions produce at most two memories (Lemma A.7). This means that \( M_n \) is not well-produced, which is a contradiction. Thus it must be that for all \( [\mu; l] \in N, k \neq l \).

Proof of Lemma A.32 on p. 18. By induction on the derivation \( M \equiv N \), proving the lemma first for non-\( \tau \) transitions. In the proof for \( \tau \)-transitions we use the lemma for non-\( \tau \) transitions.

Proof of Lemma A.46 on p. 20. Since \( D \models M \parallel N \overset{\alpha(k)}{\Rightarrow} D' \models R' \), by Lemma A.33 we have that \( D' = k \prec D \). We then proceed by case analysis on the reduction \( D \models M \parallel N \overset{\alpha(k)}{\Rightarrow} D' \models R' \). We have three cases: \( M \) by itself did the action, \( N \) by itself did the action or both \( M \) and \( N \) contributed to the action.

1. We have that \( D \models M \overset{\alpha(k)}{\Rightarrow} D' \models M' \) implies \( D \models M \parallel N \overset{\alpha(k)}{\Rightarrow} D' \models M' \parallel N \), with \( k \notin D, M, N \). Since \( D_1 \parallel D_2 \rightarrow D \) by Lemma A.41 we have that \( \mathsf{keys}(D_i) \subseteq \mathsf{keys}(D) \) with \( i \in \{1, 2\} \), and since \( k \notin D \) then \( k \perp D_1, D_2 \). We have then \( D_1 \models M \overset{\alpha(k)}{\Rightarrow} D_1' \models M' \) and by Lemma A.33 we have that \( D_1' = k \prec D_1 \). By rule \( \mathsf{ZTL} \) of Figure 6 we have that \( k \prec D_1 \parallel D_2 \rightarrow k \prec D \), and we can conclude by noticing that \( D' \models k \prec D \).

2. this case is similar to the previous one.

3. We have that \( D \models M \overset{\alpha(k)}{\Rightarrow} D' \models M' \) and \( D \models N \overset{\pi(k)}{\Rightarrow} D' \models N' \) imply \( D \models M \parallel N \overset{\tau(k)}{\Rightarrow} D' \models M' \parallel N' \), with \( k \notin D, M, N \). Since \( D_1 \parallel D_2 \rightarrow D \) by Lemma A.41 we have that \( \mathsf{keys}(D_i) \subseteq \mathsf{keys}(D) \) with \( i \in \{1, 2\} \), and since \( k \notin D \) then \( k \perp D_1, D_2 \). We have then \( D_1 \models M \overset{\alpha(k)}{\Rightarrow} D_1' \models M' \) and \( D_2 \models M \overset{\alpha(k)}{\Rightarrow} D_2' \models M' \) by Lemma A.33 we have that \( D_1' = k \prec D_1 \) and \( D_2' = k \prec D_2 \). By rule \( \mathsf{ZTsync} \) of Figure 6 we have that \( k \prec D_1 \parallel k \prec D_2 \rightarrow k \prec D \), and we can conclude by noticing that \( D' \models k \prec D \).

Proof of Lemma A.47 on p. 20. By induction on the structure of \( t_1 \) and \( t_2 \). In the base case the that \( t_1 = t_2 = \epsilon \) and by definition of trace typing (Definition 5.2) we have that \( D_1' = D_1 \) and \( D_2' = D_2 \). By applying rule \( \mathsf{ZT} \) we have that
\[
(D_1 \vdash \epsilon \triangleright D_1) \parallel (D_2 \vdash \epsilon \triangleright D_2) \rightarrow (D_3 \vdash \epsilon \triangleright D_3)
\]
and the lemma holds by setting \( t_3 = \epsilon \) and \( D_3' = D_3 \). In the inductive hypothesis we have to do a case analysis on the form of \( t_1 \).

