CRASH-STOP FAILURES IN ASYNCHRONOUS MULTIPARTY SESSION TYPES

ADAM D. BARWELL [a], PING HOU [b], NOBUKO YOSHIDA [b], AND FANGYI ZHOU [c]

[a] University of St Andrews, UK
e-mail address: adb23@st-andrews.ac.uk

[b] University of Oxford, UK
e-mail address: ping.hou@cs.ox.ac.uk, nobuko.yoshida@cs.ox.ac.uk

[c] No affiliation
e-mail address: me@fangyi.io

Abstract. Session types provide a typing discipline for message-passing systems. However, their theory often assumes an ideal world: one in which everything is reliable and without failures. Yet this is in stark contrast with distributed systems in the real world. To address this limitation, we introduce a new asynchronous *multiparty session types* (MPST) theory with *crash-stop failures*, where processes may crash arbitrarily and cease to interact after crashing. We augment asynchronous MPST and processes with *crash handling* branches, and integrate crash-stop failure semantics into types and processes. Our approach requires no user-level syntax extensions for global types, and features a formalisation of global semantics, which captures complex behaviours induced by crashed/crash handling processes.

Our new theory covers the entire *spectrum* of unreliability, ranging from the ideal world of total reliability to entirely unreliable scenarios where any process may crash, using *optional reliability assumptions*. Under these assumptions, we demonstrate the sound and complete correspondence between global and local type semantics, which guarantee deadlock-freedom, protocol conformance, and liveness of well-typed processes *by construction*, even in the presence of crashes.

1. Introduction

Background. As distributed programming grows increasingly prevalent, significant research effort has been devoted to improve the reliability of distributed systems. A key aspect of this research focuses on studying *un*reliability (or, more specifically, failures). Modelling unreliability and failures enables a distributed system to be designed to be more tolerant of failures, and thus more resilient.

In pursuit of methods to achieve fundamental reliability – *safety* in distributed communication systems – *session types* [HVK98] provide a lightweight, type system–based approach to message-passing concurrency. In particular, *Multiparty Session Types* (MPST) [HYC08]

Key words and phrases: Session Types, Failure Handling, Concurrency, Session Calculus.

Preprint submitted to
Logical Methods in Computer Science

facilitate the specification and verification of communication between message-passing processes in concurrent and distributed systems. The typing discipline prevents common communication-based errors, e.g. deadlocks and communication mismatches [HYC16, SY19].

Nevertheless, the challenge to account for unreliability and failures persists for session types: most session type systems assume that both participants and message transmissions are *reliable*, i.e. without failures. In a real-world setting, however, participants may crash, communication channels may fail, and messages may be lost. The lack of failure modelling in session type theories prevents their application to large-scale distributed systems.

Recent works [VHEZ21, PNW22, LNY22, LBD23, BSYZ22] address the gap in failure modelling within session types with various techniques. Viering *et al.* [VHEZ21] introduce *failure suspicion*, where a participant may suspect their communication partner has failed, and act accordingly. Peters *et al.* [PNW22] introduce *reliability annotations* at type level, and fall back to a given *default* value in case of failures. Lagaillardie *et al.* [LNY22] propose a framework of *affine* multiparty session types, where a session can terminate prematurely, e.g. in case of failures. Barwell *et al.* [BSYZ22] incorporate *crash-stop failures* in session types, where a generalised type system validates safety and liveness properties. Le Brun and Dardha [LBD23] adopt a similar approach but extend it to include additional failure types, e.g. message losses, reordering, and delays.

This Paper. Unlike the aforementioned approaches, we advance failure modelling in session types by introducing a new asynchronous multiparty session type theory that incorporates *crash-stop* failures [CGR11, §2.2], where a process may crash and cease to interact with others. This model is standard in distributed systems, and is used in related work on session types with error-handling capabilities [VCE+18, VHEZ21]. However, unlike previous work, we allow *any* process to crash arbitrarily, and support optional assumptions on non-crashing processes. Our extended asynchronous MPST theory effectively models failures with crash-stop semantics, and demonstrates that the usual session type guarantees remain valid, i.e. communication safety, deadlock-freedom, and liveness.

In our new theory, we add *crashing* and *crash handling* semantics to processes and types. With *minimal* changes to the standard surface syntax compared with the original MPST theory, we model a variety of subtle, complex behaviours arising from unreliable communicating processes. An active process P may crash arbitrarily, and a process Q interacting with P might need to be prepared to handle possible crashes. Messages sent from Q to a crashed P are lost – but if Q tries to receive from P, then Q can detect that P has crashed, and take a crash handling branch. Meanwhile, another process R may (or may not) have detected P's crash, and may be handling it – and in either case, any interaction between Q and R should remain correct.

A key design feature of our framework is to support *optional reliability assumptions*: a programmer may declare that some peers will *never* crash for the duration of the protocol. Such optional assumptions allow for simplifying protocols and programs: if a participant is declared reliable, its peers can interact with it without implementing crash detection and handling. Moreover, by making such assumptions explicit and customisable, our theory supports a *spectrum* of assumptions ranging from only having sessions with reliable participants (as in classic MPST works [HYC16, SY19]), to having no reliable participants at all.

Another fundamental aspect of our theory is its adherence to a *top-down* methodology, which stems from the original MPST theory [HYC08]. This approach starts the design of

multiparty session types with a given *global* type (top), and processes rely on *local* types (bottom) obtained from the global type. The global and local types reflect the global and local communication behaviours respectively. Well-typed processes that conform to a global type are guaranteed to be *correct by construction*, enjoying full guarantees (safety, deadlock-freedom, and liveness) from the theory.

The use of global types in our design for handling failures in multiparty session types presents three distinct advantages: *(1)* there is no user-level syntax extension of global types compared with the original MPST global types, apart from a special label to signify crash handling branches; *(2)* global types provide a simple, high-level means to both specify a protocol abstractly and automatically derive local types; and, *(3)* under (optional) reliability assumptions, desirable behavioural properties such as communication safety, deadlock-freedom, and liveness are guaranteed by construction.

In contrast to the *synchronous* semantics used by Barwell *et al.* [BSYZ22], we model *asynchronous* semantics, where messages can be buffered whilst in transit. We focus on asynchronous systems since most communication in the real distributed world is asynchronous. Although Le Brun and Dardha [LBD23] develop a generic typing system incorporating asynchronous semantics, their approach results in type-level properties becoming undecidable [LBD23, §4.4]. With global types, we restore decidability at a minor cost to expressivity.

Outline. This article represents an extended rendition of the work on asynchronous multiparty session type theory with crash-stop failures as initially introduced in [BHYZ23]. Our enhanced version provides a more comprehensive and refined presentation, offering detailed definitions, an in-depth exploration of related work, and additional examples. Moreover, we incorporate a new section (Section 6) that illustrates our approach through an extensive case study of the Non-Blocking Atomic Commits abstraction, described in [CGR11]. In contrast to [BHYZ23], we have omitted the implementation of our theory, as our primary focus is on the theoretical aspects.

Note that, for clarity and simplicity, our framework does not support delegation (session passing), as our primary focus is on developing a top-down methodology for handling failures in multiparty session types. However, all the results presented in the paper are applicable to systems that do incorporate delegation. For readers interested in the generalised MPST theory that includes both crash-stop failures and delegation, please refer to [BSYZ22].

Section 2 We begin with an overview of our methodology.
Section 3 Addressing the challenge of defining the semantics of crashing and crash handling within the context of the crash-stop model, we introduce an asynchronous multiparty session calculus with minimal syntax changes.
Section 4 We introduce an extended theory of asynchronous multiparty session types with semantic modelling of crash-stop failures. The challenges here involve capturing the non-trivial crash-stop semantics of global and local types under *optional* reliability assumptions, as well as demonstrating that the sound and complete operational correspondence between the two semantics (Theorems 4.20 and 4.21) can ensure desirable global type properties such as communication safety, deadlock-freedom, and liveness in local types through projection, even in the presence of crashes (Theorem 4.31).
Section 5 We present a typing system for our asynchronous multiparty session calculus. The challenges comprise proving behavioural properties for typed sessions: subject reduction, session fidelity, deadlock-freedom, and liveness, in corresponding theorems (Theorems 5.2, 5.3, 5.5, and 5.7).

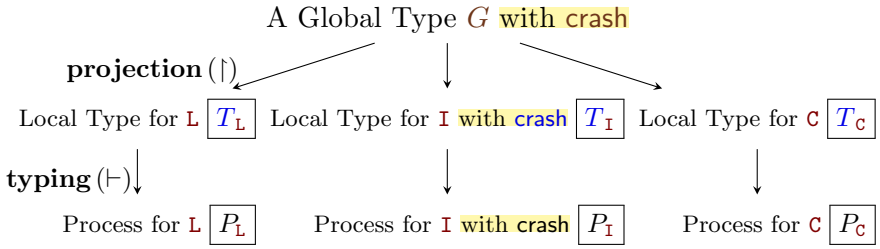

FIGURE 1. Top-down view of MPST with crash.

Section 6 We provide a case study on the Non-Blocking Atomic Commits abstraction, which facilitates the reliable execution of transactions in distributed database systems.

We discuss related work in **Section 7** and conclude in **Section 8**.

2. Overview

In this work, we follow a standard top-down design approach enabling *correctness by construction*, but enrich asynchronous MPST with crash-stop semantics. As depicted in Fig. 1, we formalise (asynchronous) multiparty protocols with crash-stop failures as global types with *crash handling branches* (crash). These are projected onto local types, which may similarly contain crash handling branches (crash). The projected local types are, in turn, used to type-check processes (also with crash handling branches (crash)) that are written in a session calculus. As an example, we consider a simple *distributed logging* scenario, which is inspired by the logging-management protocol [LNY22], but extended with a third participant.

The Simpler Logging protocol consists of a *logger* (L) that controls the logging services, an *interface* (I) that provides communications between logger and client, and a *client* (C) that requires logging services via interface. Initially, the logger process L sends a heartbeat message trigger to the interface process I. Then the client C sends a command to the interface to read the logs (read). When a read request is sent, it is forwarded to the logger, and the logger responds with a report, which is then forwarded onto the client. When all participants (i.e. logger, interface, and client) are assumed reliable, i.e. without failures or crashes, this logging behaviour can be represented by the *global type* G_0:

$$G_0 = \mathtt{L}\to\mathtt{I}:\mathtt{trigger}.\mathtt{C}\to\mathtt{I}:\mathtt{read}.\mathtt{I}\to\mathtt{L}:\mathtt{read}.\mathtt{L}\to\mathtt{I}:\mathtt{report}(\mathrm{log}).\mathtt{I}\to\mathtt{C}:\mathtt{report}(\mathrm{log}).\mathtt{end}$$

Here, G_0 is a specification of the Simpler Logging protocol between multiple roles from a global perspective.

In the real distributed world, all participants in the Simpler Logging system may fail. Ergo, we may need to model protocols with failures or crashes and handling behaviour thereof, e.g. should the client fail after the logging has started, the interface will inform the logger to stop and exit. We follow [CGR11, §2.2] to model a *crash-stop* semantics, where we assume that roles can crash *at any time* unless assumed *reliable* (never fail or crash). For simplicity, we assume the interface I and the logger L to be reliable. The above logging behaviour, incorporating crash-stop failures, can be represented by extending G_0 with a

branch handling crashes of C:

$$G = \mathtt{L}\to\mathtt{I}:\mathtt{trigger}.\mathtt{C}\to\mathtt{I}:\left\{\begin{array}{l}\mathtt{read}.\mathtt{I}\to\mathtt{L}:\mathtt{read}.\mathtt{L}\to\mathtt{I}:\mathtt{report}(\mathrm{log}).\mathtt{I}\to\mathtt{C}:\mathtt{report}(\mathrm{log}).\mathtt{end}\\ \mathtt{crash}.\mathtt{I}\to\mathtt{L}:\mathtt{fatal}.\mathtt{end}\end{array}\right\} \tag{2.1}$$

We model crash detection on receiving roles: when a role I is waiting to receive a message from role C, the receiving role I is able to detect whether C has crashed. Since crashes are detected only by the receiving role, we do not require a crash handling branch on the communication step between I and C – nor do we require them on any interaction between L and I (since we are assuming that L and I are reliable). Note that our global type syntax has minimal changes from the standard syntax, with additional crashing and crash handling semantics. We give details in Section 4.

Following the MPST top-down methodology, a global type is then *projected* onto *local types*, which describe communications from the perspective of a single role. In our unreliable Simpler Logging example, G is projected onto three local types (one for each role C, L, I):

$$T_{\mathtt{C}} = \mathtt{I}\oplus\mathtt{read}.\mathtt{I}\&\mathtt{report}(\mathrm{log}).\mathtt{end} \quad T_{\mathtt{L}} = \mathtt{I}\oplus\mathtt{trigger}.\mathtt{I}\&\left\{\begin{array}{l}\mathtt{read}.\mathtt{I}\oplus\mathtt{report}(\mathrm{log}).\mathtt{end}\\ \mathtt{fatal}.\mathtt{end}\end{array}\right\}$$

$$T_{\mathtt{I}} = \mathtt{L}\&\mathtt{trigger}.\mathtt{C}\&\left\{\begin{array}{l}\mathtt{read}.\mathtt{L}\oplus\mathtt{read}.\mathtt{L}\&\mathtt{report}(\mathrm{log}).\mathtt{C}\oplus\mathtt{report}(\mathrm{log}).\mathtt{end}\\ \mathtt{crash}.\mathtt{L}\oplus\mathtt{fatal}.\mathtt{end}\end{array}\right\}$$

Here, $T_{\mathtt{I}}$ states that the interface I first receives a trigger message from the logger L; then I either expects a read request from the client C, or detects the crash of C and handles it (in crash) by sending the fatal message to notify the logger L. Again, the syntax of local types does not depart from the standard, but we add additional crash modelling and introduce a stop type for crashed endpoints, explained in Section 4. We show the operational correspondence between global and local type semantics in Theorems 4.20 and 4.21, and demonstrate that a projectable global type always produces a safe, deadlock-free, and live typing context in Theorem 4.31.

The next step in this top-down methodology is to use local types to type-check processes P_i executed by role $\mathtt{p}_i$ in our session calculus. For example, $T_{\mathtt{I}}$ can be used to type check the interface I that executes the process:

$$\mathtt{L}?\mathtt{trigger}.\sum\left\{\begin{array}{l}\mathtt{C}?\mathtt{read}.\mathtt{L}!\mathtt{read}.\mathtt{L}?\mathtt{report}(x).\mathtt{C}!\mathtt{report}\langle x\rangle.\mathbf{0}\\ \mathtt{C}?\mathtt{crash}.\mathtt{L}!\mathtt{fatal}.\mathbf{0}\end{array}\right\}$$

In our operational semantics (Section 3), we allow active processes executed by unreliable roles to crash *arbitrarily*. Therefore, the role executing the crashed process also crashes, and is assigned the local type stop. To ensure that a communicating process is type-safe even in the presence of crashes, we require that its typing context satisfies a *safety property* accounting for possible crashes (Definition 4.24), which is preserved by projection (Theorem 4.31). Despite minor changes in the surface syntax of types and processes, additional semantics surrounding crashes adds subtleties even in standard results. We prove subject reduction and session fidelity results, accounting for crashes and sets of reliable roles, in Theorems 5.2 and 5.3.

3. Asynchronous Multiparty Session Calculus with Crash-Stop Semantics

In this section, we formalise the syntax (Section 3.1) and operational semantics (Section 3.2) of our asynchronous multiparty session calculus with process failures and crash detection. For clarity of presentation, delegation is not supported in our system.

$$v ::= \mathtt{n} \mid \mathtt{i} \mid \mathtt{true} \mid \mathtt{false} \mid \mathtt{""} \mid \mathtt{()} \mid \cdots$$
$$e ::= x \mid v \mid \mathtt{succ}(e) \mid \mathtt{neg}(e) \mid \neg e \mid e < e' \mid \cdots$$

$P, Q ::=$	**Processes**	$\mathbb{M} ::=$	**Sessions**
$\sum_{i\in I} \mathtt{p}?\mathtt{m}_i(x_i).P_i$	*external choice*	$\mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h$	*participant*
$\mathtt{p}!\mathtt{m}\langle e\rangle.P$ (*where* $\mathtt{m} \neq \mathsf{crash}$)	*output*	$\mathbb{M} \mid \mathbb{M}$	*parallel*
$\mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q$	*conditional*	$h ::=$	**Queues**
X	*process variable*	ϵ	*empty*
$\mu X.P$	*recursion*	$\oslash$	*unavailable*
$\mathbf{0}$	*inaction*	$(\mathtt{p}, \mathtt{m}(v))$	*message*
$\lightning$	*crashed*	$h \cdot h$	*concatenation*

FIGURE 2. Syntax of values, expressions, sessions, processes, and queues. Appreciable changes w.r.t. the standard session calculus [GPP+21] are highlighted.

3.1. **Syntax of Session Calculus with Crash-Stop Failures.** Our asynchronous multiparty session calculus models processes that may crash arbitrarily. Our formalisation is based on [GPP+21] – but additionally follows the *fail-stop* model in [CGR11, §2.7], where processes may crash and never recover, and where process failures can be detected by failure detectors [CGR11, §2.6.2] [CT96] when attempting to receive messages.

The syntax of our calculus is presented in Fig. 2. Values and expressions are standard: a *value*, ranged over by $v, v', \ldots$, can be a natural number $\mathtt{n}$, an integer $\mathtt{i}$, a boolean $\mathtt{true}$ or $\mathtt{false}$, a string $\mathtt{""}$, a unit $\mathtt{()}$ (often omitted for brevity), or any other specifically tailored value; an *expression*, ranged over by $e, e', \ldots$, can be a variable, a value, or a term built from expressions by applying operators, such as $\mathtt{succ}, \mathtt{neg}, \neg$, and $<$. We fix a set of *roles* $\mathcal{R}$, ranged over by $\mathtt{p}, \mathtt{q}, \mathtt{r}, \ldots$.

A process, ranged over by $P, Q, \ldots$, is a communication agent within a session. An *output* process $\mathtt{p}!\mathtt{m}\langle e\rangle.P$ sends a message to another role $\mathtt{p}$ in the session, where the message is labelled $\mathtt{m}$, and carries a payload expression e, then the process continues as P. An *external choice* (*input*) process $\sum_{i\in I} \mathtt{p}?\mathtt{m}_i(x_i).P_i$ receives a message from another role $\mathtt{p}$ in the session, among a finite set of indices I, if the message is labelled $\mathtt{m}_i$, then the payload would be received as x_i, and the process continues as P_i. Note that our calculus uses crash as a special message label denoting that a participant (role) has crashed. Such a label cannot be sent by any process, but a process can implement crash detection and handling by receiving it. Consequently, an output process *cannot* send a crash message (side condition $\mathtt{m} \neq \mathsf{crash}$), whereas an input process may include a *crash handling branch* of the form $\mathsf{crash}.P'$ meaning that P' is executed when the sending role has crashed. A *conditional* process $\mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q$ continues as either P or Q depending on the evaluation of e (which is detailed later in Section 3.2). We allow *recursion* at the process level using $\mu X.P$ and X, requiring process recursion variables to be guarded by an input or output action. Finally, we write $\mathbf{0}$ for an *inactive* process, representing a successful termination; and $\lightning$ for a *crashed* process, representing a termination due to failure.

An *incoming queue*[1], ranged over by $h, h', h_i, \ldots$, is a sequence of messages tagged with their origin. We write ϵ for an *empty* queue; $\oslash$ for an *unavailable* queue; and $(\mathtt{p}, \mathtt{m}(v))$ for a message sent from $\mathtt{p}$, labelled $\mathtt{m}$, and containing a payload value v. We write $h_1 \cdot h_2$ to denote the concatenation of two queues h_1 and h_2. When describing incoming queues, we

consider two messages from different origins as swappable: $h_1 \cdot (\mathtt{q}_1, \mathtt{m}_1(v_1)) \cdot (\mathtt{q}_2, \mathtt{m}_2(v_2)) \cdot h_2$ is structurally equivalent to $h_1 \cdot (\mathtt{q}_2, \mathtt{m}_2(v_2)) \cdot (\mathtt{q}_1, \mathtt{m}_1(v_1)) \cdot h_2$ whenever $\mathtt{q}_1 \neq \mathtt{q}_2$. Moreover, we consider concatenation $(\cdot)$ as associative, and the empty queue ϵ as the identity element for concatenation.

A session, ranged over by $\mathbb{M}, \mathbb{M}', \mathbb{M}_i, \ldots$, consists of processes and their respective incoming queue, indexed by their roles. A single entry for a role $\mathtt{p}$ is denoted $\mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h$, where P is the process for $\mathtt{p}$ and h is the incoming queue. Entries are composed together in parallel as $\mathbb{M} \mid \mathbb{M}'$, where the roles in $\mathbb{M}$ and $\mathbb{M}'$ are disjoint. We consider parallel composition as commutative and associative, with $\mathtt{p} \triangleleft \mathbf{0} \mid \mathtt{p} \triangleleft \epsilon$ as a neutral element of the operator. We write $\prod_{i \in I} (\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i)$ for the parallel composition of multiple entries in a set.

3.2. **Operational Semantics of Session Calculus with Crash-Stop Failures.** The *evaluation* of an expression is illustrated in Fig. 3, where $e \downarrow v$ indicates that the expression e evaluates to the value v, and an *evaluation context* $\mathcal{E}$ is an expression containing exactly one hole. The successor operation `succ` is defined only for natural numbers, the negation `neg` is defined for integers, the logical negation $\neg$ is defined only for boolean values, and the less-than operator $<$ is applied only to integers.

We give the operational semantics of our session calculus in Definition 3.1, using a *structural precongruence* $\Rrightarrow$ defined in Fig. 5. Structural precongruence is a *preorder* that relates sessions based on inconsequential structural modifications. A standard structural congruence $\equiv$ can be defined as the symmetric closure of $\Rrightarrow$: $\Rrightarrow \cup \Rrightarrow^{-}$.

Our semantics parameterises on a (possibly empty) set of *reliable* roles $\mathscr{R}$, i.e. roles assumed to *never crash*. This approach enables us to model *optional reliability*, allowing for the representation of a *spectrum* of reliability assumptions that range from total process reliability (as seen in standard session types work, i.e. $\mathscr{R} = \mathcal{R}$) to total unreliability (i.e. $\mathscr{R} = \emptyset$).

Definition 3.1 (Session Reductions)**.** The session reduction relation $\rightarrow_{\mathscr{R}}$ is inductively defined by the rules in Fig. 4, parameterised by a fixed set $\mathscr{R}$ of reliable roles. We write $\rightarrow$ whenever $\mathscr{R}$ is not significant. We write $\rightarrow^*_{\mathscr{R}}$ (resp. $\rightarrow^*$) for the reflexive and transitive closure of $\rightarrow_{\mathscr{R}}$ (resp. $\rightarrow$).

Our operational semantics retains the basic rules in [GPP+21], but also includes (highlighted) rules for crash-stop failures and crash handling, adapted from [BSYZ22]. Rules [R-SEND] and [R-RCV] model ordinary message delivery and reception: an output process located at $\mathtt{p}$ sending to $\mathtt{q}$ would append a message to the incoming queue of $\mathtt{q}$; and an input process located at $\mathtt{p}$ receiving from $\mathtt{q}$ would consume the first message from the incoming queue. Rules [R-COND-T] and [R-COND-F] model the conditional process; and rule [R-STRUCT] permits reductions up to structural precongruence.

With regard to crashes and related behaviour, rule [R-↯] models process crashes: an active ($P \neq \mathbf{0}$) process located at an unreliable role ($\mathtt{p} \notin \mathscr{R}$) may reduce to a crashed process $\mathtt{p} \triangleleft \lightning$, with its incoming queue becoming unavailable $\mathtt{p} \triangleleft \oslash$. Rule [R-SEND-↯] models a message delivery to a crashed role (and thus an unavailable queue), and the message becomes lost and would not be added to the queue. Rule [R-RCV-⊙] models crash detection, which activates as a 'last resort': an input process at $\mathtt{p}$ receiving from $\mathtt{q}$ would first attempt find a message

[1] In [GPP+21], the queues are outgoing instead of incoming. We use incoming queues to model our crashing semantics more easily.

$$v \downarrow v \quad \mathtt{succ(n)} \downarrow (\mathtt{n}+1) \quad \mathtt{neg(i)} \downarrow (-\mathtt{i}) \quad \neg\mathtt{true} \downarrow \mathtt{false} \quad \neg\mathtt{false} \downarrow \mathtt{true}$$

$$\mathtt{i}_1 < \mathtt{i}_2 \downarrow \begin{cases} \mathtt{true} & \text{if } \mathtt{i}_1 < \mathtt{i}_2 \\ \mathtt{false} & \text{otherwise} \end{cases} \qquad \frac{e \downarrow v \quad \mathcal{E}(v) \downarrow v'}{\mathcal{E}(e) \downarrow v'}$$

FIGURE 3. Expression evaluation.

$$\begin{array}{ll}
\text{[R-}\lightning\text{]} & \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathbb{M} \;\rightarrow_{\mathscr{R}}\; \mathtt{p} \triangleleft \lightning \mid \mathtt{p} \triangleleft \oslash \mid \mathbb{M} \qquad (P \neq \mathbf{0}, \mathtt{p} \notin \mathscr{R}) \\
\text{[R-SEND]} & \mathtt{p} \triangleleft \mathtt{q}!\mathtt{m}\langle e\rangle.P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft Q \mid \mathtt{q} \triangleleft h_{\mathtt{q}} \mid \mathbb{M} \\
& \rightarrow\; \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft Q \mid \mathtt{q} \triangleleft h_{\mathtt{q}} \cdot (\mathtt{p}, \mathtt{m}(v)) \mid \mathbb{M} \qquad (e \downarrow v, h_{\mathtt{q}} \neq \oslash) \\
\text{[R-SEND-}\lightning\text{]} & \mathtt{p} \triangleleft \mathtt{q}!\mathtt{m}\langle e\rangle.P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M} \;\rightarrow\; \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M} \\
\text{[R-RCV]} & \mathtt{p} \triangleleft \textstyle\sum_{i\in I} \mathtt{q}?\mathtt{m}_i(x_i).P_i \mid \mathtt{p} \triangleleft (\mathtt{q}, \mathtt{m}_k(v)) \cdot h_{\mathtt{p}} \mid \mathbb{M} \rightarrow\; \mathtt{p} \triangleleft P_k\{v/x_k\} \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathbb{M} \quad (k \in I) \\
\text{[R-RCV-}\odot\text{]} & \mathtt{p} \triangleleft \textstyle\sum_{i\in I} \mathtt{q}?\mathtt{m}_i(x_i).P_i \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M} \\
& \rightarrow\; \mathtt{p} \triangleleft P_k \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M} \qquad (k \in I, \mathtt{m}_k = \mathsf{crash}, \nexists \mathtt{m}, v : (\mathtt{q}, \mathtt{m}(v)) \in h_{\mathtt{p}}) \\
\text{[R-COND-T]} & \mathtt{p} \triangleleft \mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q \mid \mathtt{p} \triangleleft h \mid \mathbb{M} \;\rightarrow\; \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h \mid \mathbb{M} \qquad (e \downarrow \mathtt{true}) \\
\text{[R-COND-F]} & \mathtt{p} \triangleleft \mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q \mid \mathtt{p} \triangleleft h \mid \mathbb{M} \;\rightarrow\; \mathtt{p} \triangleleft Q \mid \mathtt{p} \triangleleft h \mid \mathbb{M} \qquad (e \downarrow \mathtt{false}) \\
\text{[R-STRUCT]} & \mathbb{M}_1 \Rrightarrow \mathbb{M}_1' \;\text{ and }\; \mathbb{M}_1' \rightarrow \mathbb{M}_2' \;\text{ and }\; \mathbb{M}_2' \Rrightarrow \mathbb{M}_2 \quad\Longrightarrow\quad \mathbb{M}_1 \rightarrow \mathbb{M}_2
\end{array}$$

FIGURE 4. Reduction relation on sessions with crash-stop failures.

$$h_1 \cdot (\mathtt{q}_i, \mathtt{m}_i(v_i)) \cdot (\mathtt{q}_j, \mathtt{m}_j(v_j)) \cdot h_2 \Rrightarrow h_1 \cdot (\mathtt{q}_j, \mathtt{m}_j(v_j)) \cdot (\mathtt{q}_i, \mathtt{m}_i(v_i)) \cdot h_2 \quad (\text{if } i \neq j,\ i,j \in \{1,2\},\ \mathtt{q}_1 \neq \mathtt{q}_2)$$

$$\epsilon \cdot h \Rrightarrow h \quad h \cdot \epsilon \Rrightarrow h \quad h_1 \cdot (h_2 \cdot h_3) \Rrightarrow (h_1 \cdot h_2) \cdot h_3 \quad (h_1 \cdot h_2) \cdot h_3 \Rrightarrow h_1 \cdot (h_2 \cdot h_3)$$

$$\mathtt{p} \triangleleft \mathbf{0} \mid \mathtt{p} \triangleleft \epsilon \mid \mathbb{M} \Rrightarrow \mathbb{M} \quad \mu X.P \Rrightarrow P\{\mu X.P/X\} \quad \mathbb{M} \Rrightarrow \mathbb{M}' \text{ and } \mathbb{M}' \Rrightarrow \mathbb{M}'' \implies \mathbb{M} \Rrightarrow \mathbb{M}''$$

$$\textstyle\prod_{i\in I}(\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i) \Rrightarrow \prod_{j\in J}(\mathtt{p}_j \triangleleft P_j \mid \mathtt{p}_j \triangleleft h_j) \quad (\text{if } I \text{ is a permutation of } J)$$

$$P \Rrightarrow Q \;\text{ and }\; h_1 \Rrightarrow h_2 \quad\Longrightarrow\quad \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h_1 \mid \mathbb{M} \Rrightarrow \mathtt{p} \triangleleft Q \mid \mathtt{p} \triangleleft h_2 \mid \mathbb{M}$$

FIGURE 5. Structural precongruence rules for queues, processes, and sessions.

from $\mathtt{q}$ in the incoming queue, which engages the usual rule [R-RECV]; if none exists and $\mathtt{q}$ has crashed ($\mathtt{q} \triangleleft \lightning$), then the crash handling branch in the input process at $\mathtt{p}$ can activate. We draw attention to the interesting fact that [R-RECV] may engage even if $\mathtt{q}$ has crashed, in cases where a message from $\mathtt{q}$ in the incoming queue may be consumed.

Example 3.2. We now illustrate our operational semantics of sessions with an example. Consider the session:

$$\mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft \epsilon \mid \mathtt{q} \triangleleft Q \mid \mathtt{q} \triangleleft \epsilon$$

where $P = \mathtt{q}!\mathtt{m}\langle\text{“}\mathtt{abc}\text{”}\rangle.P'$ with $P' = \sum \left\{ \begin{array}{l} \mathtt{q}?\mathtt{m}'(x).\mathbf{0} \\ \mathtt{q}?\mathsf{crash}.\mathbf{0} \end{array} \right\}$, and $Q = \sum \left\{ \begin{array}{l} \mathtt{p}?\mathtt{m}(x).Q' \\ \mathtt{p}?\mathsf{crash}.\mathbf{0} \end{array} \right\}$ with $Q' = \mathtt{p}!\mathtt{m}'\langle 42\rangle.\mathbf{0}$.

In this session, the process Q for $\mathtt{q}$ receives a message sent from $\mathtt{p}$ to $\mathtt{q}$; the process P for $\mathtt{p}$ sends a message from $\mathtt{p}$ to $\mathtt{q}$, and then receives a message sent from $\mathtt{q}$ to $\mathtt{p}$.

On a successful reduction (without crashes), we have:

B	::=	$\mathsf{nat} \mid \mathsf{int} \mid \mathsf{bool} \mid \mathsf{str} \mid \mathsf{unit} \mid \ldots$	Basic types
G	::=	$\mathtt{p}\to\mathtt{q}^{\dagger}\colon\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I}$	Transmission
	$\mid$	$\mathtt{p}^{\dagger}\rightsquigarrow\mathtt{q}\colon j\,\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I}\ (j\in I)$	Transmission en route (Runtime)
	$\mid$	$\mu\mathbf{t}.G \mid \mathbf{t} \mid \mathsf{end}$	Recursion, Type variable, Termination
$\dagger$	::=	$\cdot \mid \lightning$	Crash annotation (Runtime)
S,T	::=	$\mathtt{p}\&\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I}$	External choice (Receive)
	$\mid$	$\mathtt{p}\oplus\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I}$	Internal choice (Send)
	$\mid$	$\mu\mathbf{t}.T \mid \mathbf{t}$	Recursion, Type variable
	$\mid$	$\mathsf{end} \mid \mathsf{stop}$	Termination, Crash (Runtime)

FIGURE 6. Syntax of basic types, global types, and local types. Runtime types are shaded.

$$\begin{array}{rcl}\mathtt{p}\triangleleft P \mid \mathtt{p}\triangleleft\epsilon \mid \mathtt{q}\triangleleft Q \mid \mathtt{q}\triangleleft\epsilon & \to & \mathtt{p}\triangleleft P' \mid \mathtt{p}\triangleleft\epsilon \mid \mathtt{q}\triangleleft Q \mid \mathtt{q}\triangleleft(\mathtt{p},\mathtt{m}(\text{“}\mathtt{abc}\text{”})) \\ & \to & \mathtt{p}\triangleleft P' \mid \mathtt{p}\triangleleft\epsilon \mid \mathtt{q}\triangleleft Q' \mid \mathtt{q}\triangleleft\epsilon \\ & \to & \mathtt{p}\triangleleft P' \mid \mathtt{p}\triangleleft(\mathtt{q},\mathtt{m}'(42)) \mid \mathtt{q}\triangleleft\mathbf{0} \mid \mathtt{q}\triangleleft\epsilon \\ & \to & \mathtt{p}\triangleleft\mathbf{0} \mid \mathtt{p}\triangleleft\epsilon \mid \mathtt{q}\triangleleft\mathbf{0} \mid \mathtt{q}\triangleleft\epsilon\end{array}$$

Let $\mathscr{R}=\emptyset$ (i.e. each role is unreliable). Suppose that P crashes before sending, which leads to the reduction:

$$\begin{array}{rcl}\mathtt{p}\triangleleft P \mid \mathtt{p}\triangleleft\epsilon \mid \mathtt{q}\triangleleft Q \mid \mathtt{q}\triangleleft\epsilon & \to_{\mathscr{R}} & \mathtt{p}\triangleleft\lightning \mid \mathtt{p}\triangleleft\oslash \mid \mathtt{q}\triangleleft Q \mid \mathtt{q}\triangleleft\epsilon \\ & \to & \mathtt{p}\triangleleft\lightning \mid \mathtt{p}\triangleleft\oslash \mid \mathtt{q}\triangleleft\mathbf{0} \mid \mathtt{q}\triangleleft\epsilon\end{array}$$

We can observe that when the output (sending) process P located at an unreliable role $\mathtt{p}$ crashes (by [R-$\lightning$]), $\mathtt{p}$ also crashes ($\mathtt{p}\triangleleft\lightning$), with an unavailable incoming message queue ($\mathtt{p}\triangleleft\oslash$). Subsequently, the input (receiving) process Q located at $\mathtt{q}$ can detect and handle the crash by [R-RCV-$\odot$] via its handling branch.

4. Asynchronous Multiparty Session Types with Crash-Stop Semantics

In this section, we present our asynchronous multiparty session types with crash-stop semantics. We give an overview of global and local types with crashes in Section 4.1, including syntax, projection, and subtyping. Our key additions to the classic theory are *crash handling branches* in both global and local types, and a special local type stop to denote crashed processes. We give a Labelled Transition System (LTS) semantics to both global types (Section 4.2) and configurations (i.e. a collection of local types and point-to-point communication queues, Section 4.3). We discuss alternative design options of modelling crash-stop failures in Section 4.4. We relate the two semantics in Section 4.5, and show that a configuration obtained via projection is safe, deadlock-free, and live in Section 4.6.

4.1. Global and Local Types with Crash-Stop Failures.

The top-down methodology begins with *global types* to provide an overview of the communication between a number of *roles* ($\mathtt{p},\mathtt{q},\mathtt{s},\mathtt{t},\ldots$), belonging to a (fixed) set $\mathcal{R}$. On the other end, we use *local types* to describe how a *single* role communicates with other roles from a local perspective, and they are obtained via *projection* from a global type. We give the syntax of both global and local types in Fig. 6, which are similar to syntax used in [SY19, BSYZ22].

Basic Types. *Basic types* (types for payloads) are taken from a set $\mathcal{B}$, and describe the types of values such as natural numbers, integers, booleans, strings, and units.

Global Types. *Global types* are ranged over $G, G', G_i, \ldots$, and describe the behaviour for all roles from a bird's eye view. The syntactic constructs shown in shade are *runtime* constructs, which are not used for describing a system at design-time, but for describing the state of a system during execution.

We explain each global type syntactic construct: a transmission $\mathtt{p}\to\mathtt{q}^{\dagger}\colon\{\mathtt{m_i}(B_i).G_i\}_{i\in I}$ denotes a message from role $\mathtt{p}$ to role $\mathtt{q}$ (with possible crash annotations), with labels $\mathtt{m}_i$, payload types B_i, and continuations G_i, where i is taken from an index set I. We require that the index set be non-empty ($I \neq \emptyset$), labels $\mathtt{m}_i$ be pair-wise distinct and taken from a fixed set of labels $\mathcal{M}$, and self receptions be excluded (i.e. $\mathtt{p} \neq \mathtt{q}$), as is standard in session type literature. Additionally, we require that the special crash label (explained later) not be the only label in a transmission, i.e. $\{\mathtt{m}_i \mid i \in I\} \neq \{\mathsf{crash}\}$. A transmission en route $\mathtt{p}^{\dagger}\rightsquigarrow\mathtt{q}{:}j\,\{\mathtt{m_i}(B_i).G_i\}_{i\in I}$ is a runtime construct representing a message $\mathtt{m}_j$ (index j) sent by $\mathtt{p}$, and yet to be received by $\mathtt{q}$. Recursive types are represented via $\mu\mathtt{t}.G$ and $\mathtt{t}$, where contractive requirements apply [Pie02, §21.8]. The type end describes a terminated type (omitted where unambiguous).

To model crashes and crash handling, we use crash annotations $\lightning$ and crash handling branches: a *crash annotation* $\lightning$, a new addition in this work, marks a *crashed* role (only used in the *runtime syntax*), and we omit annotations for *live* (or *active*) roles, i.e. $\mathtt{p}$ is a live role, $\mathtt{p}^{\lightning}$ is a crashed role, and $\mathtt{p}^{\dagger}$ represents a possibly crashed role, namely either $\mathtt{p}$ or $\mathtt{p}^{\lightning}$. We use a special label crash for handling crashes: this continuation denotes the protocol to follow when the sender of a message is detected to have crashed by the receiver. The special label acts as a 'pseudo'-message: when a sender role crashes, the receiver can select the 'pseudo'-message to enter crash handling.

Definition 4.1 (Set of Live and Crashed Roles)**.** The set of *live* roles in a global type G, denoted $\mathrm{roles}(G)$, and the set of *crashed* roles, denoted $\mathrm{roles}^{\lightning}(G)$, are defined inductively as:

$$\mathrm{roles}(\mathtt{p}\to\mathtt{q}\colon\{\mathtt{m_i}(B_i).G_i\}_{i\in I}) = \{\mathtt{p},\mathtt{q}\}\cup\bigcup_{i\in I}\mathrm{roles}(G_i) \qquad \mathrm{roles}^{\lightning}(\mathtt{p}\to\mathtt{q}\colon\{\mathtt{m_i}(B_i).G_i\}_{i\in I}) = \bigcup_{i\in I}\mathrm{roles}^{\lightning}(G_i)$$

$$\mathrm{roles}(\mathtt{p}\to\mathtt{q}^{\lightning}\colon\{\mathtt{m_i}(B_i).G_i\}_{i\in I}) = \{\mathtt{p}\}\cup\bigcup_{i\in I}\mathrm{roles}(G_i) \qquad \mathrm{roles}^{\lightning}(\mathtt{p}\to\mathtt{q}^{\lightning}\colon\{\mathtt{m_i}(B_i).G_i\}_{i\in I}) = \{\mathtt{q}\}\cup\bigcup_{i\in I}\mathrm{roles}^{\lightning}(G_i)$$

$$\mathrm{roles}(\mathtt{p}^{\dagger}\rightsquigarrow\mathtt{q}{:}j\,\{\mathtt{m_i}(B_i).G_i\}_{i\in I}) = \{\mathtt{q}\}\cup\bigcup_{i\in I}\mathrm{roles}(G_i) \qquad \mathrm{roles}^{\lightning}(\mathtt{p}^{\dagger}\rightsquigarrow\mathtt{q}{:}j\,\{\mathtt{m_i}(B_i).G_i\}_{i\in I}) = \bigcup_{i\in I}\mathrm{roles}^{\lightning}(G_i)$$

$$\mathrm{roles}(\mathsf{end}) = \mathrm{roles}(\mathtt{t}) = \emptyset \qquad \mathrm{roles}^{\lightning}(\mathsf{end}) = \mathrm{roles}^{\lightning}(\mathtt{t}) = \emptyset$$

$$\mathrm{roles}(\mu\mathtt{t}.G) = \mathrm{roles}(G\{^{\mu\mathtt{t}.G}/_{\mathtt{t}}\}) \qquad \mathrm{roles}^{\lightning}(\mu\mathtt{t}.G) = \mathrm{roles}^{\lightning}(G\{^{\mu\mathtt{t}.G}/_{\mathtt{t}}\})$$

Remark 4.2. Even if the sender $\mathtt{p}$ occurs as crashed during a communication in transit, it is not considered as crashed unless it appears crashed in a continuation:

$$\mathrm{roles}^{\lightning}(\mathtt{p}^{\lightning}\rightsquigarrow\mathtt{q}{:}j\,\{\mathtt{m_i}(B_i).G_i\}_{i\in I}) = \bigcup_{i\in I}\mathrm{roles}^{\lightning}(G_i).$$

Local Types. *Local types* (or *session types*) are ranged over by $S, T, U, \ldots$, and describe the behaviour of a single role. An internal choice (selection) $\mathtt{p}\oplus\{\mathtt{m_i}(B_i).T_i\}_{i\in I}$ (resp. an external choice (branching) $\mathtt{p}\&\{\mathtt{m_i}(B_i).T_i\}_{i\in I}$) indicates that the *current* role is to *send* to (resp. *receive* from) the role $\mathtt{p}$. Similarly to global types, we require pairwise-distinct, non-empty labels in local types. Moreover, we require that the crash label not appear in *internal* choices, reflecting that a crash 'pseudo'-message can never be sent; and that singleton crash labels are not permitted in external choices. The type end indicates a *successful* termination

(omitted where unambiguous), and recursive types follow a similar fashion to global types. We use a new *runtime* type stop to denote crashes.

Typographical Conventions. Whenever a payload type B is insignificant, we choose to omit it from the global or local type. This is frequently the case when we discuss the branches with crash (in a global type) or crash (in a local types) labels, where the payload type is irrelevant.

Projection. *Projection* gives the local type of a participating role in a global type, defined as a *partial* function that takes a global type G and a role $\mathtt{p}$ to project on, and returns a local type, given by Definition 4.3.

Definition 4.3 (Global Type Projection)**.** The *projection of a global type G onto a role $\mathtt{p}$*, with respect to a set of *reliable* roles $\mathscr{R}$, written $G \upharpoonright_{\mathscr{R}} \mathtt{p}$, is:

$$\left(\mathtt{q}\to\mathtt{r}^{\dagger}\colon\{\mathtt{m_i}(B_i).G_i\}_{i\in I}\right)\upharpoonright_{\mathscr{R}}\mathtt{p}=\begin{cases}\mathtt{r}\oplus\{\mathtt{m_i}(B_i).(G_i\upharpoonright_{\mathscr{R}}\mathtt{p})\}_{i\in\{j\in I\,|\,\mathtt{m}_j\neq\mathsf{crash}\}} & \text{if } \mathtt{p}=\mathtt{q}\\ \mathtt{q}\&\{\mathtt{m_i}(B_i).(G_i\upharpoonright_{\mathscr{R}}\mathtt{p})\}_{i\in I} & \text{if } \mathtt{p}=\mathtt{r}, \text{ and } \mathtt{q}\notin\mathscr{R} \text{ implies } \exists k\in I:\mathtt{m}_k=\mathsf{crash}\\ \bigsqcap_{i\in I}G_i\upharpoonright_{\mathscr{R}}\mathtt{p} & \text{if } \mathtt{p}\neq\mathtt{q}, \text{ and } \mathtt{p}\neq\mathtt{r}\end{cases}$$

$$\left(\mathtt{q}^{\dagger}\rightsquigarrow\mathtt{r}{:}j\,\{\mathtt{m_i}(B_i).G_i\}_{i\in I}\right)\upharpoonright_{\mathscr{R}}\mathtt{p}=\begin{cases}G_j\upharpoonright_{\mathscr{R}}\mathtt{p} & \text{if } \mathtt{p}=\mathtt{q}\\ \mathtt{q}\&\{\mathtt{m_i}(B_i).(G_i\upharpoonright_{\mathscr{R}}\mathtt{p})\}_{i\in I} & \text{if } \mathtt{p}=\mathtt{r}, \text{ and } \mathtt{q}\notin\mathscr{R} \text{ implies } \exists k\in I:\mathtt{m}_k=\mathsf{crash}\\ \bigsqcap_{i\in I}G_i\upharpoonright_{\mathscr{R}}\mathtt{p} & \text{if } \mathtt{p}\neq\mathtt{q}, \text{ and } \mathtt{p}\neq\mathtt{r}\end{cases}$$

$$(\mu\mathbf{t}.G)\upharpoonright_{\mathscr{R}}\mathtt{p}=\begin{cases}\mu\mathbf{t}.(G\upharpoonright_{\mathscr{R}}\mathtt{p}) & \text{if } \mathtt{p}\in G \text{ or } \mathrm{fv}(\mu\mathbf{t}.G)\neq\emptyset\\ \mathsf{end} & \text{otherwise}\end{cases}\qquad \begin{array}{r}\mathbf{t}\upharpoonright_{\mathscr{R}}\mathtt{p}=\mathbf{t}\\ \mathsf{end}\upharpoonright_{\mathscr{R}}\mathtt{p}=\mathsf{end}\end{array}$$

where $\bigsqcap$ is the *merge operator for session types* (*full merging*):

$$\begin{gathered}\mathtt{p}\&\{\mathtt{m_i}(B_i).S'_i\}_{i\in I}\sqcap\mathtt{p}\&\{\mathtt{m_j}(B_j).T'_j\}_{j\in J}\\ =\ \mathtt{p}\&\{\mathtt{m_k}(B_k).(S'_k\sqcap T'_k)\}_{k\in I\cap J}\,\&\,\mathtt{p}\&\{\mathtt{m_i}(B_i).S'_i\}_{i\in I\setminus J}\,\&\,\mathtt{p}\&\{\mathtt{m_j}(B_j).T'_j\}_{j\in J\setminus I}\\ \mathtt{p}\oplus\{\mathtt{m_i}(B_i).S'_i\}_{i\in I}\sqcap\mathtt{p}\oplus\{\mathtt{m_i}(B_i).T'_i\}_{i\in I}\ =\ \mathtt{p}\oplus\{\mathtt{m_i}(B_i).(S'_i\sqcap T'_i)\}_{i\in I}\\ \mu\mathbf{t}.S\sqcap\mu\mathbf{t}.T=\mu\mathbf{t}.(S\sqcap T)\qquad \mathbf{t}\sqcap\mathbf{t}=\mathbf{t}\qquad \mathsf{end}\sqcap\mathsf{end}=\mathsf{end}\end{gathered}$$

We parameterise our theory on a (fixed) set of *reliable* roles $\mathscr{R}$, i.e. roles assumed to *never crash*, for modelling optional reliability assumptions: if $\mathscr{R}=\emptyset$, we assume every role to be unreliable and susceptible to crash; if $\mathrm{roles}(G)\subseteq\mathscr{R}$, we assume every role in G to be reliable, and we simulate the results from the original MPST theory without crashes[2]. We base our definition of projection on the standard definition [SY19], but include more (highlighted) cases to account for reliable roles, crash branches, and runtime global types.

When projecting a transmission from $\mathtt{q}$ to $\mathtt{r}$, we remove the crash label from the internal choice at $\mathtt{q}$, reflecting our model that a crash 'pseudo'-message cannot be sent. Dually, we require a crash label to be present in the external choice at $\mathtt{r}$ – unless the sender role $\mathtt{q}$ is assumed to be reliable. Our definition of projection enforces that transmissions, whenever an unreliable role is the sender ($\mathtt{q}\notin\mathscr{R}$), *must include* a crash handling branch ($\exists k\in I:\mathtt{m}_k=\mathsf{crash}$). This requirement ensures that the receiving role $\mathtt{r}$ can *always* handle crashes whenever it happens, so that processes are not stuck when crashes occur. We

[2]Here we consider the original MPST theory without delegation (session passing).

explain how these requirements help us achieve various properties by projection, such as safety, deadlock-freedom, and liveness, in Section 4.6. The rest of the rules, including the definition of the merge operator, are taken from the literature [SY19, vGHH21], without much modification.

Subtyping. We define a *subtyping* relation $\leqslant$ on local types in Definition 4.4, which will be used later to relate the semantics of global types and configurations in Section 4.5. Our subtyping relation is mostly standard [SY19, Def. 2.5], except for the (highlighted) addition of the rule [Sub-stop] and extra requirements in [Sub-&]. In [Sub-&], we add two additional requirements: *(1)* the supertype cannot be a 'pure' crash handling branch; and *(2)* if the subtype has a crash handling branch, then the supertype must also have one. For simplicity, we do not consider subtyping on basic types B.

Definition 4.4 (Subtyping)**.** The subtyping relation $\leqslant$ is coinductively defined:

$$\frac{}{\mathsf{end} \leqslant \mathsf{end}}\ [\text{Sub-end}] \qquad \frac{\forall i \in I \quad T_i \leqslant T_i' \quad \{\mathtt{m}_k \mid k \in I\} \neq \{\mathsf{crash}\} \quad \nexists j \in J : \mathtt{m}_j = \mathsf{crash}}{\mathtt{p}\&\{\mathtt{m}_i(B_i).T_i\}_{i\in I\cup J} \leqslant \mathtt{p}\&\{\mathtt{m}_i(B_i).T_i'\}_{i\in I}}\ [\text{Sub-\&}]$$

$$\frac{}{\mathsf{stop} \leqslant \mathsf{stop}}\ [\text{Sub-stop}] \qquad \frac{\forall i \in I \quad T_i \leqslant T_i'}{\mathtt{p}\oplus\{\mathtt{m}_i(B_i).T_i\}_{i\in I} \leqslant \mathtt{p}\oplus\{\mathtt{m}_i(B_i).T_i'\}_{i\in I\cup J}}\ [\text{Sub-}\oplus]$$

$$\frac{T\{\mu \mathbf{t}.T/\mathbf{t}\} \leqslant T'}{\mu \mathbf{t}.T \leqslant T'}\ [\text{Sub-}\mu\text{L}] \qquad \frac{T \leqslant T'\{\mu \mathbf{t}.T'/\mathbf{t}\}}{T \leqslant \mu \mathbf{t}.T'}\ [\text{Sub-}\mu\text{R}]$$

As standard, our subtyping relation is reflexive and transitive. Additionally, the properties of subtyping related to merges are demonstrated in Lemmas 4.6, 4.7, and 4.8. The proofs are available in Appendix A.2.

Lemma 4.5 (Reflexivity and Transitivity of Subtyping)**.** *The subtyping relation $\leqslant$ is reflexive and transitive.*

Lemma 4.6. *Given a collection of mergable local types T_i $(i \in I)$. For all $j \in I$, $\bigsqcap_{i\in I} T_i \leqslant T_j$ holds.*

Lemma 4.7. *Given a collection of mergable local types T_i $(i \in I)$. If for all $i \in I$, $S \leqslant T_i$ for some local type S, then $S \leqslant \bigsqcap_{i\in I} T_i$.*

Lemma 4.8. *Given two collections of mergable local types S_i, T_i $(i \in I)$. If for all $i \in I$, $S_i \leqslant T_i$, then $\bigsqcap_{i\in I} S_i \leqslant \bigsqcap_{i\in I} T_i$.*

4.2. **Crash-Stop Semantics of Global Types.** We now give a Labelled Transition System (LTS) semantics to global types, with crash-stop semantics. To this end, we first introduce some auxiliary definitions. We define the transition labels in Definition 4.9, which are also used in the LTS semantics of configurations (later in Section 4.3).

Definition 4.9 (Transition Labels)**.** Let α be a transition label of the form:

α	::=	$\mathtt{p}\&\mathtt{q}:\mathtt{m}(B)$	($\mathtt{p}$ receives $\mathtt{m}(B)$ from $\mathtt{q}$)	$\mathtt{p}\oplus\mathtt{q}:\mathtt{m}(B)$	($\mathtt{p}$ sends $\mathtt{m}(B)$ to $\mathtt{q}$)
	$\mid$	$\mathtt{p}\lightning$	($\mathtt{p}$ crashes)	$\mathtt{p}\odot\mathtt{q}$	($\mathtt{p}$ detects the crash of $\mathtt{q}$)

The subject of a transition label, written $\mathrm{subj}(\alpha)$, is defined as:

$$\mathrm{subj}(\mathtt{p}\&\mathtt{q}:\mathtt{m}(B)) = \mathrm{subj}(\mathtt{p}\oplus\mathtt{q}:\mathtt{m}(B)) = \mathrm{subj}(\mathtt{p}\lightning) = \mathrm{subj}(\mathtt{p}\odot\mathtt{q}) = \mathtt{p}.$$

The labels $\mathtt{p}\oplus\mathtt{q}:\mathtt{m}(B)$ and $\mathtt{p}\&\mathtt{q}:\mathtt{m}(B)$ describe sending and receiving actions respectively. The crash of a role $\mathtt{p}$ is denoted by the label $\mathtt{p}\lightning$, and the detection of a crash by label $\mathtt{p}\odot\mathtt{q}$: we model crash detection at *reception*, the label contains a *detecting* role $\mathtt{p}$ and a *crashed* role $\mathtt{q}$. We use these labels when giving the semantics for global types and configurations.

We then define an operator to *remove* a role from a global type in Definition 4.10: the intuition is to remove any interaction of a crashed role from the given global type. When a role has crashed, we remove it by attaching a *crashed annotation*, and removing infeasible actions, e.g. when the sender and receiver of a transmission have both crashed. The removal operator is a partial function that takes a global type G and a live role $\mathtt{r}$ ($\mathtt{r} \in \mathrm{roles}(G)$) and gives a global type $G\lightning\mathtt{r}$.

Definition 4.10 (Role Removal)**.** The removal of a live role $\mathtt{p}$ in a global type G, written $G\lightning\mathtt{p}$, is defined as follows:

$$(\mathtt{p}\to\mathtt{q}\colon\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I})\lightning\mathtt{r} = \begin{cases} \mathtt{p}^\lightning \rightsquigarrow \mathtt{q}{:}j\,\{\mathtt{m}_\mathtt{i}(B_i).(G_i\lightning\mathtt{r})\}_{i\in I} & \text{if } \mathtt{p}=\mathtt{r} \text{ and } \exists j\in I : \mathtt{m}_j = \mathsf{crash} \\ \mathtt{p}\to\mathtt{q}^\lightning\colon\{\mathtt{m}_\mathtt{i}(B_i).(G_i\lightning\mathtt{r})\}_{i\in I} & \text{if } \mathtt{q}=\mathtt{r} \\ \mathtt{p}\to\mathtt{q}\colon\{\mathtt{m}_\mathtt{i}(B_i).(G_i\lightning\mathtt{r})\}_{i\in I} & \text{if } \mathtt{p}\neq\mathtt{r} \text{ and } \mathtt{q}\neq\mathtt{r} \end{cases}$$

$$(\mathtt{p}\rightsquigarrow\mathtt{q}{:}j\,\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I})\lightning\mathtt{r} = \begin{cases} \mathtt{p}^\lightning \rightsquigarrow \mathtt{q}{:}j\,\{\mathtt{m}_\mathtt{i}(B_i).(G_i\lightning\mathtt{r})\}_{i\in I} & \text{if } \mathtt{p}=\mathtt{r} \\ G_j \lightning \mathtt{r} & \text{if } \mathtt{q}=\mathtt{r} \\ \mathtt{p}\rightsquigarrow\mathtt{q}{:}j\,\{\mathtt{m}_\mathtt{i}(B_i).(G_i\lightning\mathtt{r})\}_{i\in I} & \text{if } \mathtt{p}\neq\mathtt{r} \text{ and } \mathtt{q}\neq\mathtt{r} \end{cases}$$

$$(\mathtt{p}\to\mathtt{q}^\lightning\colon\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I})\lightning\mathtt{r} = \begin{cases} G_j \lightning \mathtt{r} & \text{if } \mathtt{p}=\mathtt{r} \text{ and } \exists j\in I : \mathtt{m}_j = \mathsf{crash} \\ \mathtt{p}\to\mathtt{q}^\lightning\colon\{\mathtt{m}_\mathtt{i}(B_i).(G_i\lightning\mathtt{r})\}_{i\in I} & \text{if } \mathtt{p}\neq\mathtt{r} \text{ and } \mathtt{q}\neq\mathtt{r} \end{cases}$$

$$(\mathtt{p}^\lightning\rightsquigarrow\mathtt{q}{:}j\,\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I})\lightning\mathtt{r} = \begin{cases} G_j \lightning \mathtt{r} & \text{if } \mathtt{q}=\mathtt{r} \\ \mathtt{p}^\lightning\rightsquigarrow\mathtt{q}{:}j\,\{\mathtt{m}_\mathtt{i}(B_i).(G_i\lightning\mathtt{r})\}_{i\in I} & \text{if } \mathtt{p}\neq\mathtt{r} \text{ and } \mathtt{q}\neq\mathtt{r} \end{cases}$$

$$(\mu\mathtt{t}.G)\lightning\mathtt{r} = \begin{cases} \mu\mathtt{t}.(G\lightning\mathtt{r}) & \text{if } \mathrm{fv}(\mu\mathtt{t}.G)\neq\emptyset \text{ or } \mathrm{roles}(G\lightning\mathtt{r})\neq\emptyset \\ \mathsf{end} & \text{otherwise} \end{cases}$$

$$\mathtt{t}\lightning\mathtt{r} = \mathtt{t} \qquad\qquad \mathsf{end}\lightning\mathtt{r} = \mathsf{end}$$

We now explain the definition in detail. For simple cases, the removal of a role $G\lightning\mathtt{r}$ attaches crash annotations $\lightning$ on all occurrences of the removed role $\mathtt{r}$ throughout global type G inductively.

We draw attention to some interesting cases: when we remove the sender role $\mathtt{p}$ from a transmission prefix $\mathtt{p}\to\mathtt{q}$, the result is a 'pseudo'-transmission en route prefix $\mathtt{p}^\lightning \rightsquigarrow \mathtt{q} : j$ where $\mathtt{m}_j = \mathsf{crash}$. This enables the receiver $\mathtt{q}$ to 'receive' the special crash after the crash of $\mathtt{p}$, hence triggering the crash handling branch. Recall that our definition of projection requires that a crash handling branch be present whenever a crash may occur ($\mathtt{q}\notin\mathscr{R}$).

When we remove the sender role $\mathtt{p}$ from a transmission en route prefix $\mathtt{p}\rightsquigarrow\mathtt{q} : j$, the result *retains* the index j that was selected by $\mathtt{p}$, instead of the index associated with crash handling. This is crucial to our crash modelling: when a role crashes, the messages that the role *has sent* to other roles are still available. We discuss alternative models later in Section 4.4.

In other cases, where removing the role $\mathtt{r}$ would render a transmission (regardless of being en route or not) meaningless, e.g. both sender and receiver have crashed, we simply remove the prefix entirely.

Example 4.11. We remove role $\mathtt{C}$ in the global type G in Equation (2.1) (defined in Section 2).

$$\frac{\mathtt{p} \notin \mathscr{R} \quad \mathtt{p} \in \mathrm{roles}(G) \quad G \neq \mu \mathtt{t}.G'}{\langle \mathscr{C}; G \rangle \xrightarrow{\mathtt{p}\lightning}_{\mathscr{R}} \langle \mathscr{C} \cup \{\mathtt{p}\}; G \lightning \mathtt{p} \rangle} \text{[GR-}\lightning\text{]} \qquad \frac{\langle \mathscr{C}; G\{\mu \mathtt{t}.G/\mathtt{t}\} \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle}{\langle \mathscr{C}; \mu \mathtt{t}.G \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle} \text{[GR-}\mu\text{]}$$

$$\frac{j \in I \quad \mathtt{m}_j \neq \mathsf{crash}}{\langle \mathscr{C}; \mathtt{p} \to \mathtt{q}\colon \{\mathtt{m}_i(B_i).G'_i\}_{i \in I} \rangle \xrightarrow{\mathtt{p} \oplus \mathtt{q}:\mathtt{m}_j(B_j)}_{\mathscr{R}} \langle \mathscr{C}; \mathtt{p} \rightsquigarrow \mathtt{q}\colon j \, \{\mathtt{m}_i(B_i).G'_i\}_{i \in I} \rangle} \text{[GR-}\oplus\text{]}$$

$$\frac{j \in I \quad \mathtt{m}_j \neq \mathsf{crash}}{\langle \mathscr{C}; \mathtt{p}^\dagger \rightsquigarrow \mathtt{q}\colon j \, \{\mathtt{m}_i(B_i).G'_i\}_{i \in I} \rangle \xrightarrow{\mathtt{q} \& \mathtt{p}:\mathtt{m}_j(B_j)}_{\mathscr{R}} \langle \mathscr{C}; G'_j \rangle} \text{[GR-}\&\text{]}$$

$$\frac{j \in I \quad \mathtt{m}_j = \mathsf{crash}}{\langle \mathscr{C}; \mathtt{p}^\lightning \rightsquigarrow \mathtt{q}\colon j \, \{\mathtt{m}_i(B_i).G'_i\}_{i \in I} \rangle \xrightarrow{\mathtt{q} \odot \mathtt{p}}_{\mathscr{R}} \langle \mathscr{C}; G'_j \rangle} \text{[GR-}\odot\text{]}$$

$$\frac{j \in I \quad \mathtt{m}_j \neq \mathsf{crash}}{\langle \mathscr{C}; \mathtt{p} \to \mathtt{q}^\lightning\colon \{\mathtt{m}_i(B_i).G'_i\}_{i \in I} \rangle \xrightarrow{\mathtt{p} \oplus \mathtt{q}:\mathtt{m}_j(B_j)}_{\mathscr{R}} \langle \mathscr{C}; G'_j \rangle} \text{[GR-}\lightning\mathtt{m}\text{]}$$

$$\frac{\forall i \in I : \langle \mathscr{C}; G'_i \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G''_i \rangle \quad \mathrm{subj}(\alpha) \notin \{\mathtt{p}, \mathtt{q}\}}{\langle \mathscr{C}; \mathtt{p} \to \mathtt{q}^\dagger\colon \{\mathtt{m}_i(B_i).G'_i\}_{i \in I} \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; \mathtt{p} \to \mathtt{q}^\dagger\colon \{\mathtt{m}_i(B_i).G''_i\}_{i \in I} \rangle} \text{[GR-Ctx-I]}$$

$$\frac{\forall i \in I : \langle \mathscr{C}; G'_i \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G''_i \rangle \quad \mathrm{subj}(\alpha) \neq \mathtt{q}}{\langle \mathscr{C}; \mathtt{p}^\dagger \rightsquigarrow \mathtt{q}\colon j \, \{\mathtt{m}_i(B_i).G'_i\}_{i \in I} \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; \mathtt{p}^\dagger \rightsquigarrow \mathtt{q}\colon j \, \{\mathtt{m}_i(B_i).G''_i\}_{i \in I} \rangle} \text{[GR-Ctx-II]}$$

FIGURE 7. Global type reduction rules.

$$G \lightning \mathtt{C} = \mathtt{L} \to \mathtt{I}\colon \mathtt{trigger}.\mathtt{C}^\lightning \rightsquigarrow \mathtt{I}\colon \mathtt{crash}.\mathtt{I} \to \mathtt{L}\colon \mathtt{fatal}.\mathtt{end}$$

Role C now carries a crash annotation $\lightning$ in the resulting global type, denoting it has crashed. Crash annotations change the reductions available for global types.

We now give a Labelled Transition System (LTS) semantics to a global type G, by defining the semantics with a tuple $\langle \mathscr{C}; G \rangle$, where $\mathscr{C}$ is a set of *crashed* roles, with all roles in $\mathscr{C}$ belonging to the fixed role set $\mathcal{R}$, i.e. $\mathscr{C} \subseteq \mathcal{R}$. The transition system is parameterised by reliability assumptions, in the form of a fixed set of reliable roles $\mathscr{R}$. Where it is not ambiguous, we write G as an abbreviation of $\langle \emptyset; G \rangle$. We define the reduction rules of global types in Definition 4.12.

Definition 4.12 (Global Type Reductions)**.** The global type (annotated with a set of crashed roles $\mathscr{C}$) transition relation $\xrightarrow{\alpha}_{\mathscr{R}}$ is inductively defined by the rules in Fig. 7, parameterised by a fixed set $\mathscr{R}$ of reliable roles. We write $\langle \mathscr{C}; G \rangle \to_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle$ if there exists α such that $\langle \mathscr{C}; G \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle$; we write $\langle \mathscr{C}; G \rangle \to_{\mathscr{R}}$ if there exists $\mathscr{C}'$, G', and α such that $\langle \mathscr{C}; G \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle$, and $\to^*_{\mathscr{R}}$ for the transitive and reflexive closure of $\to_{\mathscr{R}}$.

Rules [GR-$\oplus$] and [GR-&] model sending and receiving messages respectively, as are standard in existing works [DY13]. We add a (highlighted) extra condition that the message exchanged not be a 'pseudo'-message carrying the crash label. [GR-μ] is a standard rule that deals with recursion.

We introduce (highlighted) rules to account for crash and consequential behaviour.

- Rule [GR-↯] models crashes, where a live ($\mathtt{p} \in \mathsf{roles}(G)$), but unreliable ($\mathtt{p} \notin \mathscr{R}$) role $\mathtt{p}$ may crash. The crashed role $\mathtt{p}$ is added into the set of crashed roles ($\mathscr{C} \cup \{\mathtt{p}\}$), and removed from the global type, resulting in a global type $G\!\upharpoonright\!\lightning\,\mathtt{p}$.
- Rule [GR-⊙] is for *crash detection*, where a live role $\mathtt{q}$ may detect that $\mathtt{p}$ has crashed at reception, and then continues with the crash handling continuation labelled crash. This rule only applies when the message en route is a 'pseudo'-message, since otherwise a message rests in the queue of the receiver and can be received despite the crash of the sender (cf. [GR-&]).
- Rule [GR-↯m] models the orphaning of a message sent from a live role $\mathtt{p}$ to a crashed role $\mathtt{q}$. Similar to the requirement in [GR-⊕], we add the side condition that the message sent is not a 'pseudo'-message.

Finally, rules [GR-Ctx-I] and [GR-Ctx-II] allow non-interfering reductions of (intermediate) global types under prefix, provided that all of the continuations can be reduced by that label.

Remark 4.13 (Necessity of $\mathscr{C}$ in Semantics)**.** While we can obtain the set of crashed roles in any global type G via $\mathsf{roles}^{\lightning}(G)$, we need a separate $\mathscr{C}$ for bookkeeping purposes.

Let $G = \mathtt{p}{\to}\mathtt{q}{:}\{\mathtt{m}.\mathsf{end},\quad \mathsf{crash}.\mathsf{end}\}$, we can have the following reductions:

$$\langle \emptyset; G\rangle \xrightarrow{\mathtt{q}\lightning}_{\emptyset} \langle \{\mathtt{q}\}; \mathtt{p}{\to}\mathtt{q}^{\lightning}{:}\{\mathtt{m}.\mathsf{end},\quad \mathsf{crash}.\mathsf{end}\}\rangle \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m}}_{\emptyset} \langle \{\mathtt{q}\}; \mathsf{end}\rangle$$

While we can deduce $\mathtt{q}$ is a crashed role in the interim global type, the same information cannot be recovered from the final global type end.

Both live and crashed roles in a global type remain consistent throughout a transition, except for those directly involved in the crash transition action, as demonstrated in Lemma 4.14.

Lemma 4.14 (No Revival Or Unexpected Crashes)**.** *Assume* $\langle \mathscr{C}; G\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$.

(1) *If* $\mathtt{p} \in \mathsf{roles}^{\lightning}(G')$ *and* $\alpha \neq \mathtt{p}\!\upharpoonright\!\lightning$, *then* $\mathtt{p} \in \mathsf{roles}^{\lightning}(G)$;
(2) *If* $\mathtt{p} \in \mathsf{roles}(G')$ *and* $\alpha \neq \mathtt{p}\!\upharpoonright\!\lightning$, *then* $\mathtt{p} \in \mathsf{roles}(G)$;
(3) *If* $\mathtt{p} \in \mathsf{roles}^{\lightning}(G')$ *and* $\alpha = \mathtt{p}\!\upharpoonright\!\lightning$, *then* $\mathtt{p} \in \mathsf{roles}(G)$.

Proof. By induction on global type reductions. See Appendix A.3 for details. □

We introduce an auxiliary concept of *well-annotated* global types in Definition 4.15, as a consistency requirement for crash annotations $\lightning$ in a global type G, and the set of crashed roles $\mathscr{C}$, and a fixed set of reliable roles $\mathscr{R}$. We show that well-annotatedness w.r.t. $\mathscr{R}$ is preserved by global type reductions in Lemma 4.16. It follows that, a global type G without runtime constructs is trivially well-annotated, and all reducta $G \to^*_{\mathscr{R}} \langle \mathscr{C}; G'\rangle$ are also well-annotated.

Definition 4.15 (Well-Annotated Global Types)**.** A global type G with crashed roles $\mathscr{C}$ is *well-annotated* w.r.t. a (fixed) set of reliable roles $\mathscr{R}$, iff:

(WA1) No reliable roles are crashed, $\mathsf{roles}^{\lightning}(G) \cap \mathscr{R} = \emptyset$; and,
(WA2) All roles with crash annotations are in the crashed set, $\mathsf{roles}^{\lightning}(G) \subseteq \mathscr{C}$; and,
(WA3) A role cannot be live and crashed simultaneously, $\mathsf{roles}(G) \cap \mathsf{roles}^{\lightning}(G) = \emptyset$.

Lemma 4.16 (Preservation of Well-Annotated Global Types)**.** *If* $\langle \mathscr{C}; G\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, *and* $\langle \mathscr{C}; G\rangle$ *is well-annotated w.r.t.* $\mathscr{R}$, *then* $\langle \mathscr{C}'; G'\rangle$ *is also well-annotated w.r.t.* $\mathscr{R}$.

Proof. By induction on global type reductions (Definition 4.12). See Appendix A.3 for details. □

4.3. **Crash-Stop Semantics of Configurations.** After giving semantics to global types, we now give an LTS semantics to *configurations*, i.e. a collection of local types and communication queues across roles. We first give a definition of configurations in Definition 4.17, followed by their reduction rules in Definition 4.18.

Definition 4.17 (Configurations)**.** A configuration is a tuple $\Gamma;\Delta$, where Γ is a *typing context*, denoting a partial mapping from roles to local types, defined as:

$$\Gamma \ ::= \ \emptyset \ \mid \ \Gamma, \mathtt{p} \triangleright T$$

The *context composition* Γ_1, Γ_2 is defined iff $\mathrm{dom}(\Gamma_1) \cap \mathrm{dom}(\Gamma_2) = \emptyset$. A typing context Γ can be *decomposed* (or *split*) into sub-contexts Γ_1 and Γ_2, written $\Gamma = \Gamma_1, \Gamma_2$, if $\mathrm{dom}(\Gamma) = \mathrm{dom}(\Gamma_1) \cup \mathrm{dom}(\Gamma_2)$, and $\forall \mathtt{p} \in \mathrm{dom}(\Gamma)$, $\Gamma(\mathtt{p}) = (\Gamma_1, \Gamma_2)(\mathtt{p})$. We write $\Gamma[\mathtt{p} \mapsto T]$ for typing context updates, namely $\Gamma[\mathtt{p} \mapsto T](\mathtt{p}) = T$ and $\Gamma[\mathtt{p} \mapsto T](\mathtt{q}) = \Gamma(\mathtt{q})$ (where $\mathtt{p} \neq \mathtt{q}$).

A *queue*, denoted τ, is either a (possibly empty) sequence of messages $M_1 \cdot M_2 \cdot \cdots \cdot M_n$, or an unavailable queue $\oslash$. We write ϵ for an empty queue, and $M \cdot \tau'$ for a non-empty queue with message M at the beginning. A *queue message* M is of form $\mathtt{m}(B)$, denoting a message with label $\mathtt{m}$ and payload B. We sometimes omit B when the payload is not of specific interest.

We write Δ to denote a *queue environment*, a collection of peer-to-peer queues. A queue from $\mathtt{p}$ to $\mathtt{q}$ at environment Δ is denoted $\Delta(\mathtt{p}, \mathtt{q})$. We define queue environment updates $\Delta[\mathtt{p}, \mathtt{q} \mapsto \tau]$ similarly.

We also write $\tau' \cdot M$ for appending a message at the end of a queue: the message is appended to the sequence when τ' is available, or discarded when τ' is unavailable (i.e. $\oslash \cdot M = \oslash$). Additionally, we write $\Delta[\cdot, \mathtt{q} \mapsto \oslash]$ for making all the queues to $\mathtt{q}$ unavailable: i.e. $\Delta[\mathtt{p}_1, \mathtt{q} \mapsto \oslash][\mathtt{p}_2, \mathtt{q} \mapsto \oslash] \cdots [\mathtt{p}_n, \mathtt{q} \mapsto \oslash]$.

We write Δ_ϵ to denote an *empty* queue environment, where $\Delta_\epsilon(\mathtt{p}, \mathtt{q}) = \epsilon$ for any $\mathtt{p}$ and $\mathtt{q}$ in the domain.

We give an LTS semantics of configurations in Definition 4.18. Similar to that of global types, we model the semantics of configurations in an asynchronous (a.k.a. message-passing) fashion, using a queue environment to represent the communication queues among all roles.

Definition 4.18 (Configuration Semantics)**.** The *configuration transition relation* $\xrightarrow{\alpha}$ is defined in Fig. 8. We write $\Gamma;\Delta \xrightarrow{\alpha}$ iff $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$ for some Γ' and Δ'. We define two *reductions* $\rightarrow$ and $\rightarrow_{\mathscr{R}}$ (where $\mathscr{R}$ is a fixed set of reliable roles) as follows.

- We write $\Gamma;\Delta \rightarrow \Gamma';\Delta'$ for $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$ with $\alpha \in \{\mathtt{p}\&\mathtt{q} : \mathtt{m}(B), \mathtt{p}\oplus\mathtt{q} : \mathtt{m}(B), \mathtt{p}\odot\mathtt{q}\}$. We write $\Gamma;\Delta \rightarrow$ iff $\Gamma;\Delta \rightarrow \Gamma';\Delta'$ for some $\Gamma';\Delta'$, and $\Gamma;\Delta \not\rightarrow$ for its negation, and $\rightarrow^*$ for the reflexive and transitive closure of $\rightarrow$;
- We write $\Gamma;\Delta \rightarrow_{\mathscr{R}} \Gamma';\Delta'$ for $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$ with $\alpha \notin \{\mathtt{r}\lightning \mid \mathtt{r} \in \mathscr{R}\}$. We write $\Gamma;\Delta \rightarrow_{\mathscr{R}}$ iff $\Gamma;\Delta \rightarrow_{\mathscr{R}} \Gamma';\Delta'$ for some $\Gamma';\Delta'$, and $\Gamma;\Delta' \not\rightarrow_{\mathscr{R}}$ for its negation. We define $\rightarrow^*_{\mathscr{R}}$ as the reflexive and transitive closure of $\rightarrow_{\mathscr{R}}$.