\( t_1 = a(k), t_1' \). Since \( \mathsf{obs}(t_1) = t_1' \) and \( \mathsf{obs}(t_2) = t_2' \) and \( \mathsf{T}_1 = t_2' \), then there exists \( \pi(k) \) such that \( t_2 = \tau(l), \pi(k), t_2' \) and \( \tilde{l} \perp D_i' \) which implies \( \tilde{l} \prec D_1 \). We have then that \( D_1 \parallel \tilde{l} \prec D_2 \rightarrow \tilde{l} \prec D_3 \). Since \( \mathsf{obs}(t_1) = \epsilon \) and \( D_1' \parallel D_2' \rightarrow D_3 \). Now, by applying rule \( \mathsf{ZT} \) \( n \) times, with \( n \) being the size of \( \tau(l) \) we obtain that
\[
(D_1 \vdash a(k), t_1' \triangleright D_1') \parallel (\tilde{l} \prec D_2 \vdash \pi(k), t_2' \triangleright D_2') \rightarrow (\tilde{l} \prec D_3 \vdash t_3' \triangleright D_3')
\]
with \( \mathsf{obs}(t_3') = \epsilon \) and \( D_3' \rightarrow D_3 \). Since \( \mathsf{obs}(\tau(l), t_3') = \mathsf{obs}(t_3) = \epsilon \) we can conclude.
\( t_1 = \tau(k), t_1' \). By initial hypothesis we have that \( k \not\leq D_2' \) and this implies \( k \not\leq D_2 \). We have then that \( k \prec D_1 \parallel D_2 \rightarrow k \prec D_3 \). Since \( (k \prec D_1 \vdash t_1' \triangleright D'_1) \), by inductive hypothesis we have that

\[
(k \prec D_1 \vdash t_1' \triangleright D'_1) \parallel (D_2 \vdash t_2' \triangleright D'_2) \rightarrow (k \prec D_3 \vdash t_3' \triangleright D'_3)
\]

with \( D'_1 \parallel D'_2 \rightarrow D'_3 \) and \( \text{obs}(t_3') = \epsilon \). We can now apply rule ZTR and obtain that:

\[
(D_1 \vdash \tau(k), t_1' \triangleright D'_1) \parallel (D_2 \vdash t_2' \triangleright D'_2) \rightarrow (\prec D_3 \vdash \tau(k), t_3' \triangleright D'_3)
\]

and we can conclude by letting \( t_3 = \tau(k), t_3' \) and by noticing that \( \text{obs}(t_3) = \text{obs}(t_3') = \epsilon \), as desired.

\( t_1 = \epsilon \) and \( t_2 \neq \epsilon \). By initial hypothesis, \( t_2 = \tau(k), t_2' \) with \( \text{obs}(t_2') = \epsilon \). Moreover, since \( t_1 = \epsilon \) we have that \( D_1 = D_1' \). Since \( D_2 \parallel D_2 \rightarrow D_3 \) and \( k \not\leq D_1 \) this implies also \( k \not\leq D_1 \). We then have that \( D_1 \parallel k \prec D_2 \rightarrow k \prec D_3 \). We can then apply inductive hypothesis and obtain that

\[
(D_1 \vdash \epsilon \triangleright D_1) \parallel (k \prec D_2 \vdash t_2' \triangleright D'_2) \rightarrow (k \prec D_3 \vdash t_3' \triangleright D'_3)
\]

with \( D_1 \parallel D'_2 \rightarrow D_3 \) and \( \text{obs}(t_3) = \epsilon \). We can then apply rule ZTR and obtain

\[
(D_1 \vdash \epsilon \triangleright D_1) \parallel (D_2 \vdash \tau(k), t_2' \triangleright D'_2) \rightarrow (D_3 \vdash \tau(k), t_3' \triangleright D'_3)
\]

as desired.

\( t_1 = a(k), t_1' \) and \( t_2 = a(k), t_2' \). Similar to the previous case, with the use of rule ZTsync to conclude.

\[\blacktriangleright\]

**Proof of Lemma A.48 on p. 20.** Let \( t \in \text{Tr}(C) \) and \( p \) be a permutation such that \( p \not\leq C \). We have \( C \overset{t}{\rightarrow} C' \), for some \( t' \) and \( C' \) with \( \text{obs}(t') = t \). From Lemma A.44, \( pC \overset{p!t'}{\rightarrow} pC' \).

Moreover, \( \text{obs}(pt) = pt \) and \( pC = C \) because \( p \not\leq C \). Thus \( pt \in \text{Tr}(C) \). \[\blacktriangleright\]