We first explain the standard rules: rule [Γ-⊕] (resp. [Γ-&]) says that a role can perform an output (resp. input) transition by appending (resp. consuming) a message at the corresponding queue. Recall that whenever a queue is unavailable, the resulting queue remains unavailable after appending ($\oslash \cdot M = \oslash$). Therefore, the rule [Γ-⊕] covers delivery to both

$$\frac{\Gamma(\mathtt{p}) = \mathtt{q}\oplus\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I} \quad k \in I}{\Gamma;\Delta \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m}_\mathtt{k}(B_k)} \Gamma[\mathtt{p}\mapsto T_k];\Delta[\mathtt{p},\mathtt{q}\mapsto \Delta(\mathtt{p},\mathtt{q})\cdot\mathtt{m}_k(B_k)]}\ [\Gamma\text{-}\oplus]$$

$$\frac{\Gamma(\mathtt{p}) = \mathtt{q}\&\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I} \quad k \in I \quad \Delta(\mathtt{q},\mathtt{p}) = \mathtt{m}_k(B_k)\cdot\tau' \neq \oslash}{\Gamma;\Delta \xrightarrow{\mathtt{p}\&\mathtt{q}:\mathtt{m}_\mathtt{k}(B_k)} \Gamma[\mathtt{p}\mapsto T_k];\Delta[\mathtt{q},\mathtt{p}\mapsto \tau']}\ [\Gamma\text{-}\&]$$

$$\frac{\Gamma(\mathtt{p}) = \mu\mathbf{t}.T \quad \Gamma[\mathtt{p}\mapsto T\{\mu\mathbf{t}.T/\mathbf{t}\}];\Delta \xrightarrow{\alpha} \Gamma';\Delta'}{\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'}\ [\Gamma\text{-}\mu] \qquad \frac{\Gamma(\mathtt{p}) \neq \mathsf{end} \quad \Gamma(\mathtt{p}) \neq \mathsf{stop}}{\Gamma;\Delta \xrightarrow{\mathtt{p}\lightning} \Gamma[\mathtt{p}\mapsto \mathsf{stop}];\Delta[\cdot,\mathtt{p}\mapsto\oslash]}\ [\Gamma\text{-}\lightning]$$

$$\frac{\Gamma(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I} \quad \Gamma(\mathtt{p}) = \mathsf{stop} \quad k \in I \quad \mathtt{m}_k = \mathsf{crash} \quad \Delta(\mathtt{p},\mathtt{q}) = \epsilon}{\Gamma;\Delta \xrightarrow{\mathtt{q}\odot\mathtt{p}} \Gamma[\mathtt{q}\mapsto T_k];\Delta}\ [\Gamma\text{-}\odot]$$

FIGURE 8. Configuration semantics.

crashed and live roles, whereas two separate rules are used in modelling global type semantics ([GR-$\oplus$] and [GR-$\lightning\mathtt{m}$]). We also include a standard rule [Γ-μ] for recursive types.

The key innovations are the (highlighted) rules modelling crashes and crash detection: by rule [Γ-$\lightning$], a role p may crash and become stop at any time (unless it is already ended or stopped). All of p's receiving queues become unavailable $\oslash$, so that future messages to p would be discarded. Note that any pending messages in the receiving queue are discarded (since the queue becomes unavailable $\oslash$), since the messages cannot be processed after the crash.

Rule [Γ-$\odot$] models crash detection and handling: if p is crashed and stopped, another role q attempting to receive from p can then take its crash handling branch. However, this rule only applies when the corresponding queue is empty: it is still possible to receive messages sent before crashing via [Γ-&].

4.4. **Alternative Modellings for Crash-Stop Failures.** Before we dive into the relation between two semantics, let us have a short digression to discuss our modelling choices and alternatives. In this work, we mostly follow the assumptions laid out in [BSYZ22], where a crash is detected at reception. However, they opt to use a synchronous (rendez-vous) semantics, whereas we give an asynchronous (message-passing) semantics, which entails interesting scenarios that would not arise in a synchronous semantics.

Specifically, consider the case where a role p sends a message to q, and then p crashes after sending, but before q receives the message. The situation does not arise under a synchronous semantics, since sending and receiving actions are combined into a single transmission action.

Intuitively, there are two possibilities to handle this scenario. The questions are whether the message sent immediately before crashing is deliverable to q, and consequentially, at what time q detects the crash of p.

In our semantics (Figs. 7 and 8), we opt to answer the first question positively: we argue that this model is more consistent with our 'passive' crash detection design. For example, if a role p never receives from another role q, then p does not need to react in the event of q's crash. Following a similar line of reasoning, if the message sent by p arrives in the receiving queue of q, then q should be able to receive the message, without triggering a crash

detection (although it may be triggered later). As a consequence, we require in [Γ-⊙] that the queue $\Delta(\mathtt{p},\mathtt{q})$ be empty, to reflect the idea that crash detection should be a 'last resort'.

For an alternative model, we can opt to detect the crash after it has occurred. This is possibly better modelled with using outgoing queues (cf. [DY15]), instead of incoming queues in the semantics presented. Practically, this may be the scenario that a TCP connection is closed (or reset) when a peer has crashed, and the content in the queue is lost. It is worth noting that this kind of alternative model will not affect our main theoretical results: the operational correspondence between global and local type semantics, and furthermore, global type properties guaranteed by projection.

4.5. **Relating Global Type and Configuration Semantics.** We have given LTS semantics for both global types (Definition 4.12) and configurations (Definition 4.18). We will now relate these two semantics with the help of the projection operator $\upharpoonright$ (Definition 4.3) and the subtyping relation $\leqslant$ (Definition 4.4).

We associate configurations $\Gamma;\Delta$ with global types G (as annotated with a set of crashed roles $\mathscr{C}$) by projection, written $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$. Naturally, there are two components of the association: *(1)* the local types in Γ need to correspond to the projections of the global type G and the set of crashed roles $\mathscr{C}$; and *(2)* the queues in Δ corresponds to the transmissions en route in the global type G and also the set of crashed roles $\mathscr{C}$.

Definition 4.19 (Association of Global Types and Configurations)**.** A configuration $\Gamma;\Delta$ is associated to a (well-annotated w.r.t. $\mathscr{R}$) global type $\langle \mathscr{C}; G\rangle$, written $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$, iff

(1) Γ can be split into disjoint (possibly empty) sub-contexts $\Gamma = \Gamma_G, \Gamma_{\lightning}, \Gamma_{\mathsf{end}}$ where:
  - (A1) Γ_G contains projections of G: $\mathrm{dom}(\Gamma_G) = \mathrm{roles}(G)$, and $\forall \mathtt{p} \in \mathrm{dom}(\Gamma_G) : \Gamma(\mathtt{p}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{p}$;
  - (A2) $\Gamma_{\lightning}$ contains crashed roles: $\mathrm{dom}(\Gamma_{\lightning}) = \mathscr{C}$, and $\forall \mathtt{p} \in \mathrm{dom}(\Gamma_{\lightning}) : \Gamma(\mathtt{p}) = \mathsf{stop}$;
  - (A3) Γ_{end} contains only end endpoints: $\forall \mathtt{p} \in \Gamma_{\mathsf{end}} : \Gamma(\mathtt{p}) = \mathsf{end}$.

(2) (A4) Δ is associated with global type $\langle \mathscr{C}; G\rangle$, given as follows:
  - (i) Receiving queues for a role is unavailable if and only if it has crashed: $\forall \mathtt{q} : \mathtt{q} \in \mathscr{C} \iff \Delta(\cdot, \mathtt{q}) = \oslash$;
  - (ii) If $G = \mathsf{end}$ or $G = \mu \mathbf{t}.G'$, then queues between all roles are empty (expect receiving queue for crashed roles): $\forall \mathtt{p}, \mathtt{q} : \mathtt{q} \notin \mathscr{C} \implies \Delta(\mathtt{p},\mathtt{q}) = \epsilon$;
  - (iii) If $G = \mathtt{p} \to \mathtt{q}^{\dagger} : \{\mathtt{m}_{\mathtt{i}}(B_i).G'_i\}_{i\in I}$, or $G = \mathtt{p}^{\dagger} \rightsquigarrow \mathtt{q} : j \, \{\mathtt{m}_{\mathtt{i}}(B_i).G'_i\}_{i\in I}$ with $\mathtt{m}_j = \mathsf{crash}$ (i.e. a 'pseudo'-message is en route), then *(i)* if $\mathtt{q}$ is live, then the queue from $\mathtt{p}$ to $\mathtt{q}$ is empty: $\mathtt{q}^{\dagger} \neq \mathtt{q}^{\lightning} \implies \Delta(\mathtt{p},\mathtt{q}) = \epsilon$, and *(ii)* $\forall i \in I : \Delta$ is associated with $\langle \mathscr{C}; G'_i\rangle$; and,
  - (iv) If $G = \mathtt{p}^{\dagger} \rightsquigarrow \mathtt{q} : j \, \{\mathtt{m}_{\mathtt{i}}(B_i).G'_i\}_{i\in I}$ with $\mathtt{m}_j \neq \mathsf{crash}$, then *(i)* the queue from $\mathtt{p}$ to $\mathtt{q}$ begins with the message $\mathtt{m}_j(B_j)$: $\Delta(\mathtt{p},\mathtt{q}) = \mathtt{m}_j(B_j) \cdot \tau$; *(ii)* $\forall i \in I :$ removing the message from the head of the queue, $\Delta[\mathtt{p},\mathtt{q} \mapsto \tau]$ is associated with $\langle \mathscr{C}; G'_i\rangle$.

We write $\Gamma \sqsubseteq_{\mathscr{R}} G$ as an abbreviation of $\Gamma;\Delta_{\epsilon} \sqsubseteq_{\mathscr{R}} \langle \emptyset; G\rangle$. We sometimes say a Γ (resp. Δ) is associated with $\langle \mathscr{C}; G\rangle$ for stating Item 1 (resp. Item 2) is satisfied.

We demonstrate the relation between the two semantics via association, by showing two main theorems: all possible reductions of a configuration have a corresponding action in reductions of the associated global type (Theorem 4.20); and the reducibility of a global type is the same as its associated configuration (Theorem 4.21).

Theorem 4.20 (Completeness of Association)**.** *Given associated global type G and configuration $\Gamma;\Delta$: $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$. If $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$, where $\alpha \neq \mathtt{p}\lightning$ for all $\mathtt{p} \in \mathscr{R}$, then there exists $\langle \mathscr{C}'; G'\rangle$ such that $\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$ and $\langle \mathscr{C}; G\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$.*

Proof. By induction on configuration reductions (Definition 4.18). See Appendix A.5 for details. □

Theorem 4.21 (Soundness of Association)**.** *Given associated global type G and configuration $\Gamma;\Delta$: $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$. If $\langle \mathscr{C}; G\rangle \rightarrow_{\mathscr{R}}$, then there exists $\Gamma';\Delta'$, α and $\langle \mathscr{C}'; G'\rangle$, such that $\langle \mathscr{C}; G\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, $\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, and $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$.*

Proof. By induction on global type reductions (Definition 4.12). See Appendix A.5 for details. □

By Theorems 4.20 and 4.21, we obtain, as a corollary, that a global type G is in operational correspondence with the typing context $\Gamma = \{\mathtt{p} \triangleright G \upharpoonright_{\mathscr{R}} \mathtt{p}\}_{\mathtt{p}\in \mathrm{roles}(G)}$, which contains the projections of all roles in G.

Remark 4.22 (Sufficiency of Soundness Theorem)**.** Curious readers may wonder why we proved a soundness theorem that is not the dual of the completeness theorem, e.g. as seen in the literature [DY13]. This is a consequence of using 'full' subtyping (Definition 4.4, notably [Sub-⊕]). A local type in the typing context may have fewer branches to choose from than the projected local type, resulting in uninhabited sending actions in the global type.

For example, let $G = \mathtt{p}\to\mathtt{q}\colon\{\mathtt{m}_1.\mathtt{end};\ \mathtt{m}_2.\mathtt{end}\}$. An associated typing context Γ (assuming $\mathtt{p}$ reliable) may have $\Gamma(\mathtt{p}) = \mathtt{q}\oplus\{\mathtt{m}_1.\mathtt{end}\} \leqslant \mathtt{q}\oplus\{\mathtt{m}_1.\mathtt{end};\ \mathtt{m}_2.\mathtt{end}\}$ (via [Sub-⊕]). The global type G may make a transition $\mathtt{p}\oplus\mathtt{q} : \mathtt{m}_2$, where an associated configuration $\Gamma;\Delta_\epsilon$ cannot.

Our soundness theorem is nevertheless *sufficient* for concluding that desired properties are guaranteed via association, e.g. safety, deadlock-freedom, and liveness, as illustrated in Section 4.6.

Remark 4.23 (Relation between Well-Annotated and Well-Formed Global Types)**.** In the multiparty session type literature, a global type is *well-formed* if it can be projected onto every declared protocol participant. Readers may wonder how well-formedness is applied to the global type in this paper and how well-annotated global types (Definition 4.15) relate to well-formed ones. Our definition of association (Definition 4.19) ensures that a global type associated with a configuration, with respect to a set of reliable roles $\mathscr{R}$, is also well-formed with respect to $\mathscr{R}$. This is because: (1) every global type is closed, i.e. it has no free type variables; (2) every global type is contractive; and (3) condition (A1) in Definition 4.19 ensures that the associated global type is projectable onto all roles with respect to a set of reliable roles, which is a key requirement for well-formedness.

Since the soundness (Theorem 4.21) and completeness (Theorem 4.20) of association, along with the results on typed session properties (discussed in Section 5), depend on the concept of association, it follows that all involved global types are well-formed.

Furthermore, there is no direct relationship between well-formedness and well-annotation; a well-annotated global type may not be well-formed, and vice versa. All main results apply to global types that are both well-formed and well-annotated.

4.6. **Properties Guaranteed by Projection.** A key benefit of our top-down approach of multiparty protocol design is that desirable properties are guaranteed by the methodology. As a consequence, processes following the local types obtained from projections are correct *by construction*. In this subsection, we focus on three properties: *communication safety*, *deadlock-freedom*, and *liveness*, and show that the three properties are guaranteed from configurations associated with global types.

Communication Safety. We begin by defining communication safety for configurations (Definition 4.24). We focus on the following two safety requirements: (i) each role must be able to handle any message that may end up in their receiving queue (so that there are no label mismatches); and (ii) each receiver must be able to handle the potential crash of the sender, unless the sender is reliable.

Definition 4.24 (Configuration Safety)**.** Given a fixed set of reliable roles $\mathscr{R}$, we say that φ is an $\mathscr{R}$*-safety property* of configurations iff, whenever $\varphi(\Gamma;\Delta)$, we have:

[S-⊕&] $\Gamma(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m}_i(B_i).S'_i\}_{i\in I}$ and $\Delta(\mathtt{p},\mathtt{q}) \neq \oslash$ and $\Delta(\mathtt{p},\mathtt{q}) \neq \epsilon$ implies $\Gamma;\Delta \xrightarrow{\mathtt{q}\&\mathtt{p}:\mathtt{m}'(B')}$;

[S-↯&] $\Gamma(\mathtt{p}) = \mathtt{stop}$ and $\Gamma(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m}_i(S_i).S'_i\}_{i\in I}$ and $\Delta(\mathtt{p},\mathtt{q}) = \epsilon$ implies $\Gamma;\Delta \xrightarrow{\mathtt{q}\odot\mathtt{p}}$;

[S-μ] $\Gamma(\mathtt{p}) = \mu\mathbf{t}.S$ implies $\varphi(\Gamma[\mathtt{p}\mapsto S\{\mu\mathbf{t}.S/\mathbf{t}\}];\Delta)$;

[S-→↯] $\Gamma;\Delta \to_{\mathscr{R}} \Gamma';\Delta'$ implies $\varphi(\Gamma';\Delta')$.

We say $\Gamma;\Delta$ *is* $\mathscr{R}$*-safe*, if $\varphi(\Gamma;\Delta)$ holds for some $\mathscr{R}$-safety property φ.

We use a coinductive view of the safety property [San11], where the predicate of $\mathscr{R}$-safe configurations is the largest $\mathscr{R}$-safety property, by taking the union of all safety properties φ. For a configuration $\Gamma;\Delta$ to be $\mathscr{R}$-safe, it has to satisfy all clauses defined in Definition 4.24.

By clause [S-⊕&], whenever a role $\mathtt{q}$ receives from another role $\mathtt{p}$, and a message is present in the queue, the receiving action must be possible for some label $\mathtt{m}'$, i.e. the receiver $\mathtt{q}$ must support all output messages that may appear at the head of the queue sent from $\mathtt{p}$.

Clause [S-↯&] states that if a role $\mathtt{q}$ receives from a crashed role $\mathtt{p}$, and there is nothing in the queue, then $\mathtt{q}$ must have a $\mathtt{crash}$ branch, and a crash detection action can be fired. (Note that [S-⊕&] applies when the queue is non-empty, despite the crash of sender $\mathtt{p}$.)

Finally, clause [S-μ] extends the previous clauses by unfolding any recursive entries; and clause [S-→↯] states that any configuration $\Gamma';\Delta'$ which $\Gamma;\Delta$ transitions to must also be $\mathscr{R}$-safe. By using transition $\to_{\mathscr{R}}$, we ignore crash transitions $\mathtt{p}\lightning$ for any reliable role $\mathtt{p}\in\mathscr{R}$.

Example 4.25. Recall the local types of the Simpler Logging example in Section 2:

$$T_{\mathtt{C}} = \mathtt{I}\oplus\mathtt{read}.T'_{\mathtt{C}} \qquad T_{\mathtt{L}} = \mathtt{I}\oplus\mathtt{trigger}.T'_{\mathtt{L}}$$
$$T'_{\mathtt{C}} = \mathtt{I}\&\mathtt{report}(\mathrm{log}).\mathtt{end} \qquad T'_{\mathtt{L}} = \mathtt{I}\&\left\{\begin{array}{l}\mathtt{fatal}.\mathtt{end}\\ \mathtt{read}.\mathtt{I}\oplus\mathtt{report}(\mathrm{log}).\mathtt{end}\end{array}\right\}$$
$$T_{\mathtt{I}} = \mathtt{L}\&\mathtt{trigger}.T'_{\mathtt{I}}$$
$$T'_{\mathtt{I}} = \mathtt{C}\&\left\{\begin{array}{l}\mathtt{read}.\mathtt{L}\oplus\mathtt{read}.\mathtt{L}\&\mathtt{report}(\mathrm{log}).\mathtt{C}\oplus\mathtt{report}(\mathrm{log}).\mathtt{end}\\ \mathtt{crash}.T''_{\mathtt{I}}\end{array}\right\}$$
$$T''_{\mathtt{I}} = \mathtt{L}\oplus\mathtt{fatal}.\mathtt{end}$$

The configuration $\Gamma;\Delta$, where $\Gamma = \mathtt{C}\triangleright T_{\mathtt{C}}, \mathtt{L}\triangleright T_{\mathtt{L}}, \mathtt{I}\triangleright T_{\mathtt{I}}$ and $\Delta = \Delta_\epsilon$, is $\{\mathtt{L},\mathtt{I}\}$-safe. This can be verified by checking its possible reductions. For example, in the case where $\mathtt{C}$ crashes immediately, we have:

$$
\begin{aligned}
\Gamma;\Delta &\xrightarrow{\mathtt{C}\not\downarrow} \Gamma[\mathtt{C}\mapsto \mathsf{stop}];\Delta[\cdot,\mathtt{C}\mapsto \oslash]\\
&\xrightarrow{\mathtt{L}\oplus\mathtt{I}:\mathtt{trigger}} \Gamma[\mathtt{C}\mapsto \mathsf{stop}][\mathtt{L}\mapsto T'_{\mathtt{L}}];\Delta[\cdot,\mathtt{C}\mapsto \oslash][\mathtt{L},\mathtt{I}\mapsto \mathtt{trigger}]\\
&\xrightarrow{\mathtt{I}\&\mathtt{L}:\mathtt{trigger}} \Gamma[\mathtt{C}\mapsto \mathsf{stop}][\mathtt{L}\mapsto T'_{\mathtt{L}}][\mathtt{I}\mapsto T'_{\mathtt{I}}];\Delta[\cdot,\mathtt{C}\mapsto \oslash]\\
&\xrightarrow{\mathtt{I}\odot\mathtt{C}} \Gamma[\mathtt{C}\mapsto \mathsf{stop}][\mathtt{L}\mapsto T'_{\mathtt{L}}][\mathtt{I}\mapsto T''_{\mathtt{I}}];\Delta[\cdot,\mathtt{C}\mapsto \oslash]\\
&\xrightarrow{\mathtt{I}\oplus\mathtt{L}:\mathtt{fatal}} \Gamma[\mathtt{C}\mapsto \mathsf{stop}][\mathtt{L}\mapsto T'_{\mathtt{L}}][\mathtt{I}\mapsto \mathsf{end}];\Delta[\cdot,\mathtt{C}\mapsto \oslash][\mathtt{I},\mathtt{L}\mapsto \mathtt{fatal}]\\
&\xrightarrow{\mathtt{L}\&\mathtt{I}:\mathtt{fatal}} \Gamma[\mathtt{C}\mapsto \mathsf{stop}][\mathtt{L}\mapsto \mathsf{end}][\mathtt{I}\mapsto \mathsf{end}];\Delta[\cdot,\mathtt{C}\mapsto \oslash]
\end{aligned}
$$

and each reductum satisfies all clauses of Definition 4.24. The cases where C crashes after sending the reading message to I are similar. There are no other crash reductions to consider, since both L and I are assumed to be reliable. The cases where no crashes occur are similar as well, except that [Γ-⊙] and [Γ-↯] are not applied in the non-crash reductions.

Deadlock-Freedom. The property of deadlock-freedom, sometimes also known as progress, describes whether a configuration can keep reducing unless it is a terminal configuration. We give its formal definition in Definition 4.26.

Definition 4.26 (Configuration Deadlock-Freedom)**.** Given a set of reliable roles $\mathscr{R}$, we say that a configuration $\Gamma;\Delta$ is *$\mathscr{R}$-deadlock-free* iff:

(1) $\Gamma;\Delta$ is $\mathscr{R}$-safe; and,

(2) If $\Gamma;\Delta$ can reduce to a configuration $\Gamma';\Delta'$ without further reductions: $\Gamma;\Delta \to^*_{\mathscr{R}} \Gamma';\Delta' \not\to_{\mathscr{R}}$, then:

   (a) Γ' can be split into two disjoint contexts, one with only end entries, and one with only stop entries: $\Gamma' = \Gamma'_{\mathsf{end}}, \Gamma'_{\lightning}$, where $\mathrm{dom}(\Gamma'_{\mathsf{end}}) = \{\mathtt{p} \mid \Gamma'(\mathtt{p}) = \mathsf{end}\}$ and $\mathrm{dom}\left(\Gamma'_{\lightning}\right) = \{\mathtt{p} \mid \Gamma'(\mathtt{p}) = \mathsf{stop}\}$; and,

   (b) Δ' is empty for all pairs of roles, except for the receiving queues of crashed roles, which are unavailable: $\forall \mathtt{p},\mathtt{q} : \Delta'(\cdot,\mathtt{q}) = \oslash$ if $\Gamma'(\mathtt{q}) = \mathsf{stop}$, and $\Delta'(\mathtt{p},\mathtt{q}) = \epsilon$, otherwise.

It is worth noting that a (safe) configuration that reduces infinitely satisfies deadlock-freedom, as Item 2 in the premise does not hold. Otherwise, whenever a terminal configuration is reached, it must satisfy Item 2a that all local types in the typing context be terminated (either successfully end, or crashed stop), and Item 2b that all queues be empty (unless unavailable due to crash). As a consequence, a deadlock-free configuration $\Gamma;\Delta$ either does not stop reducing, or terminates in a stable configuration.

Liveness. The property of liveness describes that every pending internal/external choice is eventually triggered by means of a message transmission or crash detection. Our liveness property is based on *fairness*, which guarantees that every enabled message transmission, including crash detection, is performed successfully. We give the definitions of non-crashing, fair, and live paths of configurations respectively in Definition 4.27, and use these paths to formalise the liveness for configurations in Definition 4.28.

Definition 4.27 (Non-crashing, Fair, Live Paths)**.** A *non-crashing path* is a possibly infinite sequence of configurations $(\Gamma_n;\Delta_n)_{n\in N}$, where $N = \{0,1,2,\ldots\}$ is a set of consecutive natural numbers, and $\forall n \in N$, $\Gamma_n;\Delta_n \to \Gamma_{n+1};\Delta_{n+1}$.

We say that a non-crashing path $(\Gamma_n;\Delta_n)_{n\in N}$ is *fair* iff, $\forall n \in N$:

(F1) $\Gamma_n;\Delta_n \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m}(B)}$ implies $\exists k,\mathtt{m}',B'$ such that $n \le k \in N$ and $\Gamma_k;\Delta_k \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m}'(B')} \Gamma_{k+1};\Delta_{k+1}$;

(F2) $\Gamma_n;\Delta_n \xrightarrow{\mathtt{p\&q:m}(B)}$ implies $\exists k$ such that $n \le k \in N$ and $\Gamma_k;\Delta_k \xrightarrow{\mathtt{p\&q:m}(B)} \Gamma_{k+1};\Delta_{k+1}$;
(F3) $\Gamma_n;\Delta_n \xrightarrow{\mathtt{p}\odot\mathtt{q}}$ implies $\exists k$ such that $n \le k \in N$ and $\Gamma_k;\Delta_k \xrightarrow{\mathtt{p}\odot\mathtt{q}} \Gamma_{k+1};\Delta_{k+1}$.

We say that a non-crashing path $(\Gamma_n;\Delta_n)_{n\in N}$ is *live* iff, $\forall n \in N$:
(L1) $\Delta_n(\mathtt{p},\mathtt{q}) = \mathtt{m}(B)\cdot\tau \neq \oslash$ and $\mathtt{m} \neq \mathsf{crash}$ implies $\exists k$ such that $n \le k \in N$ and $\Gamma_k;\Delta_k \xrightarrow{\mathtt{q\&p:m}(B)} \Gamma_{k+1};\Delta_{k+1}$;
(L2) $\Gamma_n(\mathtt{p}) = \mathtt{q}\&\{\mathtt{m_i}(B_i).T_i\}_{i\in I}$ implies $\exists k, \mathtt{m}', B'$ such that $n \le k \in N$ and $\Gamma_k;\Delta_k \xrightarrow{\mathtt{p\&q:m}'(B')} \Gamma_{k+1};\Delta_{k+1}$ or $\Gamma_k;\Delta_k \xrightarrow{\mathtt{p}\odot\mathtt{q}} \Gamma_{k+1};\Delta_{k+1}$.

A non-crashing path is a (possibly infinite) sequence of reductions of a configuration without crashes. A non-crashing path is fair if along the path, every internal choice eventually sends a message (F1), every external choice eventually receives a message (F2), and every crash detection is eventually performed (F3). A non-crashing path is live if along the path, every non-crash message in the queue is eventually consumed (L1), and every hanging external choice eventually consumes a message or performs a crash detection (L2).

Definition 4.28 (Configuration Liveness)**.** Given a set of reliable roles $\mathscr{R}$, we say that a configuration $\Gamma;\Delta$ is $\mathscr{R}$*-live* iff: (1) $\Gamma;\Delta$ is $\mathscr{R}$-safe; and, (2) $\Gamma;\Delta \to^*_{\mathscr{R}} \Gamma';\Delta'$ implies all non-crashing paths starting with $\Gamma';\Delta'$ that are fair are also live.

A configuration $\Gamma;\Delta$ is $\mathscr{R}$-live when it is $\mathscr{R}$-safe and any reductum of $\Gamma;\Delta$ (via transition $\to^*_{\mathscr{R}}$) consistently leads to a live path if it is fair.

Example 4.29. We illustrate safety, deadlock-freedom, and liveness over configurations via a series of small examples. We consider the configuration $\Gamma_A;\Delta_A$, where $\Gamma_A = \Gamma_{A\mathtt{p}}, \Gamma_{A\mathtt{q}}, \Gamma_{A\mathtt{r}}$ and $\Delta_A = \Delta_\epsilon$ with:

$$\begin{array}{rcl}\Gamma_{A\mathtt{p}} & = & \mathtt{p}\triangleright\mu\mathtt{t_p}.\mathtt{q}\oplus\{\mathtt{ok}.\mathtt{q}\&\{\mathtt{ok}.\mathtt{t_p},\ \mathtt{ko}.\mathsf{end},\ \mathsf{crash}.\mathsf{end}\},\ \mathtt{ko}.\mathsf{end}\}\\ \Gamma_{A\mathtt{q}} & = & \mathtt{q}\triangleright\mu\mathtt{t_q}.\mathtt{p}\&\{\mathtt{ok}.\mathtt{p}\oplus\{\mathtt{ok}.\mathtt{t_q},\ \mathtt{ko}.\mathsf{end}\},\ \mathtt{ko}.\mathsf{end}, \mathsf{crash}.\mathtt{r}\oplus\mathtt{ok}.\mathsf{end}\}\\ \Gamma_{A\mathtt{r}} & = & \mathtt{r}\triangleright\mathtt{p}\&\{\mathsf{crash}.\mathtt{q}\&\{\mathtt{ok}.\mathsf{end},\ \mathsf{crash}.\mathsf{end}\}\}\end{array}$$

If we assume that all roles are unreliable, i.e. $\mathscr{R} = \emptyset$, $\Gamma_A;\Delta_A$ is $\emptyset$-safe since the inputs/outputs in the typing context Γ_A are dual and the queue environment Δ_A is empty. However, $\Gamma_A;\Delta_A$ is *neither* $\emptyset$-deadlock-free *nor* $\emptyset$-live since it is possible for $\mathtt{p}$ to crash immediately before $\mathtt{q}$ sends $\mathtt{ko}$ to $\mathtt{p}$. In such cases, $\mathtt{q}$ will *not* detect that $\mathtt{p}$ has crashed (since we only detect crashes on receive actions) and terminate *without* sending a message to the backup process $\mathtt{r}$. This results in a deadlock because $\mathtt{r}$ *will* detect that $\mathtt{p}$ has crashed, and *will* expect a message from $\mathtt{q}$.

We observe that changing the reliability assumptions, without changing the configuration, may influence whether a configuration property holds. For example, in the case of $\Gamma_A;\Delta_A$, we *can* obtain liveness by adjusting the reliability assumptions: in fact, if we assume $\mathtt{r} \in \mathscr{R}$, then $\Gamma_A;\Delta_A$ is both $\mathscr{R}$-deadlock-free and $\mathscr{R}$-live.

Then consider the configuration $\Gamma_B;\Delta_B$, where $\Gamma_B = \Gamma_{B\mathtt{p}}, \Gamma_{B\mathtt{q}}, \Gamma_{B\mathtt{r}}$ and $\Delta_B = \Delta_\epsilon$ with:

$$\begin{array}{rcl}\Gamma_{B\mathtt{p}} & = & \mathtt{p}\triangleright\mu\mathtt{t_p}.\mathtt{q}\oplus\mathtt{ok}.\mathtt{t_p}\\ \Gamma_{B\mathtt{q}} & = & \mathtt{q}\triangleright\mu\mathtt{t_q}.\mathtt{p}\&\{\mathtt{ok}.\mathtt{t_q},\ \mathsf{crash}.\mu\mathtt{t}'_\mathtt{q}.\mathtt{r}\&\{\mathtt{ok}.\mathtt{t}'_\mathtt{q},\ \mathsf{crash}.\mathsf{end}\}\}\\ \Gamma_{B\mathtt{r}} & = & \mathtt{r}\triangleright\mu\mathtt{t_r}.\mathtt{q}\oplus\mathtt{ok}.\mathtt{t_r}\end{array}$$

If we assume that all roles are unreliable, i.e. $\mathscr{R} = \emptyset$, $\Gamma_B;\Delta_B$ is $\emptyset$-safe and $\emptyset$-deadlock-free but *not* $\emptyset$-live – because $\mathtt{p}$ may never crash, and in this case, $\mathtt{r}$'s outputs are never received by $\mathtt{q}$. Notice that, in the case of $\Gamma_B;\Delta_B$, we are unable to make liveness hold purely via combinations of reliable roles: this is because (unless $\mathtt{p}$ crashes) $\mathtt{r}$'s output will never be

received by $\mathtt{q}$, irrespective of reliability assumptions. The configuration itself must instead be adapted accordingly, e.g. in Γ_B, $\mathtt{r}$ should be permitted to send only once it has detected that $\mathtt{p}$ has crashed.

Finally, consider the configuration $\Gamma_C; \Delta_C$, where $\Gamma_C = \Gamma_{C\mathtt{p}}, \Gamma_{C\mathtt{q}}, \Gamma_{C\mathtt{r}}$ and $\Delta_C = \Delta_\epsilon$ with:

$$\begin{array}{rcl} \Gamma_{C\mathtt{p}} & = & \mathtt{p} \triangleright \mathtt{q} \oplus \mathtt{m}_1.\mathtt{q} \& \left\{\mathtt{m}_2.\mathsf{end},\ \mathsf{crash}.\mu \mathbf{t}_\mathtt{p}.\mathtt{r} \oplus \mathtt{ok}.\mathbf{t}_\mathtt{p}\right\} \\ \Gamma_{C\mathtt{q}} & = & \mathtt{q} \triangleright \mathtt{p} \& \left\{\mathtt{m}_1.\mathtt{p} \oplus \mathtt{m}_2.\mathsf{end}\right\} \\ \Gamma_{C\mathtt{r}} & = & \mathtt{r} \triangleright \mathtt{p} \& \left\{\mathsf{crash}.\mu \mathbf{t}_\mathtt{q}.\mathtt{p} \& \left\{\mathtt{ok}.\mathbf{t}_\mathtt{q}\right\}\right\} \end{array}$$

When *all* roles are assumed to be reliable, i.e. $\mathscr{R} = \{\mathtt{p}, \mathtt{q}, \mathtt{r}\}$, $\Gamma_C; \Delta_C$ satisfies $\{\mathtt{p}, \mathtt{q}, \mathtt{r}\}$-safety and $\{\mathtt{p}, \mathtt{q}, \mathtt{r}\}$-deadlock-freedom. However, should we instead assume that no roles are reliable, i.e. $\mathscr{R} = \emptyset$, $\Gamma_C; \Delta_C$ satisfies only $\emptyset$-safety since external choices in Γ_C do not feature a crash handling branch when receiving from $\mathtt{p}$.

Properties by Projection. We conclude by showing the guarantee of safety, deadlock-freedom, and liveness in configurations associated with global types in Lemma 4.30. Furthermore, as a corollary, Theorem 4.31 demonstrates that a typing context projected from a global type (without runtime constructs) is inherently safe, deadlock-free, and live by construction.

Lemma 4.30. *If $\Gamma; \Delta \sqsubseteq_\mathscr{R} \langle \mathscr{C}; G \rangle$, then $\Gamma; \Delta$ is $\mathscr{R}$-safe, $\mathscr{R}$-deadlock-free, and $\mathscr{R}$-live.*

Proof. The results follow from the operational correspondence between global types and configurations (Theorems 4.20 and 4.21), and the respective definitions for each property (Definitions 4.24, 4.26, and 4.28). See Appendix A.6, A.7, and A.8 for details. □

Theorem 4.31 (Safety, Deadlock-Freedom, and Liveness by Projection)**.** *Let G be a global type without runtime constructs, and $\mathscr{R}$ be a set of reliable roles. If Γ is a typing context associated with the global type G: $\Gamma \sqsubseteq_\mathscr{R} G$, then $\Gamma; \Delta_\epsilon$ is $\mathscr{R}$-safe, $\mathscr{R}$-deadlock-free, and $\mathscr{R}$-live.*

Proof. Note $\Gamma \sqsubseteq_\mathscr{R} G$ is an abbreviation of $\Gamma; \Delta_\epsilon \sqsubseteq_\mathscr{R} \langle \emptyset; G \rangle$, apply Lemma 4.30. □

5. A Type System with Crash-Stop Semantics

In this section, we present a type system for our asynchronous multiparty session calculus. Our typing system extends the one in [GPP+21] with crash-stop failures – typing rules for crashed process and unavailable queues, respectively. We introduce the typing rules in Section 5.1, and show various properties of typed sessions: subject reduction (Theorem 5.2), session fidelity (Theorem 5.3), deadlock-freedom (Theorem 5.5), and liveness (Theorem 5.7) in Section 5.2.

5.1. Typing Rules.

Our type system involves four kinds of typing judgements: *(1)* for expressions; *(2)* for processes; *(3)* for queues; and *(4)* for sessions, and is defined inductively by the typing rules in Fig. 9.

Typing judgments for expressions and processes take the forms $\Theta \vdash e : B$ and $\Theta \vdash P : T$, respectively, where Θ is a typing context for expression and process variables, defined as $\Theta ::= \emptyset \mid \Theta, x : B \mid \Theta, X : T$.

With regard to queues, we use judgments of form $\vdash h : \delta$, where we use δ denote a partially applied queue lookup function. We write $\delta = \Delta(-, \mathtt{p})$ to describe the incoming

$$\frac{}{\Theta \vdash \mathtt{n} : \mathsf{nat}} \quad \frac{}{\Theta \vdash \mathtt{i} : \mathsf{int}} \quad \frac{}{\Theta \vdash \mathtt{true} : \mathsf{bool}} \quad \frac{}{\Theta \vdash \mathtt{false} : \mathsf{bool}} \quad \frac{}{\Theta \vdash \texttt{""} : \mathsf{str}}$$

$$\frac{}{\Theta \vdash () : \mathsf{unit}} \quad \frac{}{\Theta, x : B \vdash x : B} \quad \frac{\Theta \vdash e : \mathsf{nat}}{\Theta \vdash \mathtt{succ}(e) : \mathsf{nat}} \quad \frac{\Theta \vdash e : \mathsf{int}}{\Theta \vdash \mathtt{neg}(e) : \mathsf{int}}$$

$$\frac{\Theta \vdash e : \mathsf{bool}}{\Theta \vdash \neg e : \mathsf{bool}} \qquad \frac{\Theta \vdash e_1 : \mathsf{int} \quad \Theta \vdash e_2 : \mathsf{int}}{\Theta \vdash e_1 < e_2 : \mathsf{bool}}$$

$$\frac{}{\vdash \epsilon : \epsilon}\ [\text{T-}\epsilon] \qquad \frac{}{\vdash \oslash : \oslash}\ [\text{T-}\oslash] \qquad \frac{\vdash h_1 : \delta_1 \quad \vdash h_2 : \delta_2}{\vdash h_1 \cdot h_2 : \delta_1 \cdot \delta_2}\ [\text{T-}\cdot]$$

$$\frac{\vdash v : B \quad \delta(\mathtt{q}) = \mathtt{m}(B) \quad \forall \mathtt{r} \neq \mathtt{q} : \delta(\mathtt{r}) = \epsilon}{\vdash (\mathtt{q}, \mathtt{m}(v)) : \delta}\ [\text{T-MSG}]$$

$$\frac{}{\Theta \vdash \lightning : \mathsf{stop}}\ [\text{T-}\lightning] \qquad \frac{}{\Theta \vdash \mathbf{0} : \mathsf{end}}\ [\text{T-}\mathbf{0}] \qquad \frac{\Theta \vdash e : B \quad \Theta \vdash P : T}{\Theta \vdash \mathtt{q}!\mathtt{m}\langle e \rangle.P : \mathtt{q} \oplus \mathtt{m}(B).T}\ [\text{T-OUT}]$$

$$\frac{\forall i \in I \quad \Theta, x_i : B_i \vdash P_i : T_i}{\Theta \vdash \sum_{i \in I} \mathtt{q}?\mathtt{m}_i(x_i).P_i : \mathtt{q} \& \{\mathtt{m}_i(B_i).T_i\}_{i \in I}}\ [\text{T-EXT}] \qquad \frac{\Theta \vdash e : \mathsf{bool} \quad \Theta \vdash P_i : T \ (i = 1, 2)}{\Theta \vdash \mathsf{if}\ e\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 : T}\ [\text{T-COND}]$$

$$\frac{\Theta, X : T \vdash P : T}{\Theta \vdash \mu X.P : T}\ [\text{T-REC}] \qquad \frac{}{\Theta, X : T \vdash X : T}\ [\text{T-VAR}] \qquad \frac{\Theta \vdash P : T \quad T \leqslant T'}{\Theta \vdash P : T'}\ [\text{T-SUB}]$$

$$\frac{\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \quad \forall i \in I \quad \vdash P_i : \Gamma(\mathtt{p}_i) \quad \vdash h_i : \Delta(-, \mathtt{p}_i) \quad \mathrm{dom}(\Gamma) \subseteq \{\mathtt{p}_i \mid i \in I\}}{\langle \mathscr{C}; G \rangle \vdash \prod_{i \in I} (\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i)}\ [\text{T-SESS}]$$

FIGURE 9. Typing rules for expressions, queues, processes, and sessions.

queue for a role $\mathtt{p}$, as a partially applied function $\delta = \Delta(-, \mathtt{p})$ such that $\delta(\mathtt{q}) = \Delta(\mathtt{q}, \mathtt{p})$. We write $\delta_1 \cdot \delta_2$ to denote the point-wise application of concatenation. We lift the process-level constructs for empty queues (ϵ), unavailable queues ($\oslash$), queue concatenations ($\cdot$), and the structural precongruence relation ($\Rrightarrow$) on queues to their corresponding type-level counterparts. For a singleton message $(\mathtt{q}, \mathtt{m}(v))$, the appropriate partial queue δ would be a singleton of $\mathtt{m}(B)$ (where B is the type of v) for $\mathtt{q}$, and an empty queue (ϵ) for any other role.

Finally, we use judgments of form $\langle \mathscr{C}; G \rangle \vdash \mathbb{M}$ for sessions. We use a global type-guided judgment, effectively asserting that all participants in the session respect the prescribed global type, as is the case in [GJP+19]. As highlighted, the global type with crashed roles $\langle \mathscr{C}; G \rangle$ must have some associated configuration $\Gamma; \Delta$, which are used to type the processes and the queues respectively. Moreover, all the entries in the configuration must be present in the session.

We explain the rules in Fig. 9 that assign session types based on process behaviour (other rules are mostly self-explanatory). Rule [T-$\oslash$] (highlighted) assigns the unavailable queue type $\oslash$ to an unavailable queue $\oslash$. Rules [T-OUT] and [T-EXT] assign internal and external choice types to input and output processes, respectively. Additionally, rule [T-$\lightning$] (highlighted) assigns the crash termination stop to a crashed process $\lightning$, while rule [T-$\mathbf{0}$] assigns the successful termination end to an inactive process $\mathbf{0}$.

Example 5.1. Consider our Simpler Logging example (Section 2 and Example 4.25). Specifically, consider the processes that act as the roles C, L, and I respectively:

$$P_{\mathtt{C}} = \mathtt{I}!\mathtt{read}.\mathtt{I}?\mathtt{report}(x).\mathbf{0} \quad P_{\mathtt{L}} = \mathtt{I}!\mathtt{trigger}.\sum\left\{\begin{array}{l}\mathtt{I}?\mathtt{fatal}.\mathbf{0}\\ \mathtt{I}?\mathtt{read}.\mathtt{I}!\mathtt{report}\langle log\rangle.\mathbf{0}\end{array}\right\}$$
$$P_{\mathtt{I}} = \mathtt{L}?\mathtt{trigger}.\sum\left\{\begin{array}{l}\mathtt{C}?\mathtt{read}.\mathtt{L}!\mathtt{read}.\mathtt{L}?\mathtt{report}(x).\mathtt{C}!\mathtt{report}\langle log\rangle.\mathbf{0}\\ \mathtt{C}?\mathtt{crash}.\mathtt{L}!\mathtt{fatal}.\mathbf{0}\end{array}\right\}$$

and message queues $h_{\mathtt{C}} = h_{\mathtt{L}} = h_{\mathtt{I}} = \epsilon$, and the configuration $\Gamma;\Delta$ as in Example 4.25.

Process $P_{\mathtt{C}}$ (resp. $P_{\mathtt{L}}$, $P_{\mathtt{I}}$) has the type $\Gamma(\mathtt{C})$ (resp. $\Gamma(\mathtt{L})$, $\Gamma(\mathtt{I})$), and queue $h_{\mathtt{C}}$ (resp. $h_{\mathtt{L}}$, $h_{\mathtt{I}}$) has the type $\Delta(-,\mathtt{C})$ (resp. $\Delta(-,\mathtt{L})$, $\Delta(-,\mathtt{I})$), which can be verified in the standard way. Then, together with the association $\Gamma;\Delta \sqsubseteq_{\{\mathtt{L},\mathtt{I}\}} \langle\emptyset; G_s\rangle$, we can use [T-SESS] to assert that the session $\mathtt{C} \triangleleft P_{\mathtt{C}} \mid \mathtt{C} \triangleleft h_{\mathtt{C}} \mid \mathtt{L} \triangleleft P_{\mathtt{L}} \mid \mathtt{L} \triangleleft h_{\mathtt{L}} \mid \mathtt{I} \triangleleft P_{\mathtt{I}} \mid \mathtt{I} \triangleleft h_{\mathtt{I}}$ is governed by the global type $\langle\emptyset; G_s\rangle$. If we follow a crash reduction, e.g. by the rule [R-↯], the session evolves as $\mathtt{C} \triangleleft P_{\mathtt{C}} \mid \mathtt{C} \triangleleft h_{\mathtt{C}} \mid \mathtt{L} \triangleleft P_{\mathtt{L}} \mid \mathtt{L} \triangleleft h_{\mathtt{L}} \mid \mathtt{I} \triangleleft P_{\mathtt{I}} \mid \mathtt{I} \triangleleft h_{\mathtt{I}} \to_{\mathscr{R}} \mathtt{C} \triangleleft \lightning \mid \mathtt{C} \triangleleft \oslash \mid \mathtt{L} \triangleleft P_{\mathtt{L}} \mid \mathtt{L} \triangleleft h_{\mathtt{L}} \mid \mathtt{I} \triangleleft P_{\mathtt{I}} \mid \mathtt{I} \triangleleft h_{\mathtt{I}}$, where, by [T-↯], $P_{\mathtt{C}}$ is typed by stop, and $h_{\mathtt{C}}$ is typed by $\oslash$.

5.2. **Properties of Typed Sessions.** We present the main properties of typed sessions: *subject reduction* (Theorem 5.2), *session fidelity* (Theorem 5.3), *deadlock-freedom* (Theorem 5.5), and *liveness* (Theorem 5.7).

Subject reduction states that well-typedness of sessions is preserved by reduction, i.e. a session that is governed by a global type continues to be governed by a global type.

Theorem 5.2 (Subject Reduction)**.** *If* $\langle\mathscr{C}; G\rangle \vdash \mathbb{M}$ *and* $\mathbb{M} \to_{\mathscr{R}} \mathbb{M}'$*, then either* $\langle\mathscr{C}; G\rangle \vdash \mathbb{M}'$*, or there exists* $\langle\mathscr{C}'; G'\rangle$ *such that* $\langle\mathscr{C}; G\rangle \to_{\mathscr{R}} \langle\mathscr{C}'; G'\rangle$ *and* $\langle\mathscr{C}'; G'\rangle \vdash \mathbb{M}'$*.*

Proof. By induction on the derivation of $\mathbb{M} \to_{\mathscr{R}} \mathbb{M}'$. See Appendix B for details. □

Session fidelity states the opposite implication with regard to subject reduction: sessions respect the progress of the governing global type.

Theorem 5.3 (Session Fidelity)**.** *If* $\langle\mathscr{C}; G\rangle \vdash \mathbb{M}$ *and* $\langle\mathscr{C}; G\rangle \to_{\mathscr{R}}$*, then there exists* $\mathbb{M}'$ *and* $\langle\mathscr{C}'; G'\rangle$ *such that* $\langle\mathscr{C}; G\rangle \to_{\mathscr{R}} \langle\mathscr{C}'; G'\rangle$*,* $\mathbb{M} \to^*_{\mathscr{R}} \mathbb{M}'$ *and* $\langle\mathscr{C}'; G'\rangle \vdash \mathbb{M}'$*.*

Proof. By induction on the derivation of $\langle\mathscr{C}; G\rangle \to_{\mathscr{R}}$. See Appendix B for details. □

Session *deadlock-freedom* means that the 'successful' termination of a session may include crashed processes and their respective unavailable incoming queues – but reliable roles (which cannot crash) can only successfully terminate by reaching inactive processes with empty incoming queues. We formalise the definition of deadlock-free sessions in Definition 5.4 and show that a well-typed session is deadlock-free in Theorem 5.5.

Definition 5.4 (Deadlock-Free Sessions)**.** A session $\mathbb{M}$ is *deadlock-free* iff $\mathbb{M} \to^*_{\mathscr{R}} \mathbb{M}' \not\to_{\mathscr{R}}$ implies either $\mathbb{M}' \Rrightarrow \mathtt{p} \triangleleft \mathbf{0} \mid \mathtt{p} \triangleleft \epsilon$ for some $\mathtt{p}$, or $\mathbb{M}' \Rrightarrow \prod_{i\in I}(\mathtt{p}_i \triangleleft \lightning \mid \mathtt{p}_i \triangleleft \oslash)$ with $I \neq \emptyset$.

Theorem 5.5 (Session Deadlock-Freedom)**.** *If* $\langle\mathscr{C}; G\rangle \vdash \mathbb{M}$*, then* $\mathbb{M}$ *is deadlock-free.*

Proof. The result follows by Lemma 4.30 (deadlock-freedom by projection), Theorem 5.2 (subject reduction), and Theorem 5.3 (session fidelity). See Appendix B for details. □

Finally, we show that well-typed sessions guarantee the property of *liveness*: a session is *live* when all its input processes will be performed eventually, and all its queued messages will be consumed eventually. We formalise the definition of live sessions in Definition 5.6 and conclude by showing that a well-typed session is live in Theorem 5.7.

$$
\begin{array}{rcl}
G & = & \mu \mathbf{t}.\mathtt{c}{\to}\mathtt{l},\mathtt{m}_1,\mathtt{m}_2{:}\mathtt{req}.\mathtt{m}_1{\to}\mathtt{l}:\left\{\begin{array}{l}
\mathtt{accept}.\mathtt{m}_2{\to}\mathtt{l}:\left\{\begin{array}{l}
\mathtt{accept}.\mathtt{l}{\to}\mathtt{c}:\left\{\begin{array}{l}\mathtt{accept}.G_1\\ \mathtt{reject}.G_2\\ \mathtt{crash}.G_3\end{array}\right.\\
\mathtt{reject}.\mathtt{l}{\to}\mathtt{c}:\left\{\begin{array}{l}\mathtt{reject}.G_2\\ \mathtt{crash}.G_5\end{array}\right.
\end{array}\right.\\
\mathtt{reject}.\mathtt{m}_2{\to}\mathtt{l}:\left\{\begin{array}{l}
\mathtt{accept}.\mathtt{l}{\to}\mathtt{c}:\left\{\begin{array}{l}\mathtt{reject}.G_2\\ \mathtt{crash}.G_4\end{array}\right.\\
\mathtt{reject}.\mathtt{l}{\to}\mathtt{c}:\left\{\begin{array}{l}\mathtt{reject}.G_2\\ \mathtt{crash}.G_5\end{array}\right.
\end{array}\right.
\end{array}\right.\\
G_1 & = & \mathtt{c}{\to}\mathtt{l},\mathtt{m}_1,\mathtt{m}_2{:}\mathtt{enact}.\mathbf{t}\\
G_2 & = & \mathtt{c}{\to}\mathtt{l},\mathtt{m}_1,\mathtt{m}_2{:}\mathtt{abort}.\mathbf{t}\\
G_3 & = & \mathtt{c}{\to}\mathtt{m}_1{:}\mathtt{promote}.\mathtt{c}{\to}\mathtt{m}_2{:}\mathtt{retry}.\mathtt{m}_2{\to}\mathtt{m}_1{:}\mathtt{accept}.\mathtt{m}_1{\to}\mathtt{c}{:}\mathtt{accept}.\mathtt{c}{\to}\mathtt{m}_1,\mathtt{m}_2{:}\mathtt{enact}.\mathtt{end}\\
G_4 & = & \mathtt{c}{\to}\mathtt{m}_1{:}\mathtt{promote}.\mathtt{c}{\to}\mathtt{m}_2{:}\mathtt{retry}.\mathtt{m}_2{\to}\mathtt{m}_1{:}\mathtt{accept}.\mathtt{m}_1{\to}\mathtt{c}{:}\mathtt{reject}.\mathtt{c}{\to}\mathtt{m}_1,\mathtt{m}_2{:}\mathtt{abort}.\mathtt{end}\\
G_5 & = & \mathtt{c}{\to}\mathtt{m}_1{:}\mathtt{promote}.\mathtt{c}{\to}\mathtt{m}_2{:}\mathtt{retry}.\mathtt{m}_2{\to}\mathtt{m}_1{:}\mathtt{reject}.\mathtt{m}_1{\to}\mathtt{c}{:}\mathtt{reject}.\mathtt{c}{\to}\mathtt{m}_1,\mathtt{m}_2{:}\mathtt{abort}.\mathtt{end}
\end{array}
$$

FIGURE 10. A global type G for a variant of the NBAC abstraction.

Definition 5.6 (Live Sessions)**.** A session $\mathbb{M}$ is *live* iff $\mathbb{M} \to_{\mathscr{R}}^* \mathbb{M}' \Rrightarrow \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathbb{M}''$ implies:

(i) if $h_{\mathtt{p}} = (\mathtt{q}, \mathtt{m}(v)) \cdot h'_{\mathtt{p}}$, then $\exists P', \mathbb{M}''' : \mathbb{M}' \to_{\mathscr{R}}^* \mathtt{p} \triangleleft P' \mid \mathtt{p} \triangleleft h'_{\mathtt{p}} \mid \mathbb{M}'''$;

(ii) if $P = \sum_{i \in I} \mathtt{q}?\mathtt{m}_i(x_i).P_i$, then $\exists k \in I, w, h'_{\mathtt{p}}, \mathbb{M}''' : \mathbb{M}' \to_{\mathscr{R}}^* \mathtt{p} \triangleleft P_k\{{}^{w}/_{x_k}\} \mid \mathtt{p} \triangleleft h'_{\mathtt{p}} \mid \mathbb{M}'''$.

Theorem 5.7 (Session Liveness)**.** *If* $\langle \mathscr{C}; G \rangle \vdash \mathbb{M}$, *then* $\mathbb{M}$ *is live.*

Proof. The result follows by Lemma 4.30 (liveness by projection), Theorem 5.2 (subject reduction), and Theorem 5.3 (session fidelity). See Appendix B for details. □

6. Case Study: Non-Blocking Atomic Commits

We demonstrate our approach via the Non-Blocking Atomic Commits (NBAC) abstraction [CGR11], which enables consistent application of transactions in distributed database systems. The abstraction is straightforward: a given transaction is applied iff all data managers – each representing part of the database – accept the transaction. We present a variant of this abstraction via the global type G in Fig. 10, which involves a co-ordinator $\mathtt{c}$, two data managers $\mathtt{m}_1, \mathtt{m}_2$, and one unreliable leading data manager $\mathtt{l}$. The protocol exhibits failover behaviour, where $\mathtt{m}_1$ is the failover role for $\mathtt{l}$. For clarity, we opt to eliminate the right-hand braces from global types.

The recursive protocol in Fig. 10 begins with the co-ordinator $\mathtt{c}$ broadcasting a transaction proposal, denoted $\mathtt{req}$, to all three data managers. For brevity, we omit payload types and write $\mathtt{c}{\to}\mathtt{l},\mathtt{m}_1,\mathtt{m}_2{:}\mathtt{req}.G'$ for $\mathtt{c}{\to}\mathtt{l}{:}\mathtt{req}.\mathtt{c}{\to}\mathtt{m}_1{:}\mathtt{req}.\mathtt{c}{\to}\mathtt{m}_2{:}\mathtt{req}.G'$. Data managers $\mathtt{m}_1$ and $\mathtt{m}_2$ send their votes to the leading data manager $\mathtt{l}$. In cases where $\mathtt{l}$ does not crash, it sends $\mathtt{reject}$ to $\mathtt{c}$ when it receives a $\mathtt{reject}$ion from its peers, or if $\mathtt{l}$ itself $\mathtt{reject}$s

$$
\begin{aligned}
T_{\mathtt{c}} &= \mu \mathtt{t}.\mathtt{l}, \mathtt{m}_1, \mathtt{m}_2 \oplus \mathtt{req}.\mathtt{l} \,\&\, \begin{cases} \mathtt{accept}.\mathtt{l}, \mathtt{m}_1, \mathtt{m}_2 \oplus \mathtt{enact}.\mathtt{t} \\ \mathtt{reject}.\mathtt{l}, \mathtt{m}_1, \mathtt{m}_2 \oplus \mathtt{abort}.\mathtt{t} \\ \mathtt{crash}.S_{\mathtt{c}} \end{cases} \\
S_{\mathtt{c}} &= \mathtt{m}_1 \oplus \mathtt{promote}.\mathtt{m}_2 \oplus \mathtt{retry}.\mathtt{m}_1 \,\&\, \begin{cases} \mathtt{accept}.\mathtt{m}_1, \mathtt{m}_2 \oplus \mathtt{enact}.\mathtt{end} \\ \mathtt{reject}.\mathtt{m}_1, \mathtt{m}_2 \oplus \mathtt{abort}.\mathtt{end} \end{cases}
\end{aligned}
$$

$$
T_{\mathtt{l}} = \mu \mathtt{t}.\mathtt{c} \,\&\, \mathtt{req}.\mathtt{m}_1 \,\&\, \begin{cases} \mathtt{accept}.\mathtt{m}_2 \,\&\, \begin{cases} \mathtt{accept}.\mathtt{c} \oplus \begin{cases} \mathtt{accept}.\mathtt{c} \,\&\, \mathtt{enact}.\mathtt{t} \\ \mathtt{reject}.\mathtt{c} \,\&\, \mathtt{abort}.\mathtt{t} \end{cases} \\ \mathtt{reject}.\mathtt{c} \oplus \mathtt{reject}.\mathtt{c} \,\&\, \mathtt{abort}.\mathtt{t} \end{cases} \\ \mathtt{reject}.\mathtt{m}_2 \,\&\, \begin{cases} \mathtt{accept}.\mathtt{c} \oplus \mathtt{reject}.\mathtt{c} \,\&\, \mathtt{abort}.\mathtt{t} \\ \mathtt{reject}.\mathtt{c} \oplus \mathtt{reject}.\mathtt{c} \,\&\, \mathtt{abort}.\mathtt{t} \end{cases} \end{cases}
$$

$$
\begin{aligned}
T_{\mathtt{m}_1} &= \mu \mathtt{t}.\mathtt{c} \,\&\, \mathtt{req}.\mathtt{l} \oplus \begin{cases} \mathtt{accept}.\mathtt{c} \,\&\, \{\mathtt{abort}.\mathtt{t},\ \mathtt{enact}.\mathtt{t},\ \mathtt{promote}.S_{\mathtt{m}_1}\} \\ \mathtt{reject}.\mathtt{c} \,\&\, \{\mathtt{abort}.\mathtt{t},\ \mathtt{enact}.\mathtt{t},\ \mathtt{promote}.S'_{\mathtt{m}_1}\} \end{cases} \\
S_{\mathtt{m}_1} &= \mathtt{m}_2 \,\&\, \begin{cases} \mathtt{accept}.\mathtt{c} \oplus \mathtt{accept}.\mathtt{c} \,\&\, \mathtt{enact}.\mathtt{end} \\ \mathtt{reject}.\mathtt{c} \oplus \mathtt{reject}.\mathtt{c} \,\&\, \mathtt{abort}.\mathtt{end} \end{cases} \\
S'_{\mathtt{m}_1} &= \mathtt{m}_2 \,\&\, \begin{cases} \mathtt{accept}.\mathtt{c} \oplus \mathtt{reject}.\mathtt{c} \,\&\, \mathtt{abort}.\mathtt{end} \\ \mathtt{reject}.\mathtt{c} \oplus \mathtt{reject}.\mathtt{c} \,\&\, \mathtt{abort}.\mathtt{end} \end{cases}
\end{aligned}
$$

$$
T_{\mathtt{m}_2} = \mu \mathtt{t}.\mathtt{c} \,\&\, \mathtt{req}.\mathtt{l} \oplus \begin{cases} \mathtt{accept}.\mathtt{c} \,\&\, \begin{cases} \mathtt{abort}.\mathtt{t} \\ \mathtt{enact}.\mathtt{t} \\ \mathtt{retry}.\mathtt{m}_1 \oplus \mathtt{accept}.\mathtt{c} \,\&\, \begin{cases} \mathtt{abort}.\mathtt{end} \\ \mathtt{enact}.\mathtt{end} \end{cases} \end{cases} \\ \mathtt{reject}.\mathtt{c} \,\&\, \begin{cases} \mathtt{abort}.\mathtt{t} \\ \mathtt{retry}.\mathtt{m}_1 \oplus \mathtt{reject}.\mathtt{c} \,\&\, \mathtt{abort}.\mathtt{end} \end{cases} \end{cases}
$$

FIGURE 11. Projected local types for the NBAC abstraction G.

the transaction; c subsequently broadcasts an abort message (in G_2) prior to the protocol recursing. In cases where l, $\mathtt{m}_1$, and $\mathtt{m}_2$ all vote to accept the transaction, c broadcasts a success message (in G_1). In cases where c detects that l has crashed, it promotes $\mathtt{m}_1$ to leader. Once promoted, $\mathtt{m}_1$ receives a vote from $\mathtt{m}_2$, and informs c of the data managers' decision. Should the transaction be rejected (resp. accepted), c broadcasts an abort (resp. a success) message to all live data managers, before the protocol terminates. The global types G_3, G_4, and G_5 define this crash handling behaviour.

All projected local types for the NBAC protocol G are listed in Fig. 11. The type obtained by projecting onto c, $T_{\mathtt{c}}$, is the sole local type that contains a crash handling

$$
\begin{aligned}
P_{\mathtt{c}} &= \mu X.\mathtt{l},\mathtt{m}_1,\mathtt{m}_2!\mathtt{req}.\sum\left\{\begin{array}{l}\mathtt{l}?\mathtt{accept}.\mathtt{l},\mathtt{m}_1,\mathtt{m}_2!\mathtt{enact}.X\\ \mathtt{l}?\mathtt{reject}.\mathtt{l},\mathtt{m}_1,\mathtt{m}_2!\mathtt{abort}.X\\ \mathtt{l}?\mathtt{crash}.Q_{\mathtt{c}}\end{array}\right\}\\
Q_{\mathtt{c}} &= \mathtt{m}_1!\mathtt{promote}.\mathtt{m}_2!\mathtt{retry}.\sum\left\{\begin{array}{l}\mathtt{m}_1?\mathtt{accept}.\mathtt{m}_1,\mathtt{m}_2!\mathtt{enact}.\mathbf{0}\\ \mathtt{m}_1?\mathtt{reject}.\mathtt{m}_1,\mathtt{m}_2!\mathtt{abort}.\mathbf{0}\end{array}\right\}\\
P_{\mathtt{m}_1} &= \mu X.\mathtt{c}?\mathtt{req}.\mathtt{l}!\mathtt{accept}.\sum\left\{\ \mathtt{c}?\mathtt{enact}.X,\ \mathtt{c}?\mathtt{abort}.X,\ \mathtt{c}?\mathtt{promote}.Q_{\mathtt{m}_1}\ \right\}\\
Q_{\mathtt{m}_1} &= \sum\left\{\begin{array}{l}\mathtt{m}_2?\mathtt{accept}.\mathtt{c}!\mathtt{accept}.\mathtt{c}?\mathtt{enact}.\mathbf{0}\\ \mathtt{m}_2?\mathtt{reject}.\mathtt{c}!\mathtt{reject}.\mathtt{c}?\mathtt{abort}.\mathbf{0}\end{array}\right\}
\end{aligned}
$$

FIGURE 12. Process definitions for roles $\mathtt{c}$ and $\mathtt{m}_1$ in G.

branch since $\mathtt{l}$ is the only unreliable role, and both $\mathtt{m}_1$ and $\mathtt{m}_2$ do not receive messages from $\mathtt{l}$. Its crash handling branch is defined by $S_{\mathtt{c}}$. Similarly to the global type above, broadcast operations are denoted as $\mathtt{p}_1,\ldots,\mathtt{p}_n\oplus\mathtt{m}$, and stand for $\mathtt{p}_1\oplus\mathtt{m}.\cdots.\mathtt{p}_n\oplus\mathtt{m}$. Additionally, we remove the right-hand braces from local types as well. Projections for roles $\mathtt{l}$, $\mathtt{m}_1$, and $\mathtt{m}_2$ are as standard, where labels $\mathtt{promote}$ and $\mathtt{retry}$ trigger the crash handling behaviour in $\mathtt{m}_1$ and $\mathtt{m}_2$ respectively.

Let $\Gamma;\Delta$ be the $\{\mathtt{c},\mathtt{m}_1,\mathtt{m}_2\}$-safe configuration, where $\Gamma = \mathtt{c}\triangleright T_{\mathtt{c}},\mathtt{l}\triangleright T_{\mathtt{l}},\mathtt{m}_1\triangleright T_{\mathtt{m}_1},\mathtt{m}_2\triangleright T_{\mathtt{m}_2}$, and $\Delta = \Delta_\epsilon$. The processes $P_{\mathtt{c}}$ and $P_{\mathtt{m}_1}$ in Fig. 12, with message queues $h_{\mathtt{c}} = h_{\mathtt{m}_1} = \epsilon$ act as roles $\mathtt{c}$ and $\mathtt{m}_1$ in G. The former, $P_{\mathtt{c}}$, has type $\Gamma(\mathtt{c})$; the latter, $P_{\mathtt{m}_1}$ will always accept a proposed transaction, and has type $\Gamma(\mathtt{m}_1)$. Processes $P_{\mathtt{l}}$ and $P_{\mathtt{m}_2}$ can be defined similarly such that $P_{\mathtt{l}}$ has type $\Gamma(\mathtt{l})$, and $P_{\mathtt{m}_2}$ has type $\Gamma(\mathtt{m}_2)$. Since the local types in Γ are derived via projection from G, we have the association $\Gamma;\Delta \sqsubseteq_{\{\mathtt{c},\mathtt{m}_1,\mathtt{m}_2\}} \langle\emptyset;G\rangle$. This allows us to assert that the session $\mathtt{c}\triangleleft P_{\mathtt{c}} \mid \mathtt{c}\triangleleft h_{\mathtt{c}} \mid \mathtt{l}\triangleleft P_{\mathtt{l}} \mid \mathtt{l}\triangleleft h_{\mathtt{l}} \mid \mathtt{m}_1\triangleleft P_{\mathtt{m}_1} \mid \mathtt{m}_1\triangleleft h_{\mathtt{m}_1} \mid \mathtt{m}_2\triangleleft P_{\mathtt{m}_2} \mid \mathtt{m}_2\triangleleft h_{\mathtt{m}_2}$, where $h_{\mathtt{l}} = h_{\mathtt{m}_2} = \epsilon$, is governed by the global type $\langle\emptyset;G\rangle$. Consequently, this session is both deadlock-free (by Theorem 5.5) and live (by Theorem 5.7).

7. Related Work

We summarise related work on session types with failure handling. While most of the literature discussed includes delegation, our framework does not. However, this distinction does not impact the core comparison, which focuses on failure handling within session types. The discussion starts with the closest related work [VHEZ21, PNW22, BSYZ22, LBD23] in which multiparty session types are extended to model crashes or failures.

Multiparty Session Types with Failures. Peters *et al.* [PNW22] propose an MPST framework to model fine-grained unreliability. In their work, each transmission in a global type is parameterised by a reliability annotation, which can be unreliable (sender/receiver can crash, and messages can be lost), weakly reliable (sender/receiver can crash, messages are not lost), or reliable (no crashes or message losses). The design choice taken in our work roughly falls under weakly reliable in their work, where a crash handling branch (in their work, a default branch) needs to be present to handle failures. In our work, the reliability assumptions operate on a coarse level, but nonetheless are *consistent* within a given global type – if a role $\mathtt{p}$ is assumed reliable, $\mathtt{p}\in\mathscr{R}$, then it does not crash for the duration of the protocol, and vice versa. Therefore, in a transmission $\mathtt{p}\to\mathtt{q}$, our model allows *one* of the

two roles to be unreliable, whereas their work does not permit the 'mixing' of reliability of sending and receiving roles.

Viering *et al.* [VHEZ21] utilise MPST as a guidance for fault-tolerant distributed systems with recovery mechanisms. Their framework includes various features, such as sub-sessions, event-driven programming, dynamic role assignments, and most importantly failure handling. Our work handles unreliability in distributed programming, with the following differences.

(1) Failure detection *assumptions* and *models* are different: in our work, we assume a perfect failure detector, where all detected crashes are genuine. Their work uses a less strict assumption to allow *false suspicions*. This difference subsequently gives rise to how failures are handled in both approaches: in our work, we use a special crash handling branch in global types to specify how the global protocol should progress after a crash has been detected. In contrast, they use a *try-catch* construct in global types. In such a try-catch construct, crash detection within a sub-session G_1 is specified G_1 with $\mathtt{p@q}$. G_2, where G_2 is a global type for failure handling (where p cannot occur), and q is a *monitor* that monitors p for possible failures. Moreover, the well-formedness condition (2) in [VHEZ21, §4.1] requires the first message in G_2 to be a message broadcast of failure notification from the monitor q to all roles participating in the sub-session (except the crashed role p).

    On this matter, we consider our framework more *flexible* when detecting and handling crashes: every communication construct can have a crash handling branch (when the sender is not assumed reliable), and the failure broadcast is not necessary (failure detection only occurs when receiving from a crashed role).

(2) The *merge* operators, used when projecting global types to obtain local types, are different: we use a more expressive *full* merge operator (Definition 4.3), whereas they use a *plain* merge operator, i.e. requiring all continuations to project to the *same* local type.

(3) *Reliability assumptions* are different: in our work, we support a range of assumptions from every role being unreliable, to totally reliable (as in the literature). In their work, they require *at least one* reliable role, because they use a *monitoring tree* for detecting crashes. Our work allows a role to detect the crash of its communication partner during reception, thus requiring neither such trees nor reliable roles.

Barwell *et al.* [BSYZ22] develop a theory of multiparty session types with crash-stop failures. Their theory models crash-stop failures in the semantics of processes and session types, where the type system uses a model checker to validate type safety. Our theory follows a similar model of crash-stop failures, but differs in the following ways.

(1) We model an asynchronous (message-passing) semantics, whereas they model a synchronous (rendez-vous) semantics. We focus on asynchronous systems, where a message can be buffered while in transit, since most of the interactions in the real distributed world are asynchronous.

(2) We follow a top-down methodology, beginning with protocol specification using global types, whereas they follow [SY19] to analyse only local types. Our method dispenses with the need to use a model checker. More specifically, it is not feasible to model check asynchronous systems with buffers, since the model may be infinite [SY19, Appendix §G].

Le Brun and Dardha [LBD23] follow a similar framework to [BSYZ22]. They model an asynchronous semantics, and support more patterns of failure, including message losses,

delays, reordering, as well as link failures and network partitioning. However, their typing system suffers from its genericity, when type-level properties become undecidable [LBD23, §4.4]. Our work uses global types for guidance, and recovers decidability of properties at a small expense of expressivity.

Other Session Types with Failures. Most other session type works on modelling failures can be briefly categorised into two: using *affine types* or *exceptions* [LNY22,MV18,FLMD19], and using *coordinators* or *supervision* [APN17, VCE+18]. The former adapts session types to an *affine* representation, in which endpoints may cease prematurely; the latter, instead, are usually reliant on one or more *reliable* processes that *coordinate* in the event of failure.

Affine Failure Handling. Mostrous and Vasconcelos [MV18] first proposed the affine approach to failure handling. They present a π-calculus with affine binary sessions (i.e. limited to two participants) and concomitant reduction rules that represent a minimal extension to standard (binary) session types. Their extension is primarily comprised of a *cancel operator*, which is semantically similar to our crash construct: it represents a process that has terminated early. Besides these similarities, our work differs from [MV18] in several ways.

(1) We address multiparty protocols and sessions rather than binary session types.
(2) In [MV18], a cancel operator can have an arbitrary session type; consequently, crashes are not visible at the type level. Instead, we type crashed session endpoints with the special type stop, which lets us model crashes in the type semantics, and helps us in ensuring that a process implements its failure handling as expected in its (global or local) type.
(3) The reduction rules of [MV18] do not permit a process to terminate early arbitrarily: cancellations must be raised explicitly by the programmer (or automatically by attempting to receive messages from crashed endpoints).
(4) Finally, cancellations in [MV18] may be caught and handled via a *do-catch* construct. This construct catches only the first cancellation and cannot be nested, thus providing little help in handling failure across multiple roles. Our global and local types seamlessly support protocols where the failure of a role is detected (and handled) while handling the failure of another role.

Fowler *et al.* [FLMD19] present a concurrent λ-calculus, *EGV*, with asynchronous session-typed communication and exception handling. Their approach is based on [MV18], and therefore shares many of the same differences to our approach: the use of the cancel operator and binary session types, and the lack of a reduction rule enabling a process to crash arbitrarily; the cancel operator used in EGV takes an arbitrary session type – whereas we reflect the crashed status with the dedicated stop type. Similar to our work, [FLMD19] has asynchronous communication channels: messages are queued when sent, and delivered at a later stage.

Lagaillardie *et al.* [LNY22] propose a framework of *affine* multiparty session types. They utilise the affine type system to enforce that failures are handled. In their system, a multiparty session can terminate prematurely. While their theory can be used to model crash-stop failures, such failures are not built into the semantics, so manual encoding of failures is necessary. Moreover, there is no way to recover from a cancellation (i.e. failure) besides propagating the cancellation. In our work, we provide the ability to follow a *different* protocol when a crash is detected, which gives rise to more flexibility and expressivity.

There are some other works adopting *exception-based* approaches, e.g. Capecchi *et al.* [CGY16] and Carbone *et al.* [CHY08] model failure at the *application* level (similar to [MV18, FLMD19]) by equipping processes with constructs to actively produce or catch failures. Conversely, our approach models arbitrary failures (such as hardware failures).

Coordinator Model Failure Handling. Coordinator model approaches find their roots in work by Demangeon *et al.* [DHH$^+$15]. In their model, global types are extended by *interrupt blocks* wherein one role may interrupt the protocol. Interruptions are broadcast to all active roles within the block, and, once interrupted, the roles follow a continuation specified separately. Although their original intention was primarily to interrupt streaming behaviour, interrupt blocks (or similar constructs) have been used to model crashes and failure handling in [APN17, VCE$^+$18].

Adameit *et al.* [APN17] extend the standard MPST syntax with *optional blocks*, representing regions of a protocol that are susceptible to communication failures. In their approach, if a process P expects a value from an optional block which fails, then a *default value* is provided to P, so P can continue running. This ensures termination and deadlock-freedom. Although this approach does not feature an explicit reliable coordinator process, we describe it here due to the inherent coordination required for multiple processes to start and end an optional block. Our approach differs in three key ways.

(1) We model crash-stop process failures instead of impermanent link failures.
(2) We extend the semantics of communications in lieu of introducing a new syntactic construct to enclose the potentially crashing regions of a protocol – our global type projections and typing context safety ensure that crash detection is performed at every pertinent communication point.
(3) We allow crashes to significantly affect the evolution of protocols: our global and local types can have crash detection branches specifying significantly different behaviours w.r.t. non-crashing executions. Conversely, the approach in [APN17] does not discriminate between the presence and absence of failures: both have the same protocol in the optional block's continuation.

Viering *et al.* [VCE$^+$18] similarly extend the standard global type syntax with a *try-handle* construct, which is facilitated by the presence of a reliable coordinator process, and via a construct to specify reliable processes. When the coordinator detects a failure, it broadcasts notifications to all remaining live processes; then, the protocol proceeds according to the failure handling continuation specified as part of the try-handle construct. Our approach and [VCE$^+$18] share several modelling choices: crash-stop semantics, perfect links, and the possibility of specifying reliable processes. However, unlike [VCE$^+$18], our approach does *not* depend on a reliable coordinator that broadcasts failure notifications: all roles in a protocol can be unreliable, all processes may crash.

Besides the differences discussed above, we decided not to adopt coordinator processes nor failure broadcasts in order to avoid their inherent drawbacks. The use of a coordinator requires additional run-time resources and increases the overall complexity of a distributed system. Furthermore, the broadcasting of failure notifications introduces an effective synchronisation point for all roles, with additional overheads. Such synchronisation points may also make it harder to extend the theory to support scenarios with unreliable communication.

Alternative Session Types with Failure Handling. Caires and Pérez [CP17] present a linearly-typed calculus that supports non-determinism and process failures. They present a core calculus with binary session types (based on linear logic), extended with constructs permitting non-determinism and control effects (e.g. exceptions). Failure is simulated via a non-deterministic choice operator. The main difference is that we model arbitrary failures, and support multiparty sessions.

Neykova and Yoshida [NY17] build upon the *Let it Crash* and supervisor-based design of the Erlang programming language. Using an MPST specification, they build a dependency graph between the processes interacting in a system; in case of failure, this information is then used by supervisor processes to determine which subset of processes should be restarted, and which messages should be re-sent. As in the coordinator model approaches above, Erlang supervisors result in additional overhead compared with our approach. Furthermore, as in the affine session types approach [MV18, FLMD19], failure states are not reflected in the type system. Instead, there is an underlying assumption that the supervisors will restart (a subset of) the system until it terminates successfully.

Chen *et al.* [CVB+16] handle partial failures in MPST by transforming programs to include synchronisation points, at which failures can be detected and handled. Our approach is less rigid due to the handling of failures: synchronisation points are not needed (unless required by the protocol), and the original protocols do not need to meet certain criteria to permit their transformation.

8. Conclusion and Future Work

To overcome the challenge of accounting for failure handling in distributed systems using session types, we present an asynchronous multiparty session type framework with the ability to model and handle crash-stop failures, e.g. caused by hardware faults. In our session calculus, processes may crash arbitrarily, and other processes can detect their crash and handle the event. Additionally, our system supports the specification of *optional reliability assumptions*: by influencing type-checking, they allow our approach to cover the spectrum between idealised scenarios where no process ever fails (as typically assumed in standard session type works), and more realistic scenarios where any process can fail at any time. With only minimal changes to the syntax of standard asynchronous MPST in our theory, desirable global type properties such as deadlock-freedom, protocol conformance, and liveness are preserved by construction in typed processes, even in the presence of crashes.

This work is a new step towards modelling and handling real-world failures by utilising session types, effectively bridging the gap between session type theory and applications. As future work, we plan to study different crash models (e.g. crash-recover) and failures of other components (e.g. link failures). These further steps would lead us to our longer term goal, i.e. to eventually model and type-check implementations of well-known consensus algorithms used in large-scale distributed systems.

Acknowledgments. We thank the reviewers for their detailed and helpful comments. Additionally, we thank Yoshinori Kamegai for highlighting an issue related to structural congruence. The work is partially supported by EPSRC grants EP/T006544/2, EP/K011715/1, EP/K034413/1, EP/L00058X/1, EP/N027833/2, EP/N028201/1, EP/T014709/2, EP/V000462/1, EP/X015955/1, EP/Y005244/1, NCSS/EPSRC VeTSS, Horizon EU TaRDIS 101093006, Advanced Research and Invention Agency (ARIA) Safeguarded AI, and a grant from the Simons Foundation.

References

[APN17] Manuel Adameit, Kirstin Peters, and Uwe Nestmann. Session types for link failures. In Ahmed Bouajjani and Alexandra Silva, editors, *Formal Techniques for Distributed Objects, Components, and Systems - 37th IFIP WG 6.1 International Conference, FORTE 2017, Held as Part of the 12th International Federated Conference on Distributed Computing Techniques, DisCoTec 2017, Neuchâtel, Switzerland, June 19-22, 2017, Proceedings*, volume 10321 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2017. doi:10.1007/978-3-319-60225-7_1.

[BHYZ23] Adam D. Barwell, Ping Hou, Nobuko Yoshida, and Fangyi Zhou. Designing asynchronous multiparty protocols with crash-stop failures. In Karim Ali and Guido Salvaneschi, editors, *37th European Conference on Object-Oriented Programming, ECOOP 2023, July 17-21, 2023, Seattle, Washington, United States*, volume 263 of *LIPIcs*, pages 1:1–1:30. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ECOOP.2023.1.

[BSYZ22] Adam D. Barwell, Alceste Scalas, Nobuko Yoshida, and Fangyi Zhou. Generalised Multiparty Session Types with Crash-Stop Failures. In Bartek Klin, Sławomir Lasota, and Anca Muscholl, editors, *33rd International Conference on Concurrency Theory (CONCUR 2022)*, volume 243 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:25, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2022/17098, doi:10.4230/LIPIcs.CONCUR.2022.35.

[CGR11] Christian Cachin, Rachid Guerraoui, and Luís E. T. Rodrigues. *Introduction to Reliable and Secure Distributed Programming (2. ed.)*. Springer, 2011. doi:10.1007/978-3-642-15260-3.

[CGY16] Sara Capecchi, Elena Giachino, and Nobuko Yoshida. Global escape in multiparty sessions. *Math. Struct. Comput. Sci.*, 26(2):156–205, 2016. doi:10.1017/S0960129514000164.

[CHY08] Marco Carbone, Kohei Honda, and Nobuko Yoshida. Structured Interactional Exceptions in Session Types. In Franck van Breugel and Marsha Chechik, editors, *CONCUR 2008 - Concurrency Theory, 19th International Conference, CONCUR 2008, Toronto, Canada, August 19-22, 2008. Proceedings*, volume 5201 of *Lecture Notes in Computer Science*, pages 402–417. Springer, 2008. doi:10.1007/978-3-540-85361-9_32.

[CP17] Luís Caires and Jorge A. Pérez. Linearity, Control Effects, and Behavioral Types. In Hongseok Yang, editor, *Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, volume 10201 of *Lecture Notes in Computer Science*, pages 229–259. Springer, 2017. doi:10.1007/978-3-662-54434-1_9.

[CT96] Tushar Deepak Chandra and Sam Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *J. ACM*, 43(2):225–267, March 1996. doi:10.1145/226643.226647.

[CVB+16] Tzu-Chun Chen, Malte Viering, Andi Bejleri, Lukasz Ziarek, and Patrick Eugster. A Type Theory for Robust Failure Handling in Distributed Systems. In Elvira Albert and Ivan Lanese, editors, *Formal Techniques for Distributed Objects, Components, and Systems - 36th IFIP WG 6.1 International Conference, FORTE 2016, Held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec 2016, Heraklion, Crete, Greece, June 6-9, 2016, Proceedings*, volume 9688 of *Lecture Notes in Computer Science*, pages 96–113. Springer, 2016. doi:10.1007/978-3-319-39570-8_7.

[DHH+15] Romain Demangeon, Kohei Honda, Raymond Hu, Rumyana Neykova, and Nobuko Yoshida. Practical interruptible conversations: distributed dynamic verification with multiparty session types and Python. *Formal Methods Syst. Des.*, 46(3):197–225, 2015. doi:10.1007/s10703-014-0218-8.

[DY13] Pierre-Malo Deniélou and Nobuko Yoshida. Multiparty Compatibility in Communicating Automata: Characterisation and Synthesis of Global Session Types. In *40th International Colloquium on Automata, Languages and Programming*, volume 7966 of *LNCS*, pages 174–186, Berlin, Heidelberg, 2013. Springer. doi:10.1007/978-3-642-39212-2_18.

[DY15] Romain Demangeon and Nobuko Yoshida. On the Expressiveness of Multiparty Sessions. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015)*, volume 45 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 560–574, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2015/5621, doi:10.4230/LIPIcs.FSTTCS.2015.560.

[FLMD19] Simon Fowler, Sam Lindley, J. Garrett Morris, and Sára Decova. Exceptional Asynchronous Session Types: Session Types without Tiers. *Proc. ACM Program. Lang.*, 3(POPL):28:1–28:29, 2019. doi:10.1145/3290341.

[GJP+19] Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic, Alceste Scalas, and Nobuko Yoshida. Precise subtyping for synchronous multiparty sessions. *J. Log. Algebraic Methods Program.*, 104:127–173, 2019. doi:10.1016/j.jlamp.2018.12.002.

[GPP+21] Silvia Ghilezan, Jovanka Pantović, Ivan Prokić, Alceste Scalas, and Nobuko Yoshida. Precise Subtyping for Asynchronous Multiparty Sessions. *Proc. ACM Program. Lang.*, 5(POPL), January 2021. doi:10.1145/3434297.

[HVK98] Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo. Language primitives and type discipline for structured communication-based programming. In Chris Hankin, editor, *Programming Languages and Systems*, pages 122–138, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. doi:10.1007/BFb0053567.

[HYC08] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty Asynchronous Session Types. In *35th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 273–284. ACM, 2008. doi:10.1145/1328897.1328472.

[HYC16] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty Asynchronous Session Types. *Journal of the ACM*, 63:1–67, 2016. doi:10.1145/2827695.

[LBD23] Matthew Alan Le Brun and Ornela Dardha. MAGπ: Types for Failure-Prone Communication. In Thomas Wies, editor, *Programming Languages and Systems*, pages 363–391, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-30044-8_14.

[LNY22] Nicolas Lagaillardie, Rumyana Neykova, and Nobuko Yoshida. Stay Safe Under Panic: Affine Rust Programming with Multiparty Session Types. In Karim Ali and Jan Vitek, editors, *36th European Conference on Object-Oriented Programming (ECOOP 2022)*, volume 222 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:29, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2022/16232.

[MV18] Dimitris Mostrous and Vasco T. Vasconcelos. Affine Sessions. *Logical Methods in Computer Science*, Volume 14, Issue 4, November 2018. URL: https://lmcs.episciences.org/4973, doi:10.23638/LMCS-14(4:14)2018.

[NY17] Rumyana Neykova and Nobuko Yoshida. Let It Recover: Multiparty Protocol-Induced Recovery. In *26th International Conference on Compiler Construction*, pages 98–108. ACM, 2017. doi:10.1145/3033019.3033031.

[Pie02] Benjamin C. Pierce. *Types and Programming Languages*. The MIT Press, 1st edition, 2002.

[PNW22] Kirstin Peters, Uwe Nestmann, and Christoph Wagner. Fault-tolerant multiparty session types. In Mohammad Reza Mousavi and Anna Philippou, editors, *Formal Techniques for Distributed Objects, Components, and Systems*, pages 93–113, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-08679-3_7.

[San11] Davide Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge University Press, 2011. doi:10.1017/CBO9780511777110.

[SY19] Alceste Scalas and Nobuko Yoshida. Less is More: Multiparty Session Types Revisited. *Proc. ACM Program. Lang.*, 3(POPL):30:1–30:29, January 2019. doi:10.1145/3290343.

[VCE+18] Malte Viering, Tzu-Chun Chen, Patrick Eugster, Raymond Hu, and Lukasz Ziarek. A Typing Discipline for Statically Verified Crash Failure Handling in Distributed Systems. In Amal Ahmed, editor, *Programming Languages and Systems*, pages 799–826, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-89884-1_28.

[vGHH21] Rob van Glabbeek, Peter Höfner, and Ross Horne. Assuming Just Enough Fairness to make Session Types Complete for Lock-freedom. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470531.

[VHEZ21] Malte Viering, Raymond Hu, Patrick Eugster, and Lukasz Ziarek. A Multiparty Session Typing Discipline for Fault-Tolerant Event-Driven Distributed Programming. *Proc. ACM Program. Lang.*, 5(OOPSLA), October 2021. doi:10.1145/3485501.

Table of Contents for Appendix

Appendix A. Proofs for Section 4

With regard to recursive global types, we define their *unfolding* as $\mathrm{unf}(\mu \mathbf{t}.G) = \mathrm{unf}(G\{^{\mu \mathbf{t}.G}/_{\mathbf{t}}\})$, and $\mathrm{unf}(G) = G$ otherwise. A recursive type $\mu \mathbf{t}.G$ must be guarded (or contractive), i.e. the unfolding leads to a progressive prefix, e.g. a transmission. Unguarded types, such as $\mu \mathbf{t}.\mathbf{t}$ and $\mu \mathbf{t}.\mu \mathbf{t}'.\mathbf{t}$, are excluded. Similar definitions and requirements apply for local types.

A.1. Live and Crashed Roles.

Lemma A.1. *If* $\mathtt{p} \in \mathrm{roles}(G)$, *then* $\mathtt{p} \notin \mathrm{roles}(G \not\downarrow \mathtt{p})$ *and* $\mathrm{roles}(G \not\downarrow \mathtt{p}) \subseteq \mathrm{roles}(G)$.

Proof. By induction on Definition 4.10. We detail interesting cases here:

(1)

$$\mathrm{roles}((\mathtt{p} \to \mathtt{q}\colon \{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I}) \not\downarrow \mathtt{p}) = \mathrm{roles}(\mathtt{p}^{\not\downarrow} \rightsquigarrow \mathtt{q}\colon j\, \{\mathtt{m}_\mathtt{i}(B_i).(G_i \not\downarrow \mathtt{p})\}_{i\in I}) = \{\mathtt{q}\} \cup \bigcup_{i\in I} \mathrm{roles}(G_i \not\downarrow \mathtt{p}).$$

The required result follows by inductive hypothesis that $\mathtt{p} \notin \mathrm{roles}(G_i \not\downarrow \mathtt{p})$, and $\mathrm{roles}(G_i \not\downarrow \mathtt{p}) \subseteq \mathrm{roles}(G_i)$.

(2)

$$\mathrm{roles}((\mathtt{p}^{\not\downarrow} \rightsquigarrow \mathtt{q}\colon j\, \{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I}) \not\downarrow \mathtt{q}) = \mathrm{roles}(G_j \not\downarrow \mathtt{q})$$

The required result follows by inductive hypothesis that $\mathtt{q} \notin \mathrm{roles}(G_j \not\downarrow \mathtt{q})$, and $\mathrm{roles}(G_j \not\downarrow \mathtt{q}) \subseteq \mathrm{roles}(G_j)$.

The rest of the cases are similar or straightforward. □

Lemma A.2. *If* $\mathtt{p} \in \mathrm{roles}(G)$, *then* $\mathrm{roles}^{\not\downarrow}(G \not\downarrow \mathtt{p}) \setminus \{\mathtt{p}\} \subseteq \mathrm{roles}^{\not\downarrow}(G)$.

Proof. By induction on Definition 4.10. We detail interesting cases here:

(1)

$$\mathrm{roles}^{\not\downarrow}((\mathtt{p} \to \mathtt{q}\colon \{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I}) \not\downarrow \mathtt{p}) = \mathrm{roles}^{\not\downarrow}(\mathtt{p}^{\not\downarrow} \rightsquigarrow \mathtt{q}\colon j\, \{\mathtt{m}_\mathtt{i}(B_i).(G_i \not\downarrow \mathtt{p})\}_{i\in I}) = \bigcup_{i\in I} \mathrm{roles}^{\not\downarrow}(G_i \not\downarrow \mathtt{p}).$$

The required result follows by inductive hypothesis that $\mathrm{roles}^{\not\downarrow}(G_i \not\downarrow \mathtt{p}) \setminus \{\mathtt{p}\} \subseteq \mathrm{roles}^{\not\downarrow}(G_i)$.

(2)

$$\mathrm{roles}^{\not\downarrow}((\mathtt{p}^{\not\downarrow} \rightsquigarrow \mathtt{q}\colon j\, \{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I}) \not\downarrow \mathtt{q}) = \mathrm{roles}^{\not\downarrow}(G_j \not\downarrow \mathtt{q})$$

The required result follows by inductive hypothesis that $\mathrm{roles}^{\not\downarrow}(G_j \not\downarrow \mathtt{q}) \setminus \{\mathtt{q}\} \subseteq \mathrm{roles}^{\not\downarrow}(G_j)$.

The rest of the cases are similar or straightforward. □

Lemma A.3. *If* $G \neq \mu \mathbf{t}.G'$ *and* $\mathtt{p} \in \mathrm{roles}(G)$*, then* $G \upharpoonright_{\mathscr{R}} \mathtt{p} \neq \mathsf{end}$.

Proof. We know that $G \neq \mathsf{end}$; otherwise, we may have $\mathrm{roles}(G) = \emptyset$, a contradiction to $\mathtt{p} \in \mathrm{roles}(G)$. By induction on the structure of G:

- Case $G = \mathtt{q} \to \mathtt{r} : \{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i \in I}$:
  We perform case analysis on $\mathtt{p}$:
  - $\mathtt{p} = \mathtt{q}$: we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{r} \oplus \{\mathtt{m}_\mathtt{i}(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i \in \{j \in I \mid \mathtt{m}_j \neq \mathsf{crash}\}} \neq \mathsf{end}$.
  - $\mathtt{p} = \mathtt{r}$: we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{q} \& \{\mathtt{m}_\mathtt{i}(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i \in I} \neq \mathsf{end}$.
  - $\mathtt{p} \neq \mathtt{q}$ and $\mathtt{p} \neq \mathtt{r}$: we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \bigsqcap_{i \in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{p}$. Since $\mathtt{p} \in \mathrm{roles}(G)$, $\mathtt{p} \neq \mathtt{q}$, and $\mathtt{p} \neq \mathtt{r}$, there exists $j \in I$ such that $\mathtt{p} \in \mathrm{roles}(G_j)$. Then, by applying inductive hypothesis, $G_j \upharpoonright_{\mathscr{R}} \mathtt{p} \neq \mathsf{end}$, and therefore, we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \bigsqcap_{i \in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{p} = G_j \upharpoonright_{\mathscr{R}} \mathtt{p} \sqcap \bigsqcap_{i \in I \setminus \{j\}} G_i \upharpoonright_{\mathscr{R}} \mathtt{p} \neq \mathsf{end}$.
- Case $G = \mathtt{q} \rightsquigarrow \mathtt{r} : j \, \{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i \in I}$:
  We perform case analysis on $\mathtt{p}$:
  - $\mathtt{p} \neq \mathtt{q}$ and $\mathtt{p} \neq \mathtt{r}$: we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \bigsqcap_{i \in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{p}$. Since $\mathtt{p} \in \mathrm{roles}(G)$, $\mathtt{p} \neq \mathtt{q}$, and $\mathtt{p} \neq \mathtt{r}$, there exists $k \in I$ such that $\mathtt{p} \in \mathrm{roles}(G_k)$. Then, by applying inductive hypothesis, $G_k \upharpoonright_{\mathscr{R}} \mathtt{p} \neq \mathsf{end}$, and therefore, we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \bigsqcap_{i \in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{p} = G_k \upharpoonright_{\mathscr{R}} \mathtt{p} \sqcap \bigsqcap_{i \in I \setminus \{k\}} G_i \upharpoonright_{\mathscr{R}} \mathtt{p} \neq \mathsf{end}$, as desired. Meanwhile, we obtain that $\forall l \in I : G_l \upharpoonright_{\mathscr{R}} \mathtt{p} \neq \mathsf{end}$.
  - $\mathtt{p} = \mathtt{q}$: we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = G_j \upharpoonright_{\mathscr{R}} \mathtt{p}$, which follows that $G \upharpoonright_{\mathscr{R}} \mathtt{p} \neq \mathsf{end}$ by the fact that $\forall l \in I : G_l \upharpoonright_{\mathscr{R}} \mathtt{p} \neq \mathsf{end}$.
  - $\mathtt{p} = \mathtt{r}$: we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{q} \& \{\mathtt{m}_\mathtt{i}(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i \in I} \neq \mathsf{end}$.

Other cases are similar. □

Lemma A.4. *If* $\mathtt{p} \notin \mathrm{roles}(G)$ *and* $\mathtt{p} \notin \mathrm{roles}^{\lightning}(G)$*, then* $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathsf{end}$.

Proof. By induction on the structure of G:

- Case $G = \mathtt{q} \to \mathtt{r} : \{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i \in I}$: since $\mathtt{p} \notin \mathrm{roles}(G)$ and $\mathtt{p} \notin \mathrm{roles}^{\lightning}(G)$, we have $\mathtt{p} \neq \mathtt{q}$, $\mathtt{p} \neq \mathtt{r}$, and for all $i \in I$, $\mathtt{p} \notin \mathrm{roles}(G_i)$ and $\mathtt{p} \notin \mathrm{roles}^{\lightning}(G_i)$ by Definition 4.1. Thus, $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \bigsqcap_{i \in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathsf{end}$ by applying inductive hypothesis and $\mathsf{end} \sqcap \mathsf{end} = \mathsf{end}$.
- Case $G = \mu \mathbf{t}.G'$: since $\mathtt{p} \notin \mathrm{roles}(G)$ and $\mathtt{p} \notin \mathrm{roles}^{\lightning}(G)$, we have $\mathtt{p} \notin \mathrm{roles}(G')$ and $\mathtt{p} \notin \mathrm{roles}^{\lightning}(G')$ by Definition 4.1. We have two further subcases to consider:
  - If $\mathrm{fv}(\mu \mathbf{t}.G') \neq \emptyset$, we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \mu \mathbf{t}.(G' \upharpoonright_{\mathscr{R}} \mathtt{p}) = \mu \mathbf{t}.\mathsf{end} = \mathsf{end}$ by applying inductive hypothesis.
  - Otherwise, we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathsf{end}$ immediately.

Other cases are similar or trivial. □

A.2. **Subtyping.**

Lemma A.5 (Subtyping is Reflexive)**.** *For any closed, well-guarded local type* T*,* $T \leqslant T$ *holds.*

Proof. We construct a relation $R = \{(T, T)\}$. It is straightforward that R satisfies all clauses of Definition 4.4. Hence, since $\leqslant$ is the largest relation satisfying such rules, $R \subseteq \leqslant$. □

Lemma A.6 (Subtyping is Transitive)**.** *For any closed, well-guarded local type* S*,* T*,* U*, if* $S \leqslant T$ *and* $T \leqslant U$ *hold, then* $S \leqslant U$ *holds.*

Proof. We construct a relation $R = \{(S, U) \mid \exists T \text{ such that } S \leqslant T \text{ and } T \leqslant U\}$. By showing that R satisfies all clauses of Definition 4.4, it follows that $R \subseteq \leqslant$. □

Lemma 4.5 (Reflexivity and Transitivity of Subtyping)**.** *The subtyping relation $\leqslant$ is reflexive and transitive.*

Proof. Immediate consequence of Lemma A.5 and Lemma A.6. □

Lemma A.7. *For any closed, well-guarded local type T,* (1) $\mathrm{unf}(T) \leqslant T$*; and* (2) $T \leqslant \mathrm{unf}(T)$.

Proof. (1) If $T = \mu \mathbf{t}.T'$, $\mathrm{unf}(T) \leqslant T$ holds by [Sub-μR]. Otherwise, by Lemma A.5. (2) If $T = \mu \mathbf{t}.T'$, $T \leqslant \mathrm{unf}(T)$ holds by [Sub-μL]. Otherwise, by Lemma A.5. □

Lemma 4.6. *Given a collection of mergable local types T_i $(i \in I)$. For all $j \in I$, $\bigsqcap_{i \in I} T_i \leqslant T_j$ holds.*

Proof. By constructing a relation $R = \left\{\left(\bigsqcap_{i \in I} T_i, T_j\right) \middle| j \in I\right\}$, and showing that R satisfies all clauses of Definition 4.4. □

Lemma 4.7. *Given a collection of mergable local types T_i $(i \in I)$. If for all $i \in I$, $S \leqslant T_i$ for some local type S, then $S \leqslant \bigsqcap_{i \in I} T_i$.*

Proof. By constructing a relation $R = \left\{\left(S, \bigsqcap_{i \in I} T_i\right)\right\}$, and showing that R satisfies all clauses of Definition 4.4. □

Lemma 4.8. *Given two collections of mergable local types S_i, T_i $(i \in I)$. If for all $i \in I$, $S_i \leqslant T_i$, then $\bigsqcap_{i \in I} S_i \leqslant \bigsqcap_{i \in I} T_i$.*

Proof. By constructing a relation $R = \left\{\left(\bigsqcap_{i \in I} S_i, \bigsqcap_{i \in I} T_i\right)\right\}$, and showing that R satisfies all clauses of Definition 4.4. □

Lemma A.8. *If $\mathtt{p}, \mathtt{q} \in \mathrm{roles}(G)$ with $\mathtt{p} \neq \mathtt{q}$ and $\mathtt{q} \notin \mathscr{R}$, then $G \upharpoonright_{\mathscr{R}} \mathtt{p} \leqslant (G \downarrow \mathtt{q}) \upharpoonright_{\mathscr{R}} \mathtt{p}$.*

Proof. We construct a relation $R = \{(G \upharpoonright_{\mathscr{R}} \mathtt{p}, (G \downarrow \mathtt{q}) \upharpoonright_{\mathscr{R}} \mathtt{p}) \mid \mathtt{p}, \mathtt{q} \in \mathcal{R}, \mathtt{p} \neq \mathtt{q}, \mathtt{q} \in \mathscr{R}\}$, and show that $R \subseteq \leqslant$. Consider all possible shapes of G:

- Case $G = \mathtt{p} \to \mathtt{q}\colon \{\mathtt{m}_i(B_i).G_i\}_{i \in I}$:
  We perform case analysis on the role being projected upon:
  - On LHS, we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{q} \oplus \{\mathtt{m}_i(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i \in \{j \in I \mid \mathtt{m}_j \neq \mathsf{crash}\}}$.
    On RHS, we perform case analysis on the role being removed:
    (1) we have $G \downarrow \mathtt{q} = \mathtt{p} \to \mathtt{q}^{\downarrow}\colon \{\mathtt{m}_i(B_i).(G_i \downarrow \mathtt{q})\}_{i \in I}$, and thus $(G \downarrow \mathtt{q}) \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{q} \oplus \{\mathtt{m}_i(B_i).((G_i \downarrow \mathtt{q}) \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i \in \{j \in I \mid \mathtt{m}_j \neq \mathsf{crash}\}}$, apply [Sub-$\oplus$] and coinductive hypothesis.
    (2) ($\mathtt{r} \neq \mathtt{q}$) we have $G \downarrow \mathtt{r} = \mathtt{p} \to \mathtt{q}\colon \{\mathtt{m}_i(B_i).(G_i \downarrow \mathtt{r})\}_{i \in I}$, and thus $(G \downarrow \mathtt{r}) \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{q} \oplus \{\mathtt{m}_i(B_i).((G_i \downarrow \mathtt{r}) \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i \in \{j \in I \mid \mathtt{m}_j \neq \mathsf{crash}\}}$, apply [Sub-$\oplus$] and coinductive hypothesis.
  - On LHS, we have $G \upharpoonright_{\mathscr{R}} \mathtt{q} = \mathtt{p} \& \{\mathtt{m}_i(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{q})\}_{i \in I}$.
    On RHS, we perform case analysis on the role being removed:
    (1) we have $G \downarrow \mathtt{p} = \mathtt{p}^{\downarrow} \rightsquigarrow \mathtt{q}\colon j\, \{\mathtt{m}_i(B_i).(G_i \downarrow \mathtt{p})\}_{i \in I}$, and thus $(G \downarrow \mathtt{p}) \upharpoonright_{\mathscr{R}} \mathtt{q} = \mathtt{p} \& \{\mathtt{m}_i(B_i).((G_i \downarrow \mathtt{p}) \upharpoonright_{\mathscr{R}} \mathtt{q})\}_{i \in I}$, apply [Sub-&] and coinductive hypothesis.
    (2) ($\mathtt{r} \neq \mathtt{p}$) we have $G \downarrow \mathtt{r} = \mathtt{p} \to \mathtt{q}\colon \{\mathtt{m}_i(B_i).(G_i \downarrow \mathtt{r})\}_{i \in I}$, and thus $(G \downarrow \mathtt{r}) \upharpoonright_{\mathscr{R}} \mathtt{q} = \mathtt{p} \& \{\mathtt{m}_i(B_i).((G_i \downarrow \mathtt{r}) \upharpoonright_{\mathscr{R}} \mathtt{q})\}_{i \in I}$, apply [Sub-&] and coinductive hypothesis.
  - ($\mathtt{r} \notin \{\mathtt{p}, \mathtt{q}\}$) On LHS, we have $G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i \in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$.
    On RHS, we perform case analysis on the role being removed:

(1) we have $G\!\upharpoonright\!\mathtt{p} = \mathtt{p}^{\natural} \rightsquigarrow \mathtt{q}{:}j\,\{\mathtt{m}_{\mathtt{i}}(B_i).(G_i\!\upharpoonright\!\mathtt{p})\}_{i\in I}$, and thus $(G\!\upharpoonright\!\mathtt{p}) \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i\in I}((G_i\!\upharpoonright\!\mathtt{p}) \upharpoonright_{\mathscr{R}} \mathtt{r})$, apply Lemma 4.8 and coinductive hypothesis.
(2) we have $G\!\upharpoonright\!\mathtt{q} = \mathtt{p} \rightarrow \mathtt{q}^{\natural}{:}\{\mathtt{m}_{\mathtt{i}}(B_i).(G_i\!\upharpoonright\!\mathtt{q})\}_{i\in I}$, and thus $(G\!\upharpoonright\!\mathtt{q}) \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i\in I}((G_i\!\upharpoonright\!\mathtt{q}) \upharpoonright_{\mathscr{R}} \mathtt{r})$, apply Lemma 4.8 and coinductive hypothesis.
(3) ($\mathtt{s} \notin \{\mathtt{p},\mathtt{q},\mathtt{r}\}$) we have $G\!\upharpoonright\!\mathtt{s} = \mathtt{p} \rightarrow \mathtt{q}{:}\{\mathtt{m}_{\mathtt{i}}(B_i).(G_i\!\upharpoonright\!\mathtt{s})\}_{i\in I}$, and thus $(G\!\upharpoonright\!\mathtt{s}) \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i\in I}((G_i\!\upharpoonright\!\mathtt{s}) \upharpoonright_{\mathscr{R}} \mathtt{r})$, apply Lemma 4.8 and coinductive hypothesis.

- Case $G = \mu\mathtt{t}.G'$:
  By coinductive hypothesis.

Other cases are similar or trivial. □

Lemma A.9 (Inversion of Subtyping)**.**

(1) *If* $S \leqslant \mathtt{p}\oplus\{\mathtt{m}_{\mathtt{i}}(B_i).T_i\}_{i\in I}$, *then* $\mathrm{unf}(S) = \mathtt{p}\oplus\left\{\mathtt{m}'_{\mathtt{j}}(B'_j).T'_j\right\}_{j\in J}$, *and* $J \subseteq I$, *and* $\forall i \in J$: $\mathtt{m}_i = \mathtt{m}'_i, B_i = B'_i$ *and* $T'_i \leqslant T_i$.
(2) *If* $S \leqslant \mathtt{p}\&\{\mathtt{m}_{\mathtt{i}}(B_i).T_i\}_{i\in I}$, *then* $\mathrm{unf}(S) = \mathtt{p}\&\left\{\mathtt{m}'_{\mathtt{j}}(B'_j).T'_j\right\}_{j\in J}$, *and* $I \subseteq J$, *and* $\forall i \in I$: $\mathtt{m}_i = \mathtt{m}'_i, B_i = B'_i$ *and* $T'_i \leqslant T_i$.

Proof. By Lemma A.7, the transitivity of subtyping, and Definition 4.4 ([Sub-&], [Sub-⊕]). □

A.3. **Semantics of Global Types.**

Lemma 4.14 (No Revival Or Unexpected Crashes)**.** *Assume* $\langle\mathscr{C};G\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle\mathscr{C}';G'\rangle$.
(1) *If* $\mathtt{p} \in \mathrm{roles}^{\natural}(G')$ *and* $\alpha \neq \mathtt{p}\natural$, *then* $\mathtt{p} \in \mathrm{roles}^{\natural}(G)$;
(2) *If* $\mathtt{p} \in \mathrm{roles}(G')$ *and* $\alpha \neq \mathtt{p}\natural$, *then* $\mathtt{p} \in \mathrm{roles}(G)$;
(3) *If* $\mathtt{p} \in \mathrm{roles}^{\natural}(G')$ *and* $\alpha = \mathtt{p}\natural$, *then* $\mathtt{p} \in \mathrm{roles}(G)$.

Proof. (1) By induction on global type reductions: since $\alpha \neq \mathtt{p}\natural$, we start from [GR-$\mu$].

- Case [GR-μ]: we have $G = \mu\mathtt{t}.G''$ and $\langle\mathscr{C};G''\{\mu\mathtt{t}.G''/\mathtt{t}\}\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle\mathscr{C}';G'\rangle$ by [GR-μ] and its inversion. Hence, by $\langle\mathscr{C};G''\{\mu\mathtt{t}.G''/\mathtt{t}\}\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle\mathscr{C}';G'\rangle$, $\mathtt{p} \in \mathrm{roles}^{\natural}(G')$, and inductive hypothesis, we have $\mathtt{p} \in \mathrm{roles}^{\natural}(G''\{\mu\mathtt{t}.G''/\mathtt{t}\})$. Therefore, by $\mathrm{roles}^{\natural}(\mu\mathtt{t}.G'') = \mathrm{roles}^{\natural}(G''\{\mu\mathtt{t}.G''/\mathtt{t}\})$, we conclude with $\mathtt{p} \in \mathrm{roles}^{\natural}(G)$, as desired.
- Case [GR-&]: we have $G = \mathtt{p}^{\dagger} \rightsquigarrow \mathtt{q}{:}j\,\{\mathtt{m}_{\mathtt{i}}(B_i).G'_i\}_{i\in I}$ and $G' = G'_j$ by [GR-&]. It follows that $\mathrm{roles}^{\natural}(G) = \bigcup_{i\in I}\mathrm{roles}^{\natural}(G'_i)$ and $\mathrm{roles}^{\natural}(G') = \mathrm{roles}^{\natural}(G'_j)$ with $j \in I$, and hence, $\mathrm{roles}^{\natural}(G') \subseteq \mathrm{roles}^{\natural}(G)$. Therefore, by $\mathtt{p} \in \mathrm{roles}^{\natural}(G')$, we conclude with $\mathtt{p} \in \mathrm{roles}^{\natural}(G)$, as desired.
- Case [GR-Ctx-i]: we have $G = \mathtt{p} \rightarrow \mathtt{q}^{\dagger}{:}\{\mathtt{m}_{\mathtt{i}}(B_i).G'_i\}_{i\in I}$, $G' = \mathtt{p} \rightarrow \mathtt{q}^{\dagger}{:}\{\mathtt{m}_{\mathtt{i}}(B_i).G''_i\}_{i\in I}$, $\forall i \in I : \langle\mathscr{C};G'_i\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle\mathscr{C}';G''_i\rangle$, and $\mathrm{subj}(\alpha) \notin \{\mathtt{p},\mathtt{q}\}$ by [GR-Ctx-i] and its inversion. It follows that $\mathrm{roles}^{\natural}(G) = \bigcup_{i\in I}\mathrm{roles}^{\natural}(G'_i)$, $\mathrm{roles}^{\natural}(G') = \bigcup_{i\in I}\mathrm{roles}^{\natural}(G''_i)$, and $\alpha \neq \mathtt{p}\natural$. Then by $\forall i \in I : \langle\mathscr{C};G'_i\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle\mathscr{C}';G''_i\rangle$, $\alpha \neq \mathtt{p}\natural$, and inductive hypothesis, we have $\forall i \in I$: if $\mathtt{p} \in \mathrm{roles}^{\natural}(G''_i)$, then $\mathtt{p} \in \mathrm{roles}^{\natural}(G'_i)$. Therefore, by $\mathtt{p} \in \bigcup_{i\in I}\mathrm{roles}^{\natural}(G''_i)$, we conclude with $\mathtt{p} \in \bigcup_{i\in I}\mathrm{roles}^{\natural}(G'_i) = \mathrm{roles}^{\natural}(G)$, as desired.

Other cases are similar.
(2) Similar to the proof of (1).

(3) The proof is trivial by [GR-↯] and its inversion. □

Lemma 4.16 (Preservation of Well-Annotated Global Types)**.** *If* $\langle \mathscr{C}; G\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, *and* $\langle \mathscr{C}; G\rangle$ *is well-annotated w.r.t.* $\mathscr{R}$, *then* $\langle \mathscr{C}'; G'\rangle$ *is also well-annotated w.r.t.* $\mathscr{R}$.

Proof. By induction on global type reductions:

- Case [GR-↯]: we have $\mathscr{C}' = \mathscr{C} \cup \{\mathtt{p}\}$, $G' = G\!\upharpoonright\!\mathtt{p}$, $\mathtt{p} \notin \mathscr{R}$, $\mathtt{p} \in \mathsf{roles}(G)$, and $G \neq \mu\mathbf{t}.G'$ by [GR-↯] and its inversion.
  Item (WA1): from the premise, we have $\mathsf{roles}^{\lightning}(G) \cap \mathscr{R} = \emptyset$. Since $\mathtt{p} \in \mathsf{roles}(G)$, by Lemma A.2, we have $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\} \subseteq \mathsf{roles}^{\lightning}(G)$. Then we consider two cases:
  – if $\mathtt{p} \in \mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p})$, then $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) = \{\mathtt{p}\} \cup (\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\})$. Hence, by $\mathtt{p} \notin \mathscr{R}$, $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\} \subseteq \mathsf{roles}^{\lightning}(G)$, and $\mathsf{roles}^{\lightning}(G) \cap \mathscr{R} = \emptyset$, we have $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \cap \mathscr{R} = \emptyset$.
  – if $\mathtt{p} \notin \mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p})$, then $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) = \mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\}$. Hence, by $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\} \subseteq \mathsf{roles}^{\lightning}(G)$ and $\mathsf{roles}^{\lightning}(G) \cap \mathscr{R} = \emptyset$, we have $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \cap \mathscr{R} = \emptyset$.
  Therefore, by $G' = G\!\upharpoonright\!\mathtt{p}$, we conclude with $\mathsf{roles}^{\lightning}(G') \cap \mathscr{R} = \emptyset$, as desired.
  Item (WA2): from the premise, we have $\mathsf{roles}^{\lightning}(G) \subseteq \mathscr{C}$. Since $\mathtt{p} \in \mathsf{roles}(G)$, by Lemma A.2, we have $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\} \subseteq \mathsf{roles}^{\lightning}(G)$. Then we consider two cases:
  – if $\mathtt{p} \in \mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p})$, then $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) = \{\mathtt{p}\} \cup (\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\})$. Hence, by $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\} \subseteq \mathsf{roles}^{\lightning}(G)$ and $\mathsf{roles}^{\lightning}(G) \subseteq \mathscr{C}$, we have $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \subseteq \mathscr{C} \cup \{\mathtt{p}\}$.
  – if $\mathtt{p} \notin \mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p})$, then $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) = \mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\}$. Hence, by $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\} \subseteq \mathsf{roles}^{\lightning}(G)$ and $\mathsf{roles}^{\lightning}(G) \subseteq \mathscr{C}$, we have $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \subseteq \mathscr{C} \cup \{\mathtt{p}\}$.
  Therefore, by $G' = G\!\upharpoonright\!\mathtt{p}$ and $\mathscr{C}' = \mathscr{C} \cup \{\mathtt{p}\}$, we conclude with $\mathsf{roles}^{\lightning}(G') \subseteq \mathscr{C}'$, as desired.
  Item (WA3): from the premise, we have $\mathsf{roles}(G) \cap \mathsf{roles}^{\lightning}(G) = \emptyset$. Since $\mathtt{p} \in \mathsf{roles}(G)$, by Lemma A.1, Lemma A.2, we have $\mathsf{roles}(G\!\upharpoonright\!\mathtt{p}) \subseteq \mathsf{roles}(G)$, $\mathtt{p} \notin \mathsf{roles}(G\!\upharpoonright\!\mathtt{p})$, and $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\} \subseteq \mathsf{roles}^{\lightning}(G)$. Then we consider two cases:
  – if $\mathtt{p} \in \mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p})$, then $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) = \{\mathtt{p}\} \cup (\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\})$. Hence, by $\mathsf{roles}(G) \cap \mathsf{roles}^{\lightning}(G) = \emptyset$, $\mathtt{p} \notin \mathsf{roles}(G\!\upharpoonright\!\mathtt{p})$, $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\} \subseteq \mathsf{roles}^{\lightning}(G)$, and $\mathsf{roles}(G\!\upharpoonright\!\mathtt{p}) \subseteq \mathsf{roles}(G)$, we have $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \cap \mathsf{roles}(G\!\upharpoonright\!\mathtt{p}) = \emptyset$.
  – if $\mathtt{p} \notin \mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p})$, then $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) = \mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\}$. Hence, by $\mathsf{roles}(G) \cap \mathsf{roles}^{\lightning}(G) = \emptyset$, $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \setminus \{\mathtt{p}\} \subseteq \mathsf{roles}^{\lightning}(G)$, and $\mathsf{roles}(G\!\upharpoonright\!\mathtt{p}) \subseteq \mathsf{roles}(G)$, we have $\mathsf{roles}^{\lightning}(G\!\upharpoonright\!\mathtt{p}) \cap \mathsf{roles}(G\!\upharpoonright\!\mathtt{p}) = \emptyset$.
  Therefore, by $G' = G\!\upharpoonright\!\mathtt{p}$, we conclude with $\mathsf{roles}(G') \cap \mathsf{roles}^{\lightning}(G') = \emptyset$, as desired.
- Case [GR-μ]: we have $G = \mu\mathbf{t}.G''$ and $\langle \mathscr{C}; G''\{^{\mu\mathbf{t}.G''}/_{\mathbf{t}}\}\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$ by [GR-μ] and its inversion. From the premise, we also have $\langle \mathscr{C}; \mu\mathbf{t}.G''\rangle$ is well-annotated. Hence, by $\mathsf{roles}^{\lightning}(\mu\mathbf{t}.G'') = \mathsf{roles}^{\lightning}(G''\{^{\mu\mathbf{t}.G''}/_{\mathbf{t}}\})$, $\mathsf{roles}(\mu\mathbf{t}.G'') = \mathsf{roles}(G''\{^{\mu\mathbf{t}.G''}/_{\mathbf{t}}\})$, and $\langle \mathscr{C}; \mu\mathbf{t}.G''\rangle$ is well-annotated, we have $\mathsf{roles}^{\lightning}(G''\{^{\mu\mathbf{t}.G''}/_{\mathbf{t}}\}) \cap \mathscr{R} = \emptyset$, $\mathsf{roles}^{\lightning}(G''\{^{\mu\mathbf{t}.G''}/_{\mathbf{t}}\}) \subseteq \mathscr{C}$, and $\mathsf{roles}(G''\{^{\mu\mathbf{t}.G''}/_{\mathbf{t}}\}) \cap \mathsf{roles}^{\lightning}(G''\{^{\mu\mathbf{t}.G''}/_{\mathbf{t}}\}) = \emptyset$. It follows directly that $\langle \mathscr{C}; G''\{^{\mu\mathbf{t}.G''}/_{\mathbf{t}}\}\rangle$ is well-annotated. Therefore, by $\langle \mathscr{C}; G''\{^{\mu\mathbf{t}.G''}/_{\mathbf{t}}\}\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$ and inductive hypothesis, we conclude with $\langle \mathscr{C}'; G'\rangle$ is well-annotated, as desired.

Other cases are similar. □

Lemma A.10. $\langle \mathscr{C}; G\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, *iff* $\langle \mathscr{C}; \mathsf{unf}(G)\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$.

Proof. By inverting or applying [GR-μ] when necessary. □

Lemma A.11 (Progress of Global Types)**.** *If* $\langle \mathscr{C}; G\rangle$ *(where* G *is a projectable global type) is well-annotated, and* $G \neq \mathsf{end}$, *then there exists* $G', \mathscr{C}'$ *such that* $\langle \mathscr{C}; G\rangle \to_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$.

Proof. By Lemma A.10, we only consider unfoldings.

- Case $\mathrm{unf}(G) = \mathsf{end}$: the premise does not hold.
- Case $\mathrm{unf}(G) = \mathtt{p}{\to}\mathtt{q}\colon\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I}$: apply [GR-⊕]. We can pick any $\mathtt{m}_i \neq \mathsf{crash}$ to reduce the global type (note that our syntax prohibits singleton crash branches).
- Case $\mathrm{unf}(G) = \mathtt{p}{\to}\mathtt{q}^{\lightning}\colon\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I}$: apply [GR-↯m]. We can pick any $\mathtt{m}_i \neq \mathsf{crash}$ to reduce the global type (note that our syntax prohibits singleton crash branches).
- Case $\mathrm{unf}(G) = \mathtt{p}^{\dagger}{\rightsquigarrow}\mathtt{q}{:}j\,\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I}$: if $\mathtt{m}_j = \mathsf{crash}$, then it $\mathtt{p}$ must have crashed, apply [GR-⊙]. Otherwise, apply [GR-&]. □

A.4. Semantics of Configurations.

Lemma A.12. *If* $\Gamma;\Delta \to_{\lightning} \Gamma';\Delta'$ *with* $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$, *then*

(1) $\mathrm{dom}(\Gamma) = \mathrm{dom}(\Gamma')$; *and*
(2) *for all* $\mathtt{p} \in \mathrm{dom}(\Gamma)$ *with* $\mathtt{p} \neq \mathrm{subj}(\alpha)$, *we have* $\Gamma(\mathtt{p}) = \Gamma'(\mathtt{p})$.

Proof. Trivial by induction on reductions of configuration. □

Lemma A.13. *If* $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$, *then for any* $\mathtt{p},\mathtt{q} \in \mathrm{dom}(\Gamma)$ *with* $\mathrm{subj}(\alpha) \notin \{\mathtt{p},\mathtt{q}\}$, *we have* $\Delta(\mathtt{p},\mathtt{q}) = \Delta'(\mathtt{p},\mathtt{q})$.

Proof. Trivial by induction on reductions of configuration. □

Lemma A.14 (Inversion of Typing Context Reduction)**.**

(1) *If* $\Gamma;\Delta \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m}_\mathtt{k}(B_k)} \Gamma';\Delta'$, *then* $\mathrm{unf}(\Gamma(\mathtt{p})) = \mathtt{q}\oplus\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I}$, $k \in I$, *and* $\Gamma'(\mathtt{p}) = T_k$;
(2) *If* $\Gamma;\Delta \xrightarrow{\mathtt{q}\&\mathtt{p}:\mathtt{m}_\mathtt{k}(B_k)} \Gamma';\Delta'$, *then* $\mathrm{unf}(\Gamma(\mathtt{q})) = \mathtt{p}\&\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I}$, $k \in I$, *and* $\Gamma'(\mathtt{q}) = T_k$.

Proof. By applying and inverting [Γ-⊕] and [Γ-&]. □

Lemma A.15 (Determinism of Configuration Reduction)**.** *If* $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$ *and* $\Gamma;\Delta \xrightarrow{\alpha} \Gamma'';\Delta''$, *then* $\Gamma' = \Gamma''$ *and* $\Delta' = \Delta''$.

Proof. Trivial by induction on reductions of configuration. □

Lemma A.16. *If* $\Gamma_1;\Delta \xrightarrow{\alpha} \Gamma_1';\Delta_1'$ *and* $\Gamma_2;\Delta \xrightarrow{\alpha} \Gamma_2';\Delta_2'$, *then* $\Delta_1' = \Delta_2'$.

Proof. Trivial by induction on reductions of configuration. □

A.5. Relating Semantics.

Proposition A.17. $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle\mathscr{C};\mu\mathbf{t}.G\rangle$ *if and only if* $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle\mathscr{C};G\{^{\mu\mathbf{t}.G}/_{\mathbf{t}}\}\rangle$.

Proof. By Lemma A.7. □

Proposition A.18. *If* $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle\mathscr{C};G\rangle$ *and* $\Gamma(\mathtt{p}) = \mu\mathbf{t}.T$, *then we have* $\Gamma[\mathtt{p}\mapsto T\{^{\mu\mathbf{t}.T}/_{\mathbf{t}}\}];\Delta \sqsubseteq_{\mathscr{R}} \langle\mathscr{C};G\rangle$.

Proof. By Lemma A.7. □

Proposition A.19. $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle\mathscr{C};G\rangle$ *if and only if* $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle\mathscr{C};\mathrm{unf}(G)\rangle$.

Proof. By applying Proposition A.17 as many times as necessary. □

Lemma A.20 (Inversion of Projection)**.** *Given a local type S, which is a subtype of projection from a global type G on a role $\mathtt{p}$ with respect to a set of reliable roles $\mathscr{R}$, i.e. $S \leqslant (G \upharpoonright_{\mathscr{R}} \mathtt{p})$, then:*

(1) *If* $\mathrm{unf}(S) = \mathtt{q}\oplus\{\mathtt{m_i}(B_i).S'_i\}_{i\in I}$, *then either*
   (a) $\mathrm{unf}(G) = \mathtt{p}\to\mathtt{q}^{\dagger}\colon\{\mathtt{m_i}(B'_i).G_i\}_{i\in I'}$, *where* $I \subseteq I'$, *and for all* $i \in I$: $\mathtt{m}_i = \mathtt{m}_i$, $S'_i \leqslant (G_i \upharpoonright_{\mathscr{R}} \mathtt{p})$, *and* $B_i = B'_i$; *or,*
   (b) $\mathrm{unf}(G) = \mathtt{s}\to\mathtt{t}^{\dagger}\colon\left\{\mathtt{m_j}(B'_j).G_j\right\}_{j\in J}$, *or* $\mathrm{unf}(G) = \mathtt{s}^{\dagger}\rightsquigarrow\mathtt{t}{:}k\left\{\mathtt{m_j}(B'_j).G_j\right\}_{j\in J}$, *where for all* $j \in J$: $S \leqslant (G_j \upharpoonright_{\mathscr{R}} \mathtt{p})$, *with* $\mathtt{p} \neq \mathtt{s}$ *and* $\mathtt{p} \neq \mathtt{t}$.

(2) *If* $\mathrm{unf}(S) = \mathtt{q}\&\{\mathtt{m_i}(B_i).S'_i\}_{i\in I}$, *then either*
   (a) $\mathrm{unf}(G) = \mathtt{q}\to\mathtt{p}^{\dagger}\colon\{\mathtt{m_i}(B'_i).G_i\}_{i\in I'}$, *or* $\mathrm{unf}(G) = \mathtt{q}^{\dagger}\rightsquigarrow\mathtt{p}{:}j\,\{\mathtt{m_i}(B'_i).G_i\}_{i\in I'}$, *where* $I' \subseteq I$, *and for all* $i \in I'$: $\mathtt{m}_i = \mathtt{m}_i$, $S'_i \leqslant (G_i \upharpoonright_{\mathscr{R}} \mathtt{p})$, $B'_i = B_i$, *and* $\mathtt{q} \notin \mathscr{R}$ *implies* $\exists k \in I' : \mathtt{m}_k = \mathsf{crash}$; *or,*
   (b) $\mathrm{unf}(G) = \mathtt{s}\to\mathtt{t}^{\dagger}\colon\left\{\mathtt{m_j}(B'_j).G_j\right\}_{j\in J}$, *or* $\mathrm{unf}(G) = \mathtt{s}^{\dagger}\rightsquigarrow\mathtt{t}{:}k\left\{\mathtt{m_j}(B'_j).G_j\right\}_{j\in J}$, *where for all* $j \in J$: $S \leqslant (G_j \upharpoonright_{\mathscr{R}} \mathtt{p})$, *with* $\mathtt{p} \neq \mathtt{s}$ *and* $\mathtt{p} \neq \mathtt{t}$.

(3) *If* $S = \mathsf{end}$, *then* $\mathtt{p} \notin \mathrm{roles}(G)$.

Proof. By the definition of global type projection (Definition 4.3). □

Lemma A.21 (Existence of Crash Handling Branch Under Projection)**.** *If role* $\mathtt{q}$ *is unreliable,* $\mathtt{q} \notin \mathscr{R}$, *and a global type projects onto* $\mathtt{p}$ *with an external choice from* $\mathtt{q}$, $\mathrm{unf}(G \upharpoonright_{\mathscr{R}} \mathtt{p}) = \mathtt{q}\&\{\mathtt{m_i}(S_i).S'_i\}_{i\in I}$, *then there must be a crash handling branch,* $\exists j \in I : \mathtt{m}_j = \mathsf{crash}$.

Proof. By induction on Item 2 of Lemma A.20. □

Lemma A.22. *If a local type T is a subtype of an external choice with a matching role, obtained via projection from a global type G, i.e., $T = \mathtt{p}\&\{\mathtt{m_i}(B_i).T_i\}_{i\in I} \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{q}$,* $\mathrm{unf}(G)$ *is of the form* $\mathtt{s}\to\mathtt{t}^{\dagger}\colon\left\{\mathtt{m_j}(B'_j).G_j\right\}_{j\in J}$ *or* $\mathtt{s}^{\dagger}\rightsquigarrow\mathtt{t}{:}k\left\{\mathtt{m_j}(B'_j).G_j\right\}_{j\in J}$, *and a queue environment Δ is associated with $\langle\mathscr{C}; G\rangle$, then there exists a global type $\langle\mathscr{C}'; G'\rangle$ and a queue environment Δ' such that $G \upharpoonright_{\mathscr{R}} \mathtt{q} \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{q}$,* $\mathrm{unf}(G')$ *is of the form* $\mathtt{p}^{\dagger}\rightsquigarrow\mathtt{q}{:}j\,\{\mathtt{m_i}(B'_i).G_i\}_{i\in I'}$ *or* $\mathtt{p}\to\mathtt{q}^{\dagger}\colon\{\mathtt{m_i}(B'_i).G_i\}_{i\in I'}$, *and Δ' is associated with $\langle\mathscr{C}'; G'\rangle$ with $\Delta'(\mathtt{p},\mathtt{q}) = \Delta(\mathtt{p},\mathtt{q})$.*

Proof. Apply Item 2 of Lemma A.20 on the premise, we have $\forall j \in J : T = \mathtt{p}\&\{\mathtt{m_i}(B_i).T_i\}_{i\in I} \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{q}$ with $\mathtt{q} \neq \mathtt{s}$ and $\mathtt{q} \neq \mathtt{t}$, which follows that $G \upharpoonright_{\mathscr{R}} \mathtt{q} = \bigsqcap_{j\in J} G_j \upharpoonright_{\mathscr{R}} \mathtt{q}$. Then, by Lemma 4.6, we get $\forall j \in J : G \upharpoonright_{\mathscr{R}} \mathtt{q} \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{q}$. We take an arbitrary G_j with $j \in J$. By applying Item 2 of Lemma A.20 again, we have two cases.

- Case (1): $\mathrm{unf}(G_j) = \mathtt{p}^{\dagger}\rightsquigarrow\mathtt{q}{:}l\,\{\mathtt{m_i}(B'_i).G_i\}_{i\in I'}$ or $\mathrm{unf}(G_j) = \mathtt{p}\to\mathtt{q}^{\dagger}\colon\{\mathtt{m_i}(B'_i).G_i\}_{i\in I'}$.
  Since $G \upharpoonright_{\mathscr{R}} \mathtt{q} \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{q}$, we only need to show that there exists Δ' such that Δ' is associated with $\langle\mathscr{C}; G_j\rangle$ and $\Delta'(\mathtt{p},\mathtt{q}) = \Delta(\mathtt{p},\mathtt{q})$. We consider two subcases:
  – $\mathrm{unf}(G) = \mathtt{s}\to\mathtt{t}^{\dagger}\colon\left\{\mathtt{m_j}(B'_j).G_j\right\}_{j\in J}$: by Definition 4.19, we have that Δ is associated with G_j. We take Δ' as Δ.
  – $\mathrm{unf}(G) = \mathtt{s}^{\dagger}\rightsquigarrow\mathtt{t}{:}k\left\{\mathtt{m_j}(B'_j).G_j\right\}_{j\in J}$, we perform case analysis on $\mathtt{m}_k = \mathsf{crash}$ and $\mathtt{m}_k \neq \mathsf{crash}$:
    * $\mathtt{m}_k = \mathsf{crash}$: by Definition 4.19, we have that Δ is associated with $\langle\mathscr{C}; G_j\rangle$. We take Δ' as Δ.

  - $\mathtt{m}_k \neq \mathsf{crash}$: by Definition 4.19, we have that $\Delta(\mathtt{s},\mathtt{t}) = \mathtt{m}_k(B'_k)\cdot\tau$ and $\Delta[\mathtt{s},\mathtt{t} \mapsto \tau]$ is associated with $\langle \mathscr{C}; G_j\rangle$. We take $\Delta' = \Delta[\mathtt{s},\mathtt{t} \mapsto \tau]$. Since $\mathtt{q} \neq \mathtt{s}$ and $\mathtt{q} \neq \mathtt{t}$, it is straightforward that $\Delta'(\mathtt{p},\mathtt{q}) = \Delta(\mathtt{p},\mathtt{q})$, as required.
- Case (2): $\mathrm{unf}(G_j) = \mathtt{r}{\to}\mathtt{u}^{\dagger}\!:\{\mathtt{m}_\mathtt{l}(B'_l).G_l\}_{l\in L}$ or $\mathtt{r}^{\dagger}{\rightsquigarrow}\mathtt{u}{:}k\,\{\mathtt{m}_\mathtt{l}(B'_l).G_l\}_{l\in L}$.
  We can construct a queue environment Δ' as in case (1), which is associated with $\langle \mathscr{C}; G_j\rangle$. The thesis is then proved by applying inductive hypothesis on $\langle \mathscr{C}; G_j\rangle$ and Δ'. □

Lemma A.23. *If Δ is associated with $\langle \mathscr{C}; G\rangle$ and $\mathtt{p} \in \mathrm{roles}(G)$, then $\Delta[\cdot,\mathtt{p} \mapsto \oslash]$ is associated with $\langle \mathscr{C} \cup \{\mathtt{p}\}; G\!\upharpoonleft\!\mathtt{p}\rangle$.*

Proof. We denote $\Delta' = \Delta[\cdot,\mathtt{p} \mapsto \oslash]$ in the subsequent proof. To show association, there are two parts: namely a shape-dependent part, and a shape-independent part.

We first show shape-independent part, which shared for all cases: that a crashed role $\mathtt{r}$ is in $\mathscr{C} \cup \{\mathtt{p}\}$ iff $\Delta'(\cdot,\mathtt{r}) = \oslash$. It follows that the roles $\mathtt{r} \in \mathscr{C}$ have the requirements satisfied from the premise, and we set $\Delta' = \Delta[\cdot,\mathtt{p} \mapsto \oslash]$, and that $\mathtt{p}$ is in the new set of crashed roles.

Shape-dependent part are by induction on the definition of $G\!\upharpoonleft\!\mathtt{p}$:

- Case $(\mathtt{p}{\to}\mathtt{q}\!:\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I})\!\upharpoonleft\!\mathtt{p} = \mathtt{p}^{\downharpoonleft}{\rightsquigarrow}\mathtt{q}{:}j\,\{\mathtt{m}_\mathtt{i}(B_i).(G_i\!\upharpoonleft\!\mathtt{p})\}_{i\in I}$ where $j \in I$ and $\mathtt{m}_j = \mathsf{crash}$.
  Since Δ is associated with $\langle \mathscr{C}; G\rangle$, we that $\forall i \in I : \Delta$ is associated with $\langle \mathscr{C}; G'_i\rangle$, and $\Delta(\mathtt{p},\mathtt{q}) = \epsilon$.
  By inductive hypothesis, we have Δ' is associated with $\langle \mathscr{C}; (G_i\!\upharpoonleft\!\mathtt{p})\rangle$. Moreover, since $\mathtt{m}_j = \mathsf{crash}$, we can show that $\Delta'(\mathtt{p},\mathtt{q}) = \Delta(\mathtt{p},\mathtt{q}) = \epsilon$.
- Case $(\mathtt{p}{\rightsquigarrow}\mathtt{q}{:}j\,\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I})\!\upharpoonleft\!\mathtt{q} = G_j\!\upharpoonleft\!\mathtt{q}$.
  Subcase $\mathtt{m}_j = \mathsf{crash}$:
  By inductive hypothesis, we know Δ' is associated with $\langle \mathscr{C} \cup \{\mathtt{q}\}; G'_j\rangle$, as required.
  Subcase $\mathtt{m}_j \neq \mathsf{crash}$:
  From association, we have $\Delta(\mathtt{p},\mathtt{q}) = \mathtt{m}_j(B_j)\cdot\tau$ and $\Delta[\mathtt{p},\mathtt{q} \mapsto \tau]$ is associated with $\langle \mathscr{C}; G'_j\rangle$.
  By inductive hypothesis, we know $\Delta[\mathtt{p},\mathtt{q} \mapsto \tau][\cdot,\mathtt{q} \mapsto \oslash]$ is associated with $\langle \mathscr{C} \cup \{\mathtt{q}\}; G'_j\!\upharpoonleft\!\mathtt{q}\rangle$.
  Since $\Delta[\mathtt{p},\mathtt{q} \mapsto \tau][\cdot,\mathtt{q} \mapsto \oslash] = \Delta[\cdot,\mathtt{q} \mapsto \oslash] = \Delta'$, so Δ' is associated with $\langle \mathscr{C} \cup \{\mathtt{q}\}; G'_j\!\upharpoonleft\!\mathtt{q}\rangle$.
- Case $(\mathtt{p}{\to}\mathtt{q}^{\downharpoonleft}\!:\{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i\in I})\!\upharpoonleft\!\mathtt{p} = G_j\!\upharpoonleft\!\mathtt{p}$, where $j \in I$ and $\mathtt{m}_j = \mathsf{crash}$.
  Since Δ is associated with $\langle \mathscr{C}; G\rangle$, we have that Δ is associated with $\langle \mathscr{C}; G_j\rangle$.
  By inductive hypothesis, we have Δ' is associated with $\langle \mathscr{C} \cup \{\mathtt{p}\}; G_j\!\upharpoonleft\!\mathtt{p}\rangle$, as required.
  Other cases follows directly by inductive hypothesis or trivially. □

Lemma A.24. *If $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$, $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$, $\Gamma_1;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G_1\rangle$, $\Gamma(\mathrm{subj}(\alpha)) = \Gamma_1(\mathrm{subj}(\alpha))$, and $\Gamma'_1 = \Gamma_1[\mathrm{subj}(\alpha) \mapsto \Gamma'(\mathrm{subj}(\alpha))]$, then $\Gamma_1;\Delta \xrightarrow{\alpha} \Gamma'_1;\Delta'$.*

Proof. By induction on reductions of configuration.

- Case [Γ-⊕]: $\alpha = \mathtt{p}{\oplus}\mathtt{q} : \mathtt{m}_\mathtt{k}(B_k)$. We apply and invert [Γ-⊕] on $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$ to get $\Gamma(\mathtt{p}) = \mathtt{q}{\oplus}\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I}$, $k \in I$, $\Gamma' = \Gamma[\mathtt{p} \mapsto T_k]$, and $\Delta' = \Delta[\mathtt{p},\mathtt{q} \mapsto \Delta(\mathtt{p},\mathtt{q})\cdot\mathtt{m}_k(B_k)]$. Then by applying [Γ-⊕] on $\Gamma_1(\mathtt{p}) = \Gamma(\mathtt{p}) = \mathtt{q}{\oplus}\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I}$ and $k \in I$, we obtain that $\Gamma_1;\Delta \xrightarrow{\alpha} \Gamma_1[\mathtt{p} \mapsto T_k];\Delta'$, which follows that $\Gamma_1;\Delta \xrightarrow{\alpha} \Gamma'_1;\Delta'$, as desired.
- Case [Γ-&]: $\alpha = \mathtt{p}\&\mathtt{q} : \mathtt{m}_\mathtt{k}(B_k)$. We apply and invert [Γ-&] on $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$ to get $\Gamma(\mathtt{p}) = \mathtt{q}\&\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I}$, $k \in I$, $\Delta(\mathtt{q},\mathtt{p}) = \mathtt{m}_k(B_k)\cdot\tau$, $\Gamma' = \Gamma[\mathtt{p} \mapsto T_k]$, and $\Delta' = \Delta[\mathtt{q},\mathtt{p} \mapsto \tau]$. Then by applying [Γ-&] on $\Gamma_1(\mathtt{p}) = \Gamma(\mathtt{p}) = \mathtt{q}\&\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I}$, $k \in I$, and $\Delta(\mathtt{q},\mathtt{p}) = \mathtt{m}_k(B_k)\cdot\tau$, we obtain that $\Gamma_1;\Delta \xrightarrow{\alpha} \Gamma_1[\mathtt{p} \mapsto T_k];\Delta'$, which follows that $\Gamma_1;\Delta \xrightarrow{\alpha} \Gamma'_1;\Delta'$, as desired.

- Case [Γ-↯]: $\alpha = \mathtt{p}\lightning$. We apply and invert [Γ-↯] on $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$ to get $\Gamma(\mathtt{p}) \neq \mathsf{stop}$, $\Gamma(\mathtt{p}) \neq \mathsf{end}$, $\Gamma' = \Gamma[\mathtt{p} \mapsto \mathsf{stop}]$, and $\Delta' = \Delta[\cdot, \mathtt{p} \mapsto \oslash]$. Then by applying [Γ-↯] on $\Gamma_1(\mathtt{p}) = \Gamma(\mathtt{p}) \neq \mathsf{stop}$ and $\Gamma_1(\mathtt{p}) = \Gamma(\mathtt{p}) \neq \mathsf{end}$, we obtain that $\Gamma_1;\Delta \xrightarrow{\alpha} \Gamma_1[\mathtt{p} \mapsto \mathsf{stop}];\Delta'$, which follows that $\Gamma_1;\Delta \xrightarrow{\alpha} \Gamma_1';\Delta'$, as desired.
- Case [Γ-⊙]: $\alpha = \mathtt{q}\odot\mathtt{p}$. We apply and invert [Γ-⊙] on $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$ to get $\Gamma(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m_i}(B_i).T_i\}_{i\in I}$, $\Gamma(\mathtt{p}) = \mathsf{stop}$, $k \in I$, $\mathtt{m}_k = \mathsf{crash}$, $\Delta(\mathtt{p},\mathtt{q}) = \epsilon$, and $\Gamma' = \Gamma[\mathtt{q} \mapsto T_k]$. Since $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$ and $\Gamma(\mathtt{p}) = \mathsf{stop}$, it holds that $\mathtt{p} \in \mathscr{C}$. Then by $\Gamma_1;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G_1\rangle$, $\Gamma_1(\mathtt{p}) = \mathsf{stop}$. We apply [Γ-⊙] on $\Gamma_1(\mathtt{q}) = \Gamma(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m_i}(B_i).T_i\}_{i\in I}$, $\Gamma_1(\mathtt{p}) = \mathsf{stop}$, $k \in I$, $\mathtt{m}_k = \mathsf{crash}$, and $\Delta(\mathtt{p},\mathtt{q}) = \epsilon$ to obtain $\Gamma_1;\Delta \xrightarrow{\alpha} \Gamma_1[\mathtt{q} \mapsto T_k];\Delta'$, which follows that $\Gamma_1;\Delta \xrightarrow{\alpha} \Gamma_1';\Delta'$, as desired.
- Case [Γ-μ]: by inductive hypothesis. □

We define $I^{\mathtt{m}_i \setminus \mathsf{crash}}$ to be an index set with the special label crash removed, i.e. $I^{\mathtt{m}_i \setminus \mathsf{crash}} = \{i \in I \mid \mathtt{m}_i \neq \mathsf{crash}\}$.

Theorem 4.21 (Soundness of Association)**.** *Given associated global type G and configuration $\Gamma;\Delta$: $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$. If $\langle \mathscr{C}; G\rangle \rightarrow_{\mathscr{R}}$, then there exists $\Gamma';\Delta'$, α and $\langle \mathscr{C}'; G'\rangle$, such that $\langle \mathscr{C}; G\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, $\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, and $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$.*

Proof. By induction on reductions of global type $\langle \mathscr{C}; G\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$.

- Case [GR-↯]:

  From the premise we have

$$\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle \tag{A.1}$$
$$\langle \mathscr{C}; G\rangle \rightarrow_{\mathscr{R}} \tag{A.2}$$
$$\mathtt{p} \notin \mathscr{R} \tag{A.3}$$
$$\mathtt{p} \in \mathrm{roles}(G) \tag{A.4}$$
$$G \neq \mu \mathbf{t}.G' \tag{A.5}$$
$$\alpha = \mathtt{p}\lightning \tag{A.6}$$
$$\mathscr{C}' = \mathscr{C} \cup \{\mathtt{p}\} \tag{A.7}$$
$$G' = G\lightning\mathtt{p} \tag{A.8}$$

  Let $\Gamma';\Delta' = \Gamma[\mathtt{p} \mapsto \mathsf{stop}];\Delta[\cdot, \mathtt{p} \mapsto \oslash]$. We show $\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}''; G''\rangle$ and $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$.

  For the first part: By association (A.1), we know that $\Gamma(\mathtt{p}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{p}$. By $G \neq \mu \mathbf{t}.G'$, $\mathtt{p} \in \mathrm{roles}(G)$, and Lemma A.3, we know that $G \upharpoonright_{\mathscr{R}} \mathtt{p} \neq \mathsf{end}$, which gives $\Gamma(\mathtt{p}) \neq \mathsf{end}$. By $\mathtt{p} \in \mathrm{roles}(G)$, we also know that $\mathtt{p} \notin \mathscr{C}$, which gives $\Gamma(\mathtt{p}) \neq \mathsf{stop}$. Thus we apply [Γ-↯] to obtain the thesis.

  For the second part:

  (A1) From association (A.1), we know that $\forall \mathtt{q} \in \mathrm{roles}(G) : \Gamma(\mathtt{q}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{q}$.
  By Lemma A.1, we know $\mathrm{roles}(G\lightning\mathtt{p}) \subseteq \mathrm{roles}(G)$.
  For any $\mathtt{q} \in \mathrm{roles}(G\lightning\mathtt{p})$, we apply Lemma A.8, to obtain $G \upharpoonright_{\mathscr{R}} \mathtt{q} \leqslant (G\lightning\mathtt{p}) \upharpoonright_{\mathscr{R}} \mathtt{q}$
  Thus we have, by transitivity of subtyping, $\forall \mathtt{q} \in \mathrm{roles}(G\lightning\mathtt{p}) : \Gamma(\mathtt{q}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{q} \leqslant (G\lightning\mathtt{p}) \upharpoonright_{\mathscr{R}} \mathtt{q}$, as required.

  (A2) From association (A.1), we know that $\forall \mathtt{q} \in \mathscr{C} : \Gamma(\mathtt{q}) = \mathsf{stop}$, and they are unchanged in Γ'.

Moreover, we have updated $\Gamma'(\mathtt{p}) = \mathsf{stop}$.

This completes the consideration of the set $\mathscr{C}'$ (A.7).

(A3) No change here.

(A4) By Lemma A.23.

- Case [GR-μ]:

From the premise we have

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \tag{A.9}$$

$$G = \mu \mathbf{t}.G_0 \tag{A.10}$$

$$\langle \mathscr{C}; G \rangle \rightarrow_{\mathscr{R}} \tag{A.11}$$

$$\langle \mathscr{C}; G_0\{^{\mu \mathbf{t}.G_0}/_{\mathbf{t}}\} \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle \tag{A.12}$$

By Proposition A.17, we have $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G_0\{^{\mu \mathbf{t}.G_0}/_{\mathbf{t}}\} \rangle$, and we can apply inductive hypothesis to obtain the desired result.

- Case [GR-$\oplus$]:

From the premise we have

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \tag{A.13}$$

$$G = \mathtt{p} \rightarrow \mathtt{q} \colon \{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i \in I} \tag{A.14}$$

$$\langle \mathscr{C}; G \rangle \rightarrow_{\mathscr{R}} \tag{A.15}$$

$$j \in I \tag{A.16}$$

$$\mathtt{m}_j \neq \mathsf{crash} \tag{A.17}$$

$$\alpha = \mathtt{p} \oplus \mathtt{q} : \mathtt{m}_\mathtt{j}(B_j) \tag{A.18}$$

$$\mathscr{C}' = \mathscr{C} \tag{A.19}$$

$$G' = \mathtt{p} \rightsquigarrow \mathtt{q} \colon j \, \{\mathtt{m}_\mathtt{i}(B_i).G_i\}_{i \in I} \tag{A.20}$$

By association (A.13) and $\mathtt{p} \in \mathrm{roles}(G)$, we know that $\Gamma(\mathtt{p}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{q} \oplus \{\mathtt{m}_\mathtt{i}(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i \in I^{\mathtt{m}_i} \setminus \mathsf{crash}}$. Then by Lemma A.9, we obtain that $\Gamma(\mathtt{p}) = \mathtt{q} \oplus \{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i \in I'}$, where $I' \subseteq I^{\mathtt{m}_i} \setminus \mathsf{crash}$ and $\forall i \in I' : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{p}$. Note that here for any $i \in I' : \mathtt{m}_i = \mathtt{m}_i$.

Since the crash label cannot appear in the internal choices, it holds that for any $i \in I'$, $\mathtt{m}_i \neq \mathsf{crash}$. Therefore, with $\forall i \in I' : \mathtt{m}_i = \mathtt{m}_i$, we can set $j \in I'$ with $\mathtt{m}_j = \mathtt{m}_j \neq \mathsf{crash}$ and $\alpha = \mathtt{p} \oplus \mathtt{q} : \mathtt{m}_\mathtt{j}(B_j) = \mathtt{p} \oplus \mathtt{q} : \mathtt{m}_\mathtt{j}(B_j)$.

Let $\Gamma'; \Delta' = \Gamma[\mathtt{p} \mapsto T_j]; \Delta[\mathtt{p}, \mathtt{q} \mapsto \Delta(\mathtt{p}, \mathtt{q}) \cdot \mathtt{m}_j(B_j)]$. We show that $\Gamma; \Delta \xrightarrow{\alpha} \Gamma'; \Delta'$ and $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle$.

For the first part: we apply [Γ-$\oplus$] on those we have: $\Gamma(\mathtt{p}) = \mathtt{q} \oplus \{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i \in I'}$ and $j \in I'$, to obtain the thesis.

For the second part:

(A1) We want to show $\forall \mathtt{r} \in \mathrm{roles}(G') : \Gamma'(\mathtt{r}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{r}$. We consider three subcases:

- $\mathtt{r} = \mathtt{p}$: since $\Gamma'(\mathtt{p}) = T_j$, $j \in I'$, and $\forall i \in I' : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{p}$, we have $\Gamma'(\mathtt{p}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{p} = G' \upharpoonright_{\mathscr{R}} \mathtt{p}$, as desired.
- $\mathtt{r} = \mathtt{q}$: by association (A.13), we have that $\Gamma'(\mathtt{q}) = \Gamma(\mathtt{q}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{q} = \mathtt{p} \& \{\mathtt{m}_\mathtt{i}(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{q})\}_{i \in I} = G' \upharpoonright_{\mathscr{R}} \mathtt{q}$, as desired.
- $\mathtt{r} \neq \mathtt{q}$ and $\mathtt{r} \neq \mathtt{p}$: by association (A.13), it holds that $\Gamma'(\mathtt{r}) = \Gamma(\mathtt{r}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i \in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{r} = G' \upharpoonright_{\mathscr{R}} \mathtt{r}$, as desired.

(A2) No change here.
(A3) No change here.
(A4) We are left to show that $\Delta' = \Delta[\mathtt{p},\mathtt{q} \mapsto \Delta(\mathtt{p},\mathtt{q}) \cdot \mathtt{m}_j(B_j)]$ is associated with $\langle \mathscr{C}; \mathtt{p} \rightsquigarrow \mathtt{q}:j\,\{\mathtt{m_i}(B_i).G_i\}_{i\in I}\rangle$.
Since Δ is associated with $\langle \mathscr{C}; \mathtt{p} \to \mathtt{q}:\{\mathtt{m_i}(B_i).G_i\}_{i\in I}\rangle$, by Definition 4.19, we have $\Delta(\mathtt{p},\mathtt{q}) = \epsilon$ and $\forall i \in I : \Delta$ is associated with $\langle \mathscr{C}; G_i\rangle$.
Since $\mathtt{m}_j \neq \mathsf{crash}$, we just need to show that $\Delta'(\mathtt{p},\mathtt{q}) = \mathtt{m}_j(B_j)$, which follows directly from $\Delta' = \Delta[\mathtt{p},\mathtt{q} \mapsto \Delta(\mathtt{p},\mathtt{q}) \cdot \mathtt{m}_j(B_j)]$, $\mathtt{m}_j = \mathtt{m}_j$, and $\Delta(\mathtt{p},\mathtt{q}) = \epsilon$, and $\forall i \in I : \Delta'[\mathtt{p},\mathtt{q} \mapsto \epsilon]$ is associated with $\langle \mathscr{C}; G_i\rangle$, which follows from $\Delta'[\mathtt{p},\mathtt{q} \mapsto \epsilon] = \Delta$ and $\forall i \in I : \Delta$ is associated with $\langle \mathscr{C}; G_i\rangle$.

- Case [GR-&]:
From the premise we have

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle \tag{A.21}$$
$$G = \mathtt{p}^\dagger \rightsquigarrow \mathtt{q}:j\,\{\mathtt{m_i}(B_i).G_i\}_{i\in I} \tag{A.22}$$
$$\langle \mathscr{C}; G\rangle \to_{\mathscr{R}} \tag{A.23}$$
$$j \in I \tag{A.24}$$
$$\mathtt{m}_j \neq \mathsf{crash} \tag{A.25}$$
$$\alpha = \mathtt{q}\&\mathtt{p} : \mathtt{m_j}(B_j) \tag{A.26}$$
$$\mathscr{C}' = \mathscr{C} \tag{A.27}$$
$$G' = G_j \tag{A.28}$$

By association (A.21) and $\mathtt{q} \in \mathrm{roles}(G)$, we know that $\Gamma(\mathtt{q}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{q} = \mathtt{p}\&\{\mathtt{m_i}(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{q})\}_{i\in I}$. Note that here for any $i \in I : \mathtt{m}_i = \mathtt{m}_i$. Furthermore, by Lemma A.9, we obtain that $\Gamma(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m_i}(B_i).T_i\}_{i\in I'}$, where $I \subseteq I'$ and $\forall i \in I : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{q}$. From association (A.21), we also get that $\Delta(\mathtt{p},\mathtt{q}) = \mathtt{m}_j(B_j) \cdot \tau = \mathtt{m}_j(B_j) \cdot \tau$.
Let $\Gamma'; \Delta' = \Gamma[\mathtt{q} \mapsto T_j]; \Delta[\mathtt{p},\mathtt{q} \mapsto \tau]$. We show $\Gamma; \Delta \xrightarrow{\alpha} \Gamma'; \Delta'$ and $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$.
For the first part: we apply [Γ-&] on those we have: $\Gamma(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m_i}(B'_i).T_i\}_{i\in I'}$, $j \in I \subseteq I'$, $\mathtt{m}_j = \mathtt{m}_j$, and $\Delta(\mathtt{p},\mathtt{q}) = \mathtt{m}_j(B_j) \cdot \tau$, to obtain the thesis.
For the second part:
(A1) We want to show $\forall \mathtt{r} \in \mathrm{roles}(G_j) : \Gamma'(\mathtt{r}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{r}$. We consider three subcases:
  - $\mathtt{r} = \mathtt{q}$ (meaning that $\mathtt{q} \in \mathrm{roles}(G_j)$): since $\Gamma'(\mathtt{q}) = T_j$ and $\forall i \in I : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{q}$, we have $\Gamma'(\mathtt{q}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{q}$, as desired.
  - $\mathtt{r} = \mathtt{p}$ (meaning that $\mathtt{p}^\dagger = \mathtt{p}$ and $\mathtt{p} \in \mathrm{roles}(G_j)$): by association (A.21), we have $\Gamma'(\mathtt{p}) = \Gamma(\mathtt{p}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{p} = G_j \upharpoonright_{\mathscr{R}} \mathtt{p}$, as desired.
  - $\mathtt{r} \neq \mathtt{q}$ and $\mathtt{r} \neq \mathtt{p}$: by association (A.21), it holds that $\Gamma'(\mathtt{r}) = \Gamma(\mathtt{r}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i\in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$. Then, by applying Lemma 4.6 and transitivity of subtyping, we conclude with $\Gamma'(\mathtt{r}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{r}$, as desired.

(A2) No change here.
(A3) No change here if $\mathtt{q} \in \mathrm{roles}(G_j)$ and $\mathtt{p} \in \mathrm{roles}(G_j)$. Otherwise, if $\mathtt{q} \notin \mathrm{roles}(G_j)$: with $\mathtt{q} \notin \mathrm{roles}^{\not\downarrow}(G_j)$, by Lemma A.4, we have $G_j \upharpoonright_{\mathscr{R}} \mathtt{q} = \mathsf{end}$. Furthermore, by the fact that $\forall i \in I : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{q}$, it holds that $T_j = \mathsf{end}$, and thus, $\Gamma'(\mathtt{q}) = \mathsf{end}$, as desired. The argument for $\mathtt{p} \notin \mathrm{roles}(G_j)$ and $\mathtt{p}^\dagger = \mathtt{p}$ follows similarly.

(A4) Since Δ is associated with $\langle \mathscr{C}; \mathtt{p}^{\dagger} \rightsquigarrow \mathtt{q}{:}j\,\{\mathtt{m_i}(B_i).G_i\}_{i\in I}\rangle$ and $\mathtt{m}_j \neq \mathsf{crash}$, by Definition 4.19, we have that $\Delta(\mathtt{p},\mathtt{q}) = \mathtt{m}_j(B_j)\cdot\tau$ and $\forall i \in I : \Delta[\mathtt{p},\mathtt{q} \mapsto \tau]$ is associated with $\langle \mathscr{C}; G_i\rangle$, which follows that $\Delta' = \Delta[\mathtt{p},\mathtt{q} \mapsto \tau]$ is associated with $\langle \mathscr{C}; G_j\rangle$, as desired.

- Case [GR-$\odot$]:
  From the premise we have

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle \tag{A.29}$$
$$G = \mathtt{p}^{\natural} \rightsquigarrow \mathtt{q}{:}j\,\{\mathtt{m_i}(B_i).G_i\}_{i\in I} \tag{A.30}$$
$$\langle \mathscr{C}; G\rangle \rightarrow_{\mathscr{R}} \tag{A.31}$$
$$j \in I \tag{A.32}$$
$$\mathtt{m}_j = \mathsf{crash} \tag{A.33}$$
$$\alpha = \mathtt{q}\odot\mathtt{p} \tag{A.34}$$
$$\mathscr{C}' = \mathscr{C} \tag{A.35}$$
$$G' = G_j \tag{A.36}$$

By association (A.29) and $\mathtt{q} \in \mathrm{roles}(G)$, we know that $\Gamma(\mathtt{q}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{q} = \mathtt{p}\&\{\mathtt{m_i}(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{q})\}_{i\in I}$. Note that here for any $i \in I : \mathtt{m}_i = \mathtt{m}_i$. Furthermore, by Lemma A.9, we obtain that $\Gamma(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m_i}(B'_i).T_i\}_{i\in I'}$, where $I \subseteq I'$ and $\forall i \in I : B_i = B'_i$ and $T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{q}$.

Let $\Gamma'; \Delta' = \Gamma[\mathtt{q} \mapsto T_j]; \Delta$. We show $\Gamma; \Delta \xrightarrow{\alpha} \Gamma'; \Delta'$ and $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$.

For the first part: By association (A.29), $\mathtt{p} \in \mathscr{C}$, and $\mathtt{m}_j = \mathsf{crash}$ (A.33), we know that $\Gamma(\mathtt{p}) = \mathsf{stop}$ and $\Delta(\mathtt{p},\mathtt{q}) = \epsilon$. Since $j \in I$ (A.32), $\mathtt{m}_j = \mathsf{crash}$ (A.33), and $\forall i \in I : \mathtt{m}_i = \mathtt{m}_i$, we have $\mathtt{m}_j = \mathsf{crash}$. We also get $j \in I'$ from $j \in I$ and $I \subseteq I'$. Thus, together with $\Gamma(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m_i}(B'_i).T_i\}_{i\in I'}$, we apply [$\Gamma$-$\odot$] to obtain the thesis.

For the second part:

(A1) We want to show $\forall \mathtt{r} \in \mathrm{roles}(G_j) : \Gamma'(\mathtt{r}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{r}$. We consider two subcases:
  - $\mathtt{r} = \mathtt{q}$ (meaning that $\mathtt{q} \in \mathrm{roles}(G_j)$): since $\Gamma'(\mathtt{q}) = T_j$, $\forall i \in I : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{q}$, and $j \in I$ (A.32), we have $\Gamma'(\mathtt{q}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{q}$, as desired.
  - $\mathtt{r} \neq \mathtt{q}$: since $\mathtt{p} \in \mathscr{C}$, we know $\mathtt{p} \notin \mathrm{roles}(G_j)$, and thus, $\mathtt{r} \neq \mathtt{p}$. Furthermore, by association (A.29), it holds that $\Gamma'(\mathtt{r}) = \Gamma(\mathtt{r}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i\in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$. Then, by applying Lemma 4.6 and transitivity of subtyping, we can conclude with $\Gamma'(\mathtt{r}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{r}$, as desired.

(A2) No change here.

(A3) No change here if $\mathtt{q} \in \mathrm{roles}(G_j)$. Otherwise, if $\mathtt{q} \notin \mathrm{roles}(G_j)$: with $\mathtt{q} \notin \mathrm{roles}^{\natural}(G_j)$, by Lemma A.4, we have $G_j \upharpoonright_{\mathscr{R}} \mathtt{q} = \mathsf{end}$. Furthermore, by the fact that $\forall i \in I : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{q}$, it holds that $T_j = \mathsf{end}$, and thus, $\Gamma'(\mathtt{q}) = \mathsf{end}$, as desired.

(A4) Since Δ is associated with $\langle \mathscr{C}; \mathtt{p}^{\natural} \rightsquigarrow \mathtt{q}{:}j\,\{\mathtt{m_i}(B_i).G_i\}_{i\in I}\rangle$ and $\mathtt{m}_j = \mathsf{crash}$, by Definition 4.19, we have that $\forall i \in I : \Delta$ is associated with $\langle \mathscr{C}; G_i\rangle$, which follows that $\Delta' = \Delta$ is associated with $\langle \mathscr{C}; G_j\rangle$, as desired.

- Case [GR-$\natural\mathtt{m}$]:

From the premise we have

$$\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle \tag{A.37}$$
$$G = \mathtt{p}{\to}\mathtt{q}^{\not\downarrow} : \{\mathtt{m_i}(B_i).G_i\}_{i\in I} \tag{A.38}$$
$$\langle \mathscr{C}; G\rangle \to_{\mathscr{R}} \tag{A.39}$$
$$j \in I \tag{A.40}$$
$$\mathtt{m}_j \neq \mathsf{crash} \tag{A.41}$$
$$\alpha = \mathtt{p}{\oplus}\mathtt{q} : \mathtt{m_j}(B_j) \tag{A.42}$$
$$\mathscr{C}' = \mathscr{C} \tag{A.43}$$
$$G' = G_j \tag{A.44}$$

By association (A.37) and $\mathtt{p} \in \mathrm{roles}(G)$, we know that $\Gamma(\mathtt{p}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{q}{\oplus}\{\mathtt{m_i}(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i\in I^{\mathtt{m}_i\setminus\mathsf{crash}}}$. Then by Lemma A.9, we obtain that $\Gamma(\mathtt{p}) = \mathtt{q}{\oplus}\{\mathtt{m_i}(B_i).T_i\}_{i\in I'}$, where $I' \subseteq I^{\mathtt{m}_i\setminus\mathsf{crash}}$ and $\forall i \in I' : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{p}$. Note that here for any $i \in I' : \mathtt{m}_i = \mathtt{m}_i$.

Since the crash label cannot appear in the internal choices, it holds that for any $i \in I'$, $\mathtt{m}_i \neq \mathsf{crash}$. Therefore, with $\forall i \in I' : \mathtt{m}_i = \mathtt{m}_i$, we can set $j \in I'$ with $\mathtt{m}_j = \mathtt{m}_j \neq \mathsf{crash}$ and $\alpha = \mathtt{p}{\oplus}\mathtt{q} : \mathtt{m_j}(B_j) = \mathtt{p}{\oplus}\mathtt{q} : \mathtt{m_j}(B_j)$.

Let $\Gamma';\Delta' = \Gamma[\mathtt{p} \mapsto T_j];\Delta[\mathtt{p},\mathtt{q} \mapsto \Delta(\mathtt{p},\mathtt{q})\cdot\mathtt{m}_j(B_j)]$. We show that $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$ and $\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$.

For the first part: we apply [Γ-⊕] on those we have: $\Gamma(\mathtt{p}) = \mathtt{q}{\oplus}\{\mathtt{m_i}(B_i).T_i\}_{i\in I'}$ and $j \in I'$, to obtain the thesis.

For the second part:

(A1) We want to show $\forall \mathtt{r} \in \mathrm{roles}(G') : \Gamma'(\mathtt{r}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{r}$. We consider two subcases:
- $\mathtt{r} = \mathtt{p}$ (meaning that $\mathtt{p} \in \mathrm{roles}(G_j)$): since $\Gamma'(\mathtt{p}) = T_j$, $\forall i \in I' : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{p}$, and $j \in I'$, we have $\Gamma'(\mathtt{p}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{p}$, as desired.
- $\mathtt{r} \neq \mathtt{p}$: since $\mathtt{q} \in \mathscr{C}$, we know $\mathtt{q} \notin \mathrm{roles}(G_j)$, and thus, $\mathtt{r} \neq \mathtt{q}$. Furthermore, by association (A.37), it holds that $\Gamma'(\mathtt{r}) = \Gamma(\mathtt{r}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i\in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$. Then, by applying Lemma 4.6 and transitivity of subtyping, we can conclude with $\Gamma'(\mathtt{r}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{r}$, as desired.

(A2) No change here.

(A3) No change here if $\mathtt{p} \in \mathrm{roles}(G_j)$. Otherwise, if $\mathtt{p} \notin \mathrm{roles}(G_j)$: with $\mathtt{p} \notin \mathrm{roles}^{\not\downarrow}(G_j)$, by Lemma A.4, we have $G_j \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathsf{end}$. Furthermore, by the fact that $\forall i \in I' : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{p}$, it holds that $T_j = \mathsf{end}$, and thus, $\Gamma'(\mathtt{p}) = \mathsf{end}$, as desired.

(A4) We are left to show $\Delta' = \Delta[\mathtt{p},\mathtt{q} \mapsto \Delta(\mathtt{p},\mathtt{q})\cdot\mathtt{m}_j(B_j)]$ is associated with $\langle \mathscr{C}; G_j\rangle$. Since Δ is associated with $\langle \mathscr{C}; \mathtt{p}{\to}\mathtt{q}^{\not\downarrow} : \{\mathtt{m_i}(B_i).G_i\}_{i\in I}\rangle$, by Definition 4.19, we have $\Delta(\mathtt{p},\mathtt{q}) = \oslash$ and $\forall i \in I : \Delta$ is associated with $\langle \mathscr{C}; G_i\rangle$. It follows from $\Delta(\mathtt{p},\mathtt{q}) = \oslash$ that $\Delta' = \Delta[\mathtt{p},\mathtt{q} \mapsto \Delta(\mathtt{p},\mathtt{q})\cdot\mathtt{m}_j(B_j)] = \Delta[\mathtt{p},\mathtt{q} \mapsto \oslash\cdot\mathtt{m}_j(B_j)] = \Delta[\mathtt{p},\mathtt{q} \mapsto \oslash] = \Delta$. Hence, with $\forall i \in I : \Delta$ is associated with $\langle \mathscr{C}; G_i\rangle$, we conclude that $\Delta' = \Delta$ is associated with $\langle \mathscr{C}; G_j\rangle$, as desired.

- Case [GR-Ctx-I]:

From the premise we have

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \tag{A.45}$$

$$G = \mathtt{p} \to \mathtt{q}^{\dagger} \colon \{\mathtt{m_i}(B_i).G_i\}_{i \in I} \tag{A.46}$$

$$\langle \mathscr{C}; G \rangle \to_{\mathscr{R}} \tag{A.47}$$

$$\forall i \in I : \langle \mathscr{C}; G_i \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}'; G_i' \rangle \tag{A.48}$$

$$\mathrm{subj}(\alpha) \notin \{\mathtt{p}, \mathtt{q}\} \tag{A.49}$$

$$G' = \mathtt{p} \to \mathtt{q}^{\dagger} \colon \{\mathtt{m_i}(B_i).G_i'\}_{i \in I} \tag{A.50}$$

We consider two subcases depending on whether $\mathtt{q}$ has crashed.

– $\mathtt{q}^{\dagger} = \mathtt{q}$:

For $\mathtt{p}$, we have that $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{q} \oplus \{\mathtt{m_i}(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i \in I^{\mathtt{m_i} \setminus \mathsf{crash}}}$ and that $G' \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{q} \oplus \{\mathtt{m_i}(B_i).(G_i' \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i \in I^{\mathtt{m_i} \setminus \mathsf{crash}}}$. For $\mathtt{q}$, we have $G \upharpoonright_{\mathscr{R}} \mathtt{q} = \mathtt{p} \,\&\, \{\mathtt{m_i}(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{q})\}_{i \in I}$ and $G' \upharpoonright_{\mathscr{R}} \mathtt{q} = \mathtt{p} \,\&\, \{\mathtt{m_i}(B_i).(G_i' \upharpoonright_{\mathscr{R}} \mathtt{q})\}_{i \in I}$. Take an arbitrary $j \in I$. Let $\Gamma_j(\mathtt{p}) = G_j \upharpoonright_{\mathscr{R}} \mathtt{p}$, $\Gamma_j(\mathtt{q}) = G_j \upharpoonright_{\mathscr{R}} \mathtt{q}$, $\Gamma_j(\mathtt{r}) = \Gamma(\mathtt{r})$ for $\mathtt{r} \in \mathrm{dom}(\Gamma) \setminus \{\mathtt{p}, \mathtt{q}\}$, $\Delta_j = \Delta$. We show $\Gamma_j; \Delta_j \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G_j \rangle$.

(A1) We want to show $\forall \mathtt{s} \in \mathrm{roles}(G_j) : \Gamma_j(\mathtt{s}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{s}$. We consider three subcases:
- $\mathtt{s} = \mathtt{p}$ (meaning that $\mathtt{p} \in \mathrm{roles}(G_j)$): trivial by $\Gamma_j(\mathtt{p}) = G_j \upharpoonright_{\mathscr{R}} \mathtt{p}$ and the reflexivity of subtyping.
- $\mathtt{s} = \mathtt{q}$ (meaning that $\mathtt{q} \in \mathrm{roles}(G_j)$): trivial by $\Gamma_j(\mathtt{q}) = G_j \upharpoonright_{\mathscr{R}} \mathtt{q}$ and the reflexivity of subtyping.
- $\mathtt{s} \neq \mathtt{p}$ and $\mathtt{s} \neq \mathtt{q}$: by association (A.45) and $\Gamma_j(\mathtt{s}) = \Gamma(\mathtt{s})$, we have $\Gamma_j(\mathtt{s}) \leqslant \bigsqcap_{i \in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{s}$. Then, by Lemma 4.6 and transitivity of subtyping, we conclude $\Gamma_j(\mathtt{s}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{s}$, as desired.

(A2) No change here.

(A3) No change here if $\mathtt{p} \in \mathrm{roles}(G_j)$ and $\mathtt{q} \in \mathrm{roles}(G_j)$. Otherwise, consider the case that $\mathtt{p} \notin \mathrm{roles}(G_j)$: with $\mathtt{p} \notin \mathrm{roles}^{\natural}(G_j)$, by Lemma A.4, we have $G_j \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathsf{end}$. Therefore, we know from $\Gamma_j(\mathtt{p}) = G_j \upharpoonright_{\mathscr{R}} \mathtt{p}$ that $\Gamma_j(\mathtt{p}) = \mathsf{end}$, as required. The argument for $\mathtt{q} \notin \mathrm{roles}(G_j)$ follows similarly.

(A4) Trivial by association (A.45), Definition 4.19, and $\Delta_j = \Delta$.

By inductive hypothesis, there exists $\Gamma_j'; \Delta_j'$ such that $\Gamma_j; \Delta_j \xrightarrow{\alpha} \Gamma_j'; \Delta_j'$ and $\Gamma_j'; \Delta_j' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G_j' \rangle$. Since $\mathrm{subj}(\alpha) \notin \{\mathtt{p}, \mathtt{q}\}$, we apply Lemma A.12, which gives $\Gamma_j(\mathtt{p}) = \Gamma_j'(\mathtt{p})$ and $\Gamma_j(\mathtt{q}) = \Gamma_j'(\mathtt{q})$.

We now construct a configuration $\Gamma'; \Delta'$ and show $\Gamma; \Delta \xrightarrow{\alpha} \Gamma'; \Delta'$ and $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle$. Let $\Gamma'(\mathtt{p}) = \Gamma(\mathtt{p})$, $\Gamma'(\mathtt{q}) = \Gamma(\mathtt{q})$, $\Gamma'(\mathtt{r}) = \bigsqcap_{i \in I} \Gamma_i'(\mathtt{r})$ for $\mathtt{r} \in \mathrm{dom}(\Gamma) \setminus \{\mathtt{p}, \mathtt{q}\}$, $\Delta' = \Delta_j'$ with an arbitrary $j \in I$.

For the first part that $\Gamma; \Delta \xrightarrow{\alpha} \Gamma'; \Delta'$:

We know that for any $i \in I$, Γ_i' is obtained from Γ_i by updating $\mathrm{subj}(\alpha)$ to a fixed type T. It follows that for any $i, k \in I$ and for any $\mathtt{r} \notin \{\mathtt{p}, \mathtt{q}\}$, $\Gamma_i'(\mathtt{r}) = \Gamma_k'(\mathtt{r})$, and hence, $\bigsqcap_{i \in I} \Gamma_i'(\mathtt{r}) = \Gamma_j'(\mathtt{r})$ with an arbitrary $j \in J$. Therefore, we have that Γ' is obtained from Γ by updating $\mathrm{subj}(\alpha)$ to $\Gamma_j'(\mathrm{subj}(\alpha))$, i.e., $\Gamma' = \Gamma[\mathrm{subj}(\alpha) \mapsto \Gamma_j'(\mathrm{subj}(\alpha))]$.

We apply Lemma A.24 on $\Gamma_j; \Delta_j \xrightarrow{\alpha} \Gamma_j'; \Delta_j'$, $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$, $\Gamma_j; \Delta_j \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G_j \rangle$, $\Gamma(\mathrm{subj}(\alpha)) = \Gamma_j(\mathrm{subj}(\alpha))$, and $\Gamma' = \Gamma[\mathrm{subj}(\alpha) \mapsto \Gamma_j'(\mathrm{subj}(\alpha))]$ to get the thesis.

For the second part that $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle$:

(A1) We want to show $\forall \mathtt{r} \in \mathrm{roles}(G') : \Gamma'(\mathtt{r}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{r}$. We consider three subcases:

* $\mathtt{r} = \mathtt{p}$: by association (A.45) and Lemma A.9, we obtain $\Gamma(\mathtt{p}) = \mathtt{q}\oplus\{\mathtt{m_i}(B_i).T_i\}_{i\in I'}$ where $I' \subseteq I^{\mathtt{m}_i\setminus\mathsf{crash}}$ and $\forall i \in I' : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{p}$. Since $\Gamma_i(\mathtt{p}) = G_i \upharpoonright_{\mathscr{R}} \mathtt{p} = \Gamma'_i(\mathtt{p})$ for each $i \in I$, with association $\Gamma'_i; \Delta'_i \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'_i\rangle$, we get $G_i \upharpoonright_{\mathscr{R}} \mathtt{p} = \Gamma'_i(\mathtt{p}) \leqslant G'_i \upharpoonright_{\mathscr{R}} \mathtt{p}$. Then by transitivity of subtyping, it holds that $\forall i \in I' : T_i \leqslant G'_i \upharpoonright_{\mathscr{R}} \mathtt{p}$, which follows that $\mathtt{q}\oplus\{\mathtt{m_i}(B_i).T_i\}_{i\in I'} \leqslant \mathtt{q}\oplus\{\mathtt{m_i}(B_i).(G'_i \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i\in I^{\mathtt{m}_i}\setminus\mathsf{crash}}$, as desired.
* $\mathtt{r} = \mathtt{q}$: by association (A.45) and Lemma A.9, we obtain $\Gamma(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m_i}(B_i).T_i\}_{i\in I'}$ where $I \subseteq I'$ and $\forall i \in I : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{q}$. Since $\Gamma_i(\mathtt{q}) = G_i \upharpoonright_{\mathscr{R}} \mathtt{q} = \Gamma'_i(\mathtt{q})$ for each $i \in I$, with association $\Gamma'_i; \Delta'_i \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'_i\rangle$, we get $G_i \upharpoonright_{\mathscr{R}} \mathtt{q} = \Gamma'_i(\mathtt{q}) \leqslant G'_i \upharpoonright_{\mathscr{R}} \mathtt{q}$. Then by transitivity of subtyping, it holds that $\forall i \in I : T_i \leqslant G'_i \upharpoonright_{\mathscr{R}} \mathtt{q}$, which follows that $\mathtt{p}\&\{\mathtt{m_i}(B_i).T_i\}_{i\in I'} \leqslant \mathtt{p}\&\{\mathtt{m_i}(B_i).(G'_i \upharpoonright_{\mathscr{R}} \mathtt{q})\}_{i\in I}$, as desired.
* $\mathtt{r} \neq \mathtt{p}$ and $\mathtt{r} \neq \mathtt{q}$: by association $\Gamma'_i; \Delta'_i \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'_i\rangle$ for each $i \in I$, it holds that $\Gamma'_i(\mathtt{r}) \leqslant G'_i \upharpoonright_{\mathscr{R}} \mathtt{r}$ for each $i \in I$. Then by applying Lemma 4.8, we conclude with $\Gamma'(\mathtt{r}) = \bigsqcap_{i\in I} \Gamma'_i(\mathtt{r}) \leqslant \bigsqcap_{i\in I} G'_i \upharpoonright_{\mathscr{R}} \mathtt{r} = G' \upharpoonright_{\mathscr{R}} \mathtt{r}$, as desired.

(A2) By association $\Gamma'_i; \Delta'_i \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'_i\rangle$ for each $i \in I$, it holds that $\forall \mathtt{r} \in \mathscr{C}' : \Gamma'_i(\mathtt{r}) = \mathsf{stop}$, which follows that for any $\mathtt{r} \in \mathscr{C}'$, $\Gamma'(\mathtt{r}) = \bigsqcap_{i\in I} \Gamma'_i(\mathtt{r}) = \mathsf{stop}$, as desired.

(A3) For any endpoint $\mathtt{r}$ in G', $\mathtt{r}$ is an endpoint in each G'_i with $i \in I$. Then by association $\Gamma'_i; \Delta'_i \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'_i\rangle$, we have $\Gamma'_i(\mathtt{r}) = \mathsf{end}$, which follows $\Gamma'(\mathtt{r}) = \bigsqcap_{i\in I} \Gamma'_i(\mathtt{r}) = \mathsf{end}$, as desired.

(A4) By Lemma A.16, it holds that for any $i, j \in I$, $\Delta'_i = \Delta'_j$. Take an arbitrary Δ'_j with $j \in I$. Note here $\Delta' = \Delta'_j$. With the fact that Δ'_i is associated with G'_i for any $i \in I$, we have that $\forall i \in I : \Delta'_j$ is associated with G'_i, and hence, $\forall i \in I : \Delta'$ is associated with G'_i. We are left to show that $\Delta'(\mathtt{p}, \mathtt{q}) = \epsilon$, which is obtained by applying Lemma A.13 on $\Delta(\mathtt{p}, \mathtt{q}) = \epsilon$ and $\Gamma_j; \Delta \xrightarrow{\alpha} \Gamma'_j; \Delta'_j$ with $\mathrm{subj}(\alpha) \notin \{\mathtt{p}, \mathtt{q}\}$.

– $\mathtt{q}^\dagger = \mathtt{q}^\natural$:

Note that $\mathtt{q} \in \mathscr{C}$. For $\mathtt{p}$, we have $G \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{q}\oplus\{\mathtt{m_i}(B_i).(G_i \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i\in I^{\mathtt{m}_i}\setminus\mathsf{crash}}$ and $G' \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathtt{q}\oplus\{\mathtt{m_i}(B_i).(G'_i \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i\in I^{\mathtt{m}_i}\setminus\mathsf{crash}}$. Take an arbitrary $j \in I$. Let $\Gamma_j(\mathtt{p}) = G_j \upharpoonright_{\mathscr{R}} \mathtt{p}$, $\Gamma_j(\mathtt{q}) = \mathsf{stop}$, $\Gamma_j(\mathtt{r}) = \Gamma(\mathtt{r})$ for $\mathtt{r} \in \mathrm{dom}(\Gamma) \setminus \{\mathtt{p}, \mathtt{q}\}$, $\Delta_j = \Delta$. We show $\Gamma_j; \Delta_j \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G_j\rangle$.

(A1) We want to show $\forall \mathtt{s} \in \mathrm{roles}(G_j) : \Gamma_j(\mathtt{s}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{s}$. We consider two subcases:
* $\mathtt{s} = \mathtt{p}$ (meaning that $\mathtt{p} \in \mathrm{roles}(G_j)$): trivial by $\Gamma_j(\mathtt{p}) = G_j \upharpoonright_{\mathscr{R}} \mathtt{p}$ and the reflexivity of subtyping.
* $\mathtt{s} \neq \mathtt{p}$: since $\mathtt{q} \in \mathscr{C}$, we have $\mathtt{s} \neq \mathtt{q}$. By association (A.45) and $\Gamma_j(\mathtt{s}) = \Gamma(\mathtt{s})$, we have $\Gamma_j(\mathtt{s}) \leqslant \bigsqcap_{i\in I} G_i \upharpoonright_{\mathscr{R}} \mathtt{s}$. Then, by Lemma 4.6 and transitivity of subtyping, we conclude $\Gamma_j(\mathtt{s}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{s}$, as desired.

(A2) No change here.

(A3) No change here if $\mathtt{p} \in \mathrm{roles}(G_j)$. Otherwise, consider the case that $\mathtt{p} \notin \mathrm{roles}(G_j)$: with $\mathtt{p} \notin \mathrm{roles}^\natural(G_j)$, by Lemma A.4, we have $G_j \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathsf{end}$. Therefore, we know from $\Gamma_j(\mathtt{p}) = G_j \upharpoonright_{\mathscr{R}} \mathtt{p}$ that $\Gamma_j(\mathtt{p}) = \mathsf{end}$, as required.

(A4) Trivial by association (A.45), Definition 4.19, and $\Delta_j = \Delta$.

By inductive hypothesis, there exists $\Gamma'_j; \Delta'_j$ such that $\Gamma_j; \Delta_j \xrightarrow{\alpha} \Gamma'_j; \Delta'_j$ and $\Gamma'_j; \Delta'_j \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'_j\rangle$. Since $\mathrm{subj}(\alpha) \notin \{\mathtt{p}, \mathtt{q}\}$, we apply Lemma A.12, which gives $\Gamma_j(\mathtt{p}) = \Gamma'_j(\mathtt{p})$ and $\Gamma_j(\mathtt{q}) = \Gamma'_j(\mathtt{q}) = \mathsf{stop}$. We know from $\Gamma'_j(\mathtt{q}) = \mathsf{stop}$ and $\Gamma'_j; \Delta'_j \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'_j\rangle$ that $\mathtt{q} \in \mathscr{C}'$.

We now construct a configuration $\Gamma'; \Delta'$ and show $\Gamma; \Delta \xrightarrow{\alpha} \Gamma'; \Delta'$ and $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$.

Let $\Gamma'(\mathtt{p}) = \Gamma(\mathtt{p})$, $\Gamma'(\mathtt{q}) = \Gamma(\mathtt{q}) = \mathsf{stop}$, $\Gamma'(\mathtt{r}) = \bigsqcap_{i\in I}\Gamma'_i(\mathtt{r})$ for $\mathtt{r} \in \mathrm{dom}(\Gamma) \setminus \{\mathtt{p},\mathtt{q}\}$, $\Delta' = \Delta'_j$ with an arbitrary $j \in I$.

For the first part that $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$:
We know that for any $i \in I$, Γ'_i is obtained from Γ_i by updating $\mathrm{subj}(\alpha)$ to a fixed type T. It follows that for any $i,k \in I$ and for any $\mathtt{r} \notin \{\mathtt{p},\mathtt{q}\}$, $\Gamma'_i(\mathtt{r}) = \Gamma'_k(\mathtt{r})$, and hence, $\bigsqcap_{i\in I}\Gamma'_i(\mathtt{r}) = \Gamma'_j(\mathtt{r})$ with an arbitrary $j \in J$. Therefore, we have that Γ' is obtained from Γ by updating $\mathrm{subj}(\alpha)$ to $\Gamma'_j(\mathrm{subj}(\alpha))$, i.e., $\Gamma' = \Gamma[\mathrm{subj}(\alpha) \mapsto \Gamma'_j(\mathrm{subj}(\alpha))]$.

We apply Lemma A.24 on $\Gamma_j;\Delta_j \xrightarrow{\alpha} \Gamma'_j;\Delta'_j$, $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle\mathscr{C};G\rangle$, $\Gamma_j;\Delta_j \sqsubseteq_{\mathscr{R}} \langle\mathscr{C};G_j\rangle$, $\Gamma(\mathrm{subj}(\alpha)) = \Gamma_j(\mathrm{subj}(\alpha))$, and $\Gamma' = \Gamma[\mathrm{subj}(\alpha) \mapsto \Gamma'_j(\mathrm{subj}(\alpha))]$ to get the thesis.

For the second part that $\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle\mathscr{C}';G'\rangle$:

(A1) We want to show $\forall \mathtt{r} \in \mathrm{roles}(G') : \Gamma'(\mathtt{r}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{r}$. We consider two subcases:
- $\mathtt{r} = \mathtt{p}$: by association (A.45) and Lemma A.9, we obtain $\Gamma(\mathtt{p}) = \mathtt{q}\oplus\{\mathtt{m}_i(B_i).T_i\}_{i\in I'}$ where $I' \subseteq I^{\mathtt{m}_i\setminus\mathsf{crash}}$ and $\forall i \in I' : T_i \leqslant G_i \upharpoonright_{\mathscr{R}} \mathtt{p}$. Since $\Gamma_i(\mathtt{p}) = G_i \upharpoonright_{\mathscr{R}} \mathtt{p} = \Gamma'_i(\mathtt{p})$ for each $i \in I$, with association $\Gamma'_i;\Delta'_i \sqsubseteq_{\mathscr{R}} \langle\mathscr{C}';G'_i\rangle$, we get $G_i \upharpoonright_{\mathscr{R}} \mathtt{p} = \Gamma'_i(\mathtt{p}) \leqslant G'_i \upharpoonright_{\mathscr{R}} \mathtt{p}$. Then by transitivity of subtyping, it holds that $\forall i \in I' : T_i \leqslant G'_i \upharpoonright_{\mathscr{R}} \mathtt{p}$, which follows that $\mathtt{q}\oplus\{\mathtt{m}_i(B_i).T_i\}_{i\in I'} \leqslant \mathtt{q}\oplus\{\mathtt{m}_i(B_i).(G'_i \upharpoonright_{\mathscr{R}} \mathtt{p})\}_{i\in I^{\mathtt{m}_i\setminus\mathsf{crash}}}$, as desired.
- $\mathtt{r} \neq \mathtt{p}$: since $\mathtt{q} \in \mathscr{C}'$, we have $\mathtt{r} \neq \mathtt{q}$. By association $\Gamma'_i;\Delta'_i \sqsubseteq_{\mathscr{R}} \langle\mathscr{C}';G'_i\rangle$ for each $i \in I$, it holds that $\Gamma'_i(\mathtt{r}) \leqslant G'_i \upharpoonright_{\mathscr{R}} \mathtt{r}$ for each $i \in I$. Then by applying Lemma 4.8, we conclude with $\Gamma'(\mathtt{r}) = \bigsqcap_{i\in I}\Gamma'_i(\mathtt{r}) \leqslant \bigsqcap_{i\in I} G'_i \upharpoonright_{\mathscr{R}} \mathtt{r} = G' \upharpoonright_{\mathscr{R}} \mathtt{r}$, as desired.

(A2) We want to show $\forall \mathtt{r} \in \mathscr{C}' : \Gamma'(\mathtt{r}) = \mathsf{stop}$. We consider two subcases:
- $\mathtt{r} = \mathtt{q}$: trivial by $\Gamma'(\mathtt{q}) = \mathsf{stop}$.
- $\mathtt{r} \neq \mathtt{q}$: by association $\Gamma'_i;\Delta'_i \sqsubseteq_{\mathscr{R}} \langle\mathscr{C}';G'_i\rangle$ for each $i \in I$, it holds that $\forall \mathtt{r} \in \mathscr{C}' : \Gamma'_i(\mathtt{r}) = \mathsf{stop}$, which follows that for any $\mathtt{r} \in \mathscr{C}'$ with $\mathtt{r} \neq \mathtt{q}$, $\Gamma'(\mathtt{r}) = \bigsqcap_{i\in I}\Gamma'_i(\mathtt{r}) = \mathsf{stop}$, as desired.

(A3) For any endpoint $\mathtt{r}$ in G', $\mathtt{r}$ is an endpoint in each G'_i with $i \in I$. Then by association $\Gamma'_i;\Delta'_i \sqsubseteq_{\mathscr{R}} \langle\mathscr{C}';G'_i\rangle$, we have $\Gamma'_i(\mathtt{r}) = \mathsf{end}$, which follows $\Gamma'(\mathtt{r}) = \bigsqcap_{i\in I}\Gamma'_i(\mathtt{r}) = \mathsf{end}$, as desired.

(A4) By Lemma A.16, it holds that for any $i,j \in I$, $\Delta'_i = \Delta'_j$. Take an arbitrary Δ'_j with $j \in I$. Note here $\Delta' = \Delta'_j$. With the fact that Δ'_i is associated with G'_i for any $i \in I$, we have that $\forall i \in I : \Delta'_j$ is associated with G'_i, and hence, $\forall i \in I : \Delta'$ is associated with G'_i. We are left to show that $\Delta'(\mathtt{p},\mathtt{q}) = \oslash$, which is obtained by applying Lemma A.13 on $\Delta(\mathtt{p},\mathtt{q}) = \oslash$ and $\Gamma_j;\Delta \xrightarrow{\alpha} \Gamma'_j;\Delta'_j$ with $\mathrm{subj}(\alpha) \notin \{\mathtt{p},\mathtt{q}\}$.

- Case [GR-Ctx-II]:
  Similar to the case [GR-Ctx-I]. □

Theorem 4.20 (Completeness of Association)**.** *Given associated global type G and configuration $\Gamma;\Delta$: $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle\mathscr{C};G\rangle$. If $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$, where $\alpha \neq \mathtt{p}\not\downarrow$ for all $\mathtt{p} \in \mathscr{R}$, then there exists $\langle\mathscr{C}';G'\rangle$ such that $\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle\mathscr{C}';G'\rangle$ and $\langle\mathscr{C};G\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle\mathscr{C}';G'\rangle$.*

Proof. By induction on reductions of configuration $\Gamma;\Delta \xrightarrow{\alpha} \Gamma';\Delta'$.

- Case [Γ-⊕]:

From the premise, we have:

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \tag{A.51}$$
$$\Gamma(\mathtt{p}) = \mathtt{q} \oplus \{\mathtt{m_i}(B_i).T_i\}_{i \in I} \tag{A.52}$$
$$\alpha = \mathtt{p} \oplus \mathtt{q} : \mathtt{m_k}(B_k) \tag{A.53}$$
$$k \in I \tag{A.54}$$
$$\Gamma' = \Gamma[\mathtt{p} \mapsto T_k] \tag{A.55}$$
$$\Delta' = \Delta[\mathtt{p}, \mathtt{q} \mapsto \Delta(\mathtt{p}, \mathtt{q}) \cdot \mathtt{m}_k(B_k)] \tag{A.56}$$

Apply Lemma A.20 Item 1 on (A.52), we have two cases.

– Case (1):

$$\mathrm{unf}(G) = \mathtt{p} \to \mathtt{q}^{\dagger} : \{\mathtt{m_i}(B'_i).G_i\}_{i \in I'} \tag{A.57}$$
$$I \subseteq I' \tag{A.58}$$
$$\forall i \in I : \mathtt{m}_i = \mathtt{m}_i, T_i \leqslant (G_i \upharpoonright_{\mathscr{R}} \mathtt{p}), B_i = B'_i \tag{A.59}$$

We have two further subcases here: namely $\mathtt{q}^{\dagger} = \mathtt{q}$ and $\mathtt{q}^{\dagger} = \mathtt{q}^{\lightning}$.

∗ In the case of $\mathtt{q}^{\dagger} = \mathtt{q}$, we have $k \in I \subseteq I'$ and crash does not appear in internal choices, we apply [GR-⊕] (via Lemma A.10) to get:

$$\langle \mathscr{C}; G \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}; \mathtt{p} \rightsquigarrow \mathtt{q} : k \{\mathtt{m_i}(B'_i).G_i\}_{i \in I'} \rangle \tag{A.60}$$

We are now left to show $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; \mathtt{p} \rightsquigarrow \mathtt{q} : k \{\mathtt{m_i}(B'_i).G_i\}_{i \in I'} \rangle$.
Note that $G' = \mathtt{p} \rightsquigarrow \mathtt{q} : k \{\mathtt{m_i}(B'_i).G_i\}_{i \in I'}$ here. By Definition 4.1, we have $\mathrm{roles}(G) = \mathrm{roles}(G')$ and $\mathrm{roles}^{\lightning}(G) = \mathrm{roles}^{\lightning}(G')$. Furthermore, with $\mathrm{dom}(\Gamma) = \mathrm{dom}(\Gamma')$ and $\mathtt{p} \in \mathrm{roles}(G)$, we can set $\Gamma' = \Gamma_G[\mathtt{p} \mapsto T_k], \Gamma_{\lightning}, \Gamma_{\mathsf{end}}$.

(A1) First, we want to show that $\mathrm{dom}(\Gamma_G[\mathtt{p} \mapsto T_k]) = \{\mathtt{p} \mid \mathtt{p} \in \mathrm{roles}(G')\}$, which follows directly from the fact that $\mathrm{dom}(\Gamma_G) = \{\mathtt{p} \mid \mathtt{p} \in \mathrm{roles}(G))\}$, $\mathrm{roles}(G) = \mathrm{roles}(G')$, and $\mathrm{dom}(\Gamma_G[\mathtt{p} \mapsto T_k]) = \mathrm{dom}(\Gamma_G)$.
Then, we are left to show $\forall \mathtt{r} \in \mathrm{roles}(G') : \Gamma_G[\mathtt{p} \mapsto T_k](\mathtt{r}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{r}$. We consider two cases:
· $\mathtt{r} = \mathtt{p}$: we have $G' \upharpoonright_{\mathscr{R}} \mathtt{p} = G_k \upharpoonright_{\mathscr{R}} \mathtt{p}$. By (A.59), we obtain that $\Gamma_G[\mathtt{p} \mapsto T_k](\mathtt{p}) = T_k \leqslant G_k \upharpoonright_{\mathscr{R}} \mathtt{p} = G' \upharpoonright_{\mathscr{R}} \mathtt{p}$, as desired.
· $\mathtt{r} \neq \mathtt{p}$: since $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$, we have that $\Gamma_G(\mathtt{r}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i \in I'} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$. Furthermore, by $\Gamma_G[\mathtt{p} \mapsto T_k](\mathtt{r}) = \Gamma_G(\mathtt{r})$ and $G' \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i \in I'} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$, we have that $\Gamma_G[\mathtt{p} \mapsto T_k](\mathtt{r}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{r}$, as desired.

(A2) Trivial by $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$.

(A3) Trivial by $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$.

(A4) By (A.56), we have $\Delta'(\mathtt{p}, \mathtt{q}) = \Delta(\mathtt{p}, \mathtt{q}) \cdot \mathtt{m}_k(B_k)$. Then we only need to show that $\forall i \in I' : \Delta'[\mathtt{p}, \mathtt{q} \mapsto \Delta(\mathtt{p}, \mathtt{q})]$ (note that $\Delta'[\mathtt{p}, \mathtt{q} \mapsto \Delta(\mathtt{p}, \mathtt{q})] = \Delta$) is associated with $\langle \mathscr{C}; G_i \rangle$, which follows directly from the fact that Δ is associated with $\langle \mathscr{C}; G \rangle$ and Definition 4.19.

∗ In the case of $\mathtt{q}^{\dagger} = \mathtt{q}^{\lightning}$, we have $k \in I \subseteq I'$, $\mathtt{q} \in \mathscr{C}$, and crash does not appear in internal choices. We apply [GR-$\lightning\mathtt{m}$] (via Lemma A.10) to get:

$$\langle \mathscr{C}; G \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}; G_k \rangle \tag{A.61}$$

We are now left to show $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G_k \rangle$.
Note that $G' = G_k$ here.

(A1) We need to show that $\forall \mathtt{r} \in \mathrm{roles}(G_k) : \Gamma'(\mathtt{r}) \leqslant G_k \upharpoonright_{\mathscr{R}} \mathtt{r}$. We consider two subcases:
· $\mathtt{r} = \mathtt{p}$, which means that $\mathtt{p} \in \mathrm{roles}(G_k)$: by (A.55) and (A.59), we have $\Gamma'(\mathtt{p}) = T_k \leqslant G_k \upharpoonright_{\mathscr{R}} \mathtt{p}$, as desired.
· $\mathtt{r} \neq \mathtt{p}$: with $\mathtt{q} \notin \mathrm{roles}(G_k)$, we have $\mathtt{r} \neq \mathtt{q}$, and hence, $G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i \in I'} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$. Furthermore, by (A.55) and $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$, it holds that $\Gamma'(\mathtt{r}) = \Gamma(\mathtt{r}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i \in I'} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$. Then, by Lemma 4.6 and transitivity of subtyping, we can conclude that $\Gamma'(\mathtt{r}) \leqslant G_k \upharpoonright_{\mathscr{R}} \mathtt{r}$, as desired.

(A2) Trivial by $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$.

(A3) Trivial by $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$ and the fact that if $\mathtt{p} \notin \mathrm{roles}(G_k)$, then $\Gamma'(\mathtt{p}) = \mathsf{end}$: with $\mathtt{p} \notin \mathrm{roles}^{\lightning}(G_k)$, by Lemma A.4, we have $G_k \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathsf{end}$. Furthermore, by (A.59), it holds that $T_k = \mathsf{end}$, and thus, $\Gamma'(\mathtt{p}) = \mathsf{end}$, as desired.

(A4) Since $\mathtt{q} \in \mathscr{C}$, by Definition 4.19, we have $\Delta(\cdot, \mathtt{q}) = \oslash$. Hence, $\Delta' = \Delta[\mathtt{p}, \mathtt{q} \mapsto \Delta(\mathtt{p}, \mathtt{q}) \cdot \mathtt{m}_k(B_k)] = \Delta[\mathtt{p}, \mathtt{q} \mapsto \oslash \cdot \mathtt{m}_k(B_k)] = \Delta[\mathtt{p}, \mathtt{q} \mapsto \oslash] = \Delta$. Then we only need to show that Δ is associated with $\langle \mathscr{C}; G_k \rangle$, which follows directly from the fact that Δ is associated with $\langle \mathscr{C}; G \rangle$ and Definition 4.19.

– Case (2):

$$\mathrm{unf}(G) = \mathtt{s} \to \mathtt{t}^{\dagger} \colon \left\{ \mathtt{m}_{\mathtt{j}}(B'_j).G_j \right\}_{j \in J} \quad \text{or} \quad \mathrm{unf}(G) = \mathtt{s}^{\dagger} \rightsquigarrow \mathtt{t} \colon k \left\{ \mathtt{m}_{\mathtt{j}}(B'_j).G_j \right\}_{j \in J} \tag{A.62}$$

$$\forall j \in J : \Gamma(\mathtt{p}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{p} \tag{A.63}$$

$$\mathtt{p} \neq \mathtt{s} \ \text{ and } \ \mathtt{p} \neq \mathtt{t} \tag{A.64}$$

We consider two subcases: $\mathrm{unf}(G) = \mathtt{s} \to \mathtt{t}^{\dagger} \colon \left\{ \mathtt{m}_{\mathtt{j}}(B'_j).G_j \right\}_{j \in J}$ and $\mathrm{unf}(G) = \mathtt{s}^{\dagger} \rightsquigarrow \mathtt{t} \colon k \left\{ \mathtt{m}_{\mathtt{j}}(B'_j).G_j \right\}_{j \in J}$.

∗ In the case of $\mathrm{unf}(G) = \mathtt{s} \to \mathtt{t}^{\dagger} \colon \left\{ \mathtt{m}_{\mathtt{j}}(B'_j).G_j \right\}_{j \in J}$:
First, we take an arbitrary index $j \in J$ and construct a configuration $\Gamma_j; \Delta_j$ such that $\Gamma_j; \Delta_j \xrightarrow{\mathtt{p} \oplus \mathtt{q} : \mathtt{m}_k(B_k)} \Gamma'_j; \Delta'_j$ and $\Gamma_j; \Delta_j \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G_j \rangle$.
We know from $\mathtt{s} \in \mathrm{roles}(G)$ and $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$, that $\Gamma(\mathtt{s}) \leqslant \mathrm{unf}(G) \upharpoonright_{\mathscr{R}} \mathtt{s}$ and $\mathrm{unf}(G) \upharpoonright_{\mathscr{R}} \mathtt{s} = \mathtt{t} \oplus \left\{ \mathtt{m}_{\mathtt{j}}(B'_j).(G_j \upharpoonright_{\mathscr{R}} \mathtt{s}) \right\}_{j \in J^{\mathtt{m}_j} \setminus \mathsf{crash}}$. By inverting [Sub-⊕] (applying Lemma A.7 where necessary), we have $\mathrm{unf}(\Gamma(\mathtt{s})) = \mathtt{t} \oplus \left\{ \mathtt{m}_{\mathtt{j}}(B''_j).T''_j \right\}_{j \in J_{\mathtt{s}}}$, where $J_{\mathtt{s}} \subseteq J^{\mathtt{m}_j} \setminus \mathsf{crash}$, and $\forall j \in J_{\mathtt{s}} : T''_j \leqslant (G_j \upharpoonright_{\mathscr{R}} \mathtt{s})$.
To construct Γ_j, let $\Gamma_j(\mathtt{s}) = T''_j$ if $j \in J_{\mathtt{s}}$ and $\Gamma_j(\mathtt{s}) = G_j \upharpoonright_{\mathscr{R}} \mathtt{s}$ otherwise. In either case, we have $\Gamma_j(\mathtt{s}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{s}$, as required.
We have two further subcases here: namely $\mathtt{t}^{\dagger} = \mathtt{t}$ and $\mathtt{t}^{\dagger} = \mathtt{t}^{\lightning}$.
· If $\mathtt{t}^{\dagger} = \mathtt{t}$, we know from $\mathtt{t} \in \mathrm{roles}(G)$ and $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$, that $\Gamma(\mathtt{t}) \leqslant \mathrm{unf}(G) \upharpoonright_{\mathscr{R}} \mathtt{t} = \mathtt{s} \,\&\, \left\{ \mathtt{m}_{\mathtt{j}}(B'_j).(G_j \upharpoonright_{\mathscr{R}} \mathtt{t}) \right\}_{j \in J}$. By inverting [Sub-&] (applying Lemma A.7 where necessary), we have $\mathrm{unf}(\Gamma(\mathtt{t})) = \mathtt{s} \,\&\, \left\{ \mathtt{m}_{\mathtt{j}}(B''_j).U''_j \right\}_{j \in J_{\mathtt{t}}}$, where $J \subseteq J_{\mathtt{t}}$, and $\forall j \in J : U''_j \leqslant (G_j \upharpoonright_{\mathscr{R}} \mathtt{t})$.
To construct Γ_j, let $\Gamma_j(\mathtt{t}) = U''_j$, and we have $\Gamma_j(\mathtt{t}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{t}$, as required.
· If $\mathtt{t}^{\dagger} = \mathtt{t}^{\lightning}$, let $\Gamma_j(\mathtt{t}) = \Gamma(\mathtt{t}) = \mathsf{stop}$, as required.
For roles $\mathtt{r} \in (\mathrm{roles}(G_j) \cup \mathscr{C})$, where $\mathtt{r} \notin \{\mathtt{s}, \mathtt{t}\}$, their typing context entry do not change, i.e. $\Gamma_j(\mathtt{r}) = \Gamma(\mathtt{r})$. For crashed roles $\mathtt{r} \in \mathscr{C}$, we have $\mathsf{stop} = \Gamma(\mathtt{r}) = \Gamma_j(\mathtt{r}) =$

stop, as required. For non-crashed roles $\mathtt{r} \in \mathrm{roles}(G_j)$, we have $\Gamma_j(\mathtt{r}) = \Gamma(\mathtt{r}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{j\in J} (G_j \upharpoonright_{\mathscr{R}} \mathtt{r}) \leqslant G_j \upharpoonright_{\mathscr{R}} \mathtt{r}$ (applying Lemma 4.6).

We know from $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$ and $\mathrm{unf}(G) = \mathtt{s}{\to}\mathtt{t}^{\dagger}\colon \left\{\mathtt{m}_{\mathtt{j}}(B'_j).G_j\right\}_{j\in J}$, that Δ is associated with G_j. Hence, to construct Δ_j, just let $\Delta_j = \Delta$. Notice that $\{\mathtt{p},\mathtt{q}\} \in \mathrm{roles}(G_j)$, so they are still able to perform the communication action $\Gamma_j; \Delta_j \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m}_k(B_k)} \Gamma'_j; \Delta'_j$.

We apply inductive hypothesis on $\Gamma_j; \Delta_j$, and obtain $\langle \mathscr{C}; G_j\rangle \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m}_k(B_k)}_{\mathscr{R}} \langle \mathscr{C}; G'_j\rangle$ and $\Gamma'_j; \Delta'_j \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G'_j\rangle$.

We apply [GR-Ctx-I] (via Lemma A.10) to get:

$$\langle \mathscr{C}; G\rangle \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m}_k(B_k)}_{\mathscr{R}} \langle \mathscr{C}; \mathtt{s}{\to}\mathtt{t}^{\dagger}\colon \left\{\mathtt{m}_{\mathtt{j}}(B'_j).G'_j\right\}_{j\in J}\rangle \tag{A.65}$$

We are now left to show $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; \mathtt{s}{\to}\mathtt{t}^{\dagger}\colon \left\{\mathtt{m}_{\mathtt{j}}(B'_j).G'_j\right\}_{j\in J}\rangle$.

Note that $G' = \mathtt{s}{\to}\mathtt{t}^{\dagger}\colon \left\{\mathtt{m}_{\mathtt{j}}(B'_j).G'_j\right\}_{j\in J}$ here.

(A1) For role $\mathtt{s}$, we know that $\mathrm{unf}(\Gamma(\mathtt{s})) = \mathtt{t}\oplus\left\{\mathtt{m}_{\mathtt{j}}(B''_j).T''_j\right\}_{j\in J_{\mathtt{s}}}$, where $J_{\mathtt{s}} \subseteq J^{\mathtt{m}_j\setminus\mathsf{crash}}$, $\forall j \in J_{\mathtt{s}} : T''_j \leqslant (G_j \upharpoonright_{\mathscr{R}} \mathtt{s})$, and $B''_j = B'_j$.
Since $\mathtt{s} \notin \mathrm{subj}(\mathtt{p}\oplus\mathtt{q} : \mathtt{m}_k(B_k))$, we apply Lemma A.12 on Γ and Γ_j for all $j \in J_{\mathtt{s}}$. For all $j \in J_{\mathtt{s}}$, we have $T''_j = \Gamma_j(\mathtt{s}) = \Gamma'_j(\mathtt{s})$ (from Lemma A.12) and $\Gamma'_j(\mathtt{s}) \leqslant G'_j \upharpoonright_{\mathscr{R}} \mathtt{s}$ (from inductive hypothesis). Therefore, we have $T''_j \leqslant G'_j \upharpoonright_{\mathscr{R}} \mathtt{s}$. We now apply Lemma A.12 on Γ, which gives $\mathrm{unf}(\Gamma(s[\mathtt{s}])) = \mathrm{unf}(\Gamma'(s[\mathtt{s}])) = \mathtt{t}\oplus\left\{\mathtt{m}_{\mathtt{j}}(B''_j).T''_j\right\}_{j\in J_{\mathtt{s}}}$. We can now apply [Sub-⊕] to conclude $\Gamma'(\mathtt{s}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{s}$, as required.
For role $\mathtt{t}$ (where $\mathtt{t}^{\dagger} = \mathtt{t}$), we know that $\mathrm{unf}(\Gamma(\mathtt{t})) = \mathtt{s}\&\left\{\mathtt{m}_{\mathtt{j}}(B''_j).U''_j\right\}_{j\in J_{\mathtt{t}}}$, where $J \subseteq J_{\mathtt{t}}$, $\forall j \in J : U''_j \leqslant (G_j \upharpoonright_{\mathscr{R}} \mathtt{t})$, and $B'_j = B''_j$. Since $\mathtt{t} \notin \mathrm{subj}(\mathtt{p}\oplus\mathtt{q} : \mathtt{m}_k(B_k))$, we apply Lemma A.12 on Γ and Γ_j for all $j \in J$. For all $j \in J$, we have $U''_j = \Gamma_j(\mathtt{t}) = \Gamma'_j(\mathtt{t})$ (from Lemma A.12) and $\Gamma'_j(\mathtt{t}) \leqslant G'_j \upharpoonright_{\mathscr{R}} \mathtt{t}$ (from inductive hypothesis). Therefore, we have $U''_j \leqslant G'_j \upharpoonright_{\mathscr{R}} \mathtt{t}$. We now apply Lemma A.12 on Γ, which gives $\mathrm{unf}(\Gamma(\mathtt{t})) = \mathrm{unf}(\Gamma'(\mathtt{t})) = \mathtt{s}\&\left\{\mathtt{m}_{\mathtt{j}}(B''_j).U''_j\right\}_{j\in J_{\mathtt{t}}}$. We can now apply [Sub-&] to conclude $\Gamma'(\mathtt{t}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{t}$, as required.
For other role $\mathtt{r} \in \mathrm{roles}(G')$ (where $\mathtt{r} \notin \{\mathtt{s},\mathtt{t}\}$), we need to show $\Gamma'(\mathtt{r}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{r}$. We know that $G' \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{j\in J} G'_j \upharpoonright_{\mathscr{R}} \mathtt{r}$. If $\mathtt{r} \notin \{\mathtt{p},\mathtt{q}\}$, we apply Lemma A.12 on Γ_j, obtaining $\Gamma_j(\mathtt{r}) = \Gamma'_j(\mathtt{r})$. The inductive hypothesis gives $\Gamma'_j(\mathtt{r}) \leqslant G'_j \upharpoonright_{\mathscr{R}} \mathtt{r}$, we apply Lemma 4.7 to obtain $\Gamma'_j(\mathtt{r}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{j\in J} G'_j \upharpoonright_{\mathscr{R}} \mathtt{r}$. Note that $\Gamma'_j(\mathtt{r}) = \Gamma_j(\mathtt{r})$ by Lemma A.12, and $\Gamma_j(\mathtt{r}) = \Gamma(\mathtt{r})$ by construction. Therefore, we have $\Gamma'(\mathtt{r}) = \Gamma(\mathtt{r}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{r}$, as required.
We are left to consider the cases of $\mathtt{p}$ and $\mathtt{q}$. We know what $G' \upharpoonright_{\mathscr{R}} \mathtt{p} = \bigsqcap_{j\in J} G'_j \upharpoonright_{\mathscr{R}} \mathtt{p}$ and $\{\mathtt{p},\mathtt{q}\} \subseteq \mathrm{roles}(G')$. The inductive hypothesis gives $\Gamma'_j(\mathtt{p}) \leqslant G'_j \upharpoonright_{\mathscr{R}} \mathtt{p}$, we apply Lemma 4.7 to obtain $\Gamma'_j(\mathtt{p}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{p} = \bigsqcap_{j\in J} G'_j \upharpoonright_{\mathscr{R}} \mathtt{p}$. We now apply Lemma A.15 on Γ and all Γ_j, which gives $\Gamma'_j = \Gamma'$ for all j. Therefore,

we have $\Gamma'(\mathtt{p}) \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{p}$. Note that $\Gamma'(\mathtt{p}) = \Gamma'_j(\mathtt{p}) = T_k$ (as in (A.55)). The argument $\mathtt{q}$ follows similarly.

(A2) For crashed roles $\mathtt{r} \in \mathscr{C}$, we have $\Gamma'(\mathtt{r}) = \Gamma(\mathtt{r}) = \mathsf{stop}$ (applying Lemma A.12). Note that if $\mathtt{t}^\dagger = \mathtt{t}^\natural$, $\mathtt{t} \in \mathscr{C}$.

(A3) Trivial by $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$.

(A4) We know from $\Delta_j = \Delta$ and $\Gamma_j; \Delta_j \xrightarrow{\mathtt{p} \oplus \mathtt{q}:\mathtt{m}_k(B_k)} \Gamma'_j; \Delta'_j$, that $\Delta'_j = \Delta_j[\mathtt{p}, \mathtt{q} \mapsto \Delta_j(\mathtt{p}, \mathtt{q}) \cdot \mathtt{m}_k(B_k)] = \Delta[\mathtt{p}, \mathtt{q} \mapsto \Delta(\mathtt{p}, \mathtt{q}) \cdot \mathtt{m}_k(B_k)] = \Delta'$. Furthermore, by inductive hypothesis, Δ'_j is associated with $\langle \mathscr{C}; G'_j \rangle$, which follows that $\forall j \in J : \Delta'$ is associated with $\langle \mathscr{C}; G'_j \rangle$. We are left to show that if $\mathtt{t}^\dagger \neq \mathtt{t}^\natural$, then $\Delta'(\mathtt{s}, \mathtt{t}) = \epsilon$, which follows directly from $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$ and $\Delta'(\mathtt{s}, \mathtt{t}) = \Delta(\mathtt{s}, \mathtt{t})$.

– In the case of $\mathrm{unf}(G) = \mathtt{s}^\dagger \rightsquigarrow \mathtt{t}:k \left\{ \mathtt{m}_\mathtt{j}(B'_j).G_j \right\}_{j \in J}$:

Similar to that of the previous subcase that $\mathrm{unf}(G) = \mathtt{s} \rightarrow \mathtt{t}^\dagger : \left\{ \mathtt{m}_\mathtt{j}(B'_j).G_j \right\}_{j \in J}$, applying [GR-Ctx-II] instead.

• Case [Γ-&]:

From the premise, we have:

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \tag{A.66}$$
$$\Gamma(\mathtt{p}) = \mathtt{q} \& \{ \mathtt{m}_\mathtt{i}(B_i).T_i \}_{i \in I} \tag{A.67}$$
$$\alpha = \mathtt{p} \& \mathtt{q} : \mathtt{m}_\mathtt{k}(B_k) \tag{A.68}$$
$$k \in I \tag{A.69}$$
$$\Delta(\mathtt{q}, \mathtt{p}) = \mathtt{m}_k(B_k) \cdot \tau' \neq \oslash \tag{A.70}$$
$$\Gamma' = \Gamma[\mathtt{p} \mapsto T_k] \tag{A.71}$$
$$\Delta' = \Delta[\mathtt{q}, \mathtt{p} \mapsto \tau'] \tag{A.72}$$

Apply Lemma A.20 Item 2 on (A.67), we have two cases.

– Case (1):

$$\mathrm{unf}(G) = \mathtt{q}^\dagger \rightsquigarrow \mathtt{p}:j \left\{ \mathtt{m}_\mathtt{i}(B'_i).G_i \right\}_{i \in I'} \text{ or } \mathrm{unf}(G) = \mathtt{q} \rightarrow \mathtt{p}^\dagger : \left\{ \mathtt{m}_\mathtt{i}(B'_i).G_i \right\}_{i \in I'} \tag{A.73}$$
$$I' \subseteq I \tag{A.74}$$
$$\forall i \in I' : \mathtt{m}_i = \mathtt{m}_i, T_i \leqslant (G_i \upharpoonright_{\mathscr{R}} \mathtt{p}), B'_i = B_i \tag{A.75}$$
$$\mathtt{q} \notin \mathscr{R} \text{ implies } \exists l \in I' : \mathtt{m}_l = \mathsf{crash} \tag{A.76}$$

First, we show that in this case, $\mathrm{unf}(G)$ cannot be of the form $\mathtt{q} \rightarrow \mathtt{p}^\dagger : \{ \mathtt{m}_\mathtt{i}(B'_i).G_i \}_{i \in I'}$. There are two subcases to be considered: $\mathtt{p}^\dagger = \mathtt{p}$ and $\mathtt{p}^\dagger = \mathtt{p}^\natural$.

∗ In the case of $\mathtt{p}^\dagger = \mathtt{p}^\natural$, we have $\mathtt{p} \in \mathscr{C}$. Hence, by applying Definition 4.19 on (A.66), we have that $\Delta(\cdot, \mathtt{p}) = \oslash$, a desired contradiction to (A.70).

∗ In the case of $\mathtt{p}^\dagger = \mathtt{p}$, by association, it holds that $\Delta(\mathtt{q}, \mathtt{p}) = \epsilon$, a desired contradiction to (A.70).

Therefore, we only need to consider the case that $\mathrm{unf}(G) = \mathtt{q}^\dagger \rightsquigarrow \mathtt{p}:j \{ \mathtt{m}_\mathtt{i}(B'_i).G_i \}_{i \in I'}$. Then we want to show that $\mathtt{m}_j \neq \mathsf{crash}$, which is proved by contradiction. Assume that $\mathtt{m}_j = \mathsf{crash}$, by association, we have that $\Delta(\mathtt{q}, \mathtt{p}) = \epsilon$, a desired contradiction to (A.70). Moreover, we want to show that $j = k$. By association and $\mathtt{m}_j \neq \mathsf{crash}$, we have $\Delta(\mathtt{q}, \mathtt{p}) = \mathtt{m}_j(B_j) \cdot \tau = \mathtt{m}_k(B_k) \cdot \tau'$. Then by (A.75), it holds that $\mathtt{m}_j = \mathtt{m}_j = \mathtt{m}_k$.

Furthermore, by $j, k \in I$ and the requirement that labels in local types must be pair-wise distinct, we have $j = k$, as required. Note that $k \in I'$ here.
We can now apply [GR-&] (via Lemma A.10) to get:

$$\langle \mathscr{C}; G \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}; G_k \rangle \tag{A.77}$$

We are left to show $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G_k \rangle$.

(A1) We need to show that $\forall \mathtt{r} \in \mathrm{roles}(G_k) : \Gamma'(\mathtt{r}) \leqslant G_k \upharpoonright_{\mathscr{R}} \mathtt{r}$. We consider three subcases:
- $\mathtt{r} = \mathtt{q}$, which means that $\mathtt{q}^\dagger = \mathtt{q}$ and $\mathtt{q} \in \mathrm{roles}(G_k)$: by (A.71), $\Gamma'(\mathtt{q}) = \Gamma(\mathtt{q})$. Then by association, we have $\Gamma(\mathtt{q}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{q} = G_k \upharpoonright_{\mathscr{R}} \mathtt{q}$, as desired.
- $\mathtt{r} = \mathtt{p}$, which means that $\mathtt{p} \in \mathrm{roles}(G_k)$: by (A.71) and (A.75), we have $\Gamma'(\mathtt{p}) = T_k \leqslant G_k \upharpoonright_{\mathscr{R}} \mathtt{p}$, as desired.
- $\mathtt{r} \neq \mathtt{q}$ and $\mathtt{r} \neq \mathtt{p}$: $G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i \in I'} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$. Furthermore, by (A.71) and $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$, it holds that $\Gamma'(\mathtt{r}) = \Gamma(\mathtt{r}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i \in I'} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$. Then, by Lemma 4.6 and transitivity of subtyping, we can conclude that $\Gamma'(\mathtt{r}) \leqslant G_k \upharpoonright_{\mathscr{R}} \mathtt{r}$, as desired.

(A2) Trivial by $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$. Note that in the case of $\mathtt{q}^\dagger = \mathtt{q}^\lightning$, we have $\mathtt{q} \in \mathscr{C}$, and hence, $\Gamma'(\mathtt{q}) = \Gamma(\mathtt{q}) = \mathsf{stop}$.

(A3) Trivial by $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle$ and the fact that if $\mathtt{p} \notin \mathrm{roles}(G_k)$, then $\Gamma'(\mathtt{p}) = \mathsf{end}$: with $\mathtt{p} \notin \mathrm{roles}^\lightning(G_k)$, by Lemma A.4, we have $G_k \upharpoonright_{\mathscr{R}} \mathtt{p} = \mathsf{end}$. Furthermore, by (A.75), it holds that $T_k = \mathsf{end}$, and thus, $\Gamma'(\mathtt{p}) = \mathsf{end}$, as desired. The argument for the case that $\mathtt{q}^\dagger = \mathtt{q}$ and $\mathtt{q} \notin \mathrm{roles}(G_k)$ follows similarly.

(A4) Since Δ is associated with $\langle \mathscr{C}; \mathtt{q}^\dagger \rightsquigarrow \mathtt{p}{:}j \, \{\mathtt{m}_i(B'_i).G_i\}_{i \in I'} \rangle$ and $\mathtt{m}_j \neq \mathsf{crash}$, by Definition 4.19, we have that $\Delta(\mathtt{q}, \mathtt{p}) = \mathtt{m}_j(B_j) \cdot \tau$ and $\forall i \in I' : \Delta[\mathtt{q}, \mathtt{p} \mapsto \tau]$ is associated with $\langle \mathscr{C}; G_i \rangle$, which follows that $\tau = \tau'$ (in (A.70)) and $\Delta' = \Delta[\mathtt{q}, \mathtt{p} \mapsto \tau'] = \Delta[\mathtt{q}, \mathtt{p} \mapsto \tau]$ is associated with $\langle \mathscr{C}; G_k \rangle$, as required.

– Case (2): similar to the case (2) in Case [Γ-⊕].

- Case [Γ-↯]:

From the premise, we have:

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \tag{A.78}$$
$$\Gamma(\mathtt{p}) \neq \mathsf{end} \tag{A.79}$$
$$\Gamma(\mathtt{p}) \neq \mathsf{stop} \tag{A.80}$$
$$\alpha = \mathtt{p}\lightning \tag{A.81}$$
$$\Gamma' = \Gamma[\mathtt{p} \mapsto \mathsf{stop}] \tag{A.82}$$
$$\Delta' = \Delta[\cdot, \mathtt{p} \mapsto \oslash] \tag{A.83}$$

By (A.80), we know that $\mathtt{p} \notin \mathscr{C}$ and $\mathtt{p} \in \mathrm{roles}(G)$. The premise requires that $\alpha \neq \mathtt{p}\lightning$ for all $\mathtt{p} \in \mathscr{R}$, therefore, $\mathtt{p} \notin \mathscr{R}$. We apply [GR-↯] (via Lemma A.10) to get:

$$\langle \mathscr{C}; G \rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C} \cup \{\mathtt{p}\}; G\lightning\mathtt{p} \rangle \tag{A.84}$$

We are now left to show $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C} \cup \{\mathtt{p}\}; G\lightning\mathtt{p} \rangle$.

Note that $G' = G\lightning\mathtt{p}$ and $\mathscr{C}' = \mathscr{C} \cup \{\mathtt{p}\}$ here. By (A.82), we can set $\Gamma' = \Gamma'_{G'}, \Gamma'_\lightning, \Gamma'_{\mathsf{end}}$, where $\mathrm{dom}(\Gamma'_{G'}) = \{\mathtt{q} \mid \mathtt{q} \in \mathrm{roles}(G\lightning\mathtt{p})\} = \{\mathtt{q} \mid \mathtt{q} \in \mathrm{roles}(G)\} \setminus \{\mathtt{p}\} = \mathrm{dom}(\Gamma_G) \setminus \{\mathtt{p}\}$, $\mathrm{dom}\left(\Gamma'_\lightning\right) = \mathscr{C} \cup \{\mathtt{p}\} = \mathrm{dom}(\Gamma_\lightning) \cup \{\mathtt{p}\}$, and $\mathrm{dom}(\Gamma'_{\mathsf{end}}) = \mathrm{dom}(\Gamma_{\mathsf{end}})$. Meanwhile, we have $\Gamma'(\mathtt{q}) = \Gamma(\mathtt{q})$ if $\mathtt{q} \neq \mathtt{p}$.

(A1) We want to show that for any $\mathtt{q} \in \mathrm{roles}(G\!\downarrow\!\mathtt{p})$, we have $\Gamma'(\mathtt{q}) = \Gamma(\mathtt{q}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{q} \leqslant (G\!\downarrow\!\mathtt{p}) \upharpoonright_{\mathscr{R}} \mathtt{q}$ by Lemma A.8 and $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$.
(A2) Trivial by $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$ and (A.82), i.e., $\Gamma'(\mathtt{p}) = \mathsf{stop}$.
(A3) Trivial by $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$.
(A4) Since Δ is associated with $\langle \mathscr{C}; G\rangle$, by Definition 4.19, we have that for any $\mathtt{q} \in \mathscr{C}$, $\Delta(\cdot, \mathtt{q}) = \oslash$. Then, with (A.83), we have that for any $\mathtt{q} \in \mathscr{C} \cup \{\mathtt{p}\}$, $\Delta'(\cdot, \mathtt{q}) = \Delta[\cdot, \mathtt{p} \mapsto \oslash](\cdot, \mathtt{q}) = \oslash$, which follows directly that Δ' is associated with $\langle \mathscr{C} \cup \{\mathtt{p}\}; G\!\downarrow\!\mathtt{p}\rangle$.

- Case [Γ-⊙]:
  From the premise, we have:

$$\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle \tag{A.85}$$
$$\Gamma(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m}_{\mathtt{i}}(B_i).T_i\}_{i\in I} \tag{A.86}$$
$$\Gamma(\mathtt{p}) = \mathsf{stop} \tag{A.87}$$
$$\alpha = \mathtt{q}\odot\mathtt{p} \tag{A.88}$$
$$k \in I \tag{A.89}$$
$$\mathtt{m}_k = \mathsf{crash} \tag{A.90}$$
$$\Delta(\mathtt{p}, \mathtt{q}) = \epsilon \tag{A.91}$$
$$\Gamma' = \Gamma[\mathtt{q} \mapsto T_k] \tag{A.92}$$
$$\Delta' = \Delta \tag{A.93}$$

  Since $\Gamma(\mathtt{p}) = \mathsf{stop}$, we have $\mathtt{p} \in \mathscr{C}$ and $\mathtt{p} \notin \mathscr{R}$. Apply Lemma A.20 Item 2 on (A.86), we have two cases.
  – Case (1):

$$\mathrm{unf}(G) = \mathtt{p}^{\downarrow} \rightsquigarrow \mathtt{q}{:}j\,\{\mathtt{m}_{\mathtt{i}}(B'_i).G_i\}_{i\in I'} \tag{A.94}$$
$$I' \subseteq I \tag{A.95}$$
$$\forall i \in I' : \mathtt{m}_i = \mathtt{m}_i, T_i \leqslant (G_i \upharpoonright_{\mathscr{R}} \mathtt{q}), B'_i = B_i \tag{A.96}$$
$$\mathtt{p} \notin \mathscr{R} \text{ implies } \exists l \in I' : \mathtt{m}_l = \mathsf{crash} \tag{A.97}$$

  Then we want to show that $\mathtt{m}_j = \mathsf{crash}$, which is proved by contradiction. Assume that $\mathtt{m}_j \neq \mathsf{crash}$, by association, we have that $\Delta(\mathtt{p}, \mathtt{q}) = \mathtt{m}_j(B_j)\cdot\tau$, a desired contradiction to (A.91). Moreover, we want to show that $j = k$. By (A.96), we have $\mathtt{m}_j = \mathtt{m}_j$, which means that $\mathtt{m}_j = \mathsf{crash}$. Since $\mathtt{m}_j = \mathtt{m}_k = \mathsf{crash}$ and $j, k \in I$, by the requirement that labels in local types must be pair-wise distinct, we have $j = k$, as required. Note that $k \in I'$ here.
  We can now apply [GR-⊙] (via Lemma A.10) to get:

$$\langle \mathscr{C}; G\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}; G_k\rangle \tag{A.98}$$

  We are left to show $\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G_k\rangle$.
  (A1) We need to show that $\forall \mathtt{r} \in \mathrm{roles}(G_k) : \Gamma'(\mathtt{r}) \leqslant G_k \upharpoonright_{\mathscr{R}} \mathtt{r}$. We consider two subcases:
    * $\mathtt{r} = \mathtt{q}$, which means that $\mathtt{q} \in \mathrm{roles}(G_k)$: by (A.92) and (A.96), we have $\Gamma'(\mathtt{q}) = T_k \leqslant G_k \upharpoonright_{\mathscr{R}} \mathtt{q}$, as desired.
    * $\mathtt{r} \neq \mathtt{q}$: with $\mathtt{p} \notin \mathrm{roles}(G_k)$, we have $\mathtt{r} \neq \mathtt{p}$, and hence, $G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i\in I'} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$. Furthermore, by (A.92) and $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$, it holds that $\Gamma'(\mathtt{r}) = \Gamma(\mathtt{r}) \leqslant G \upharpoonright_{\mathscr{R}} \mathtt{r} = \bigsqcap_{i\in I'} G_i \upharpoonright_{\mathscr{R}} \mathtt{r}$. Then, by Lemma 4.6 and transitivity of subtyping, we can conclude that $\Gamma'(\mathtt{r}) \leqslant G_k \upharpoonright_{\mathscr{R}} \mathtt{r}$, as desired.

(A2) Trivial by $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$.
(A3) Trivial by $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$ and the fact that if $\mathtt{q} \notin \mathrm{roles}(G_k)$, then $\Gamma'(\mathtt{q}) = \mathsf{end}$: with $\mathtt{q} \notin \mathrm{roles}^{\lightning}(G_k)$, by Lemma A.4, we have $G_k \upharpoonright_{\mathscr{R}} \mathtt{q} = \mathsf{end}$. Furthermore, by (A.96), it holds that $T_k = \mathsf{end}$, and thus, $\Gamma'(\mathtt{q}) = \mathsf{end}$, as desired.
(A4) Since Δ is associated with $\langle \mathscr{C}; \mathtt{p}^{\lightning} \rightsquigarrow \mathtt{q}{:}j\, \{\mathtt{m_i}(B'_i).G_i\}_{i\in I'}\rangle$ and $\mathtt{m}_j = \mathsf{crash}$, by Definition 4.19, we have that $\forall i \in I' : \Delta$ is associated with $\langle \mathscr{C}; G_i\rangle$, which follows that $\Delta' = \Delta$ is associated with $\langle \mathscr{C}; G_k\rangle$, as required.

– Case (2): similar to the case (2) in Case [Γ-⊕].

- Case [Γ-μ]:
  By induction hypothesis and Proposition A.19. □

A.6. **Safety by Projection.**

Lemma A.25. *If* $\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$, *then* $\Gamma;\Delta$ *is* $\mathscr{R}$*-safe.*

Proof. Let $\varphi = \left\{\Gamma';\Delta' \,\middle|\, \exists \mathscr{C}', G' : \langle \mathscr{C}; G\rangle \to^*_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle \text{ and } \Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle\right\}$. Take any $\Gamma';\Delta' \in \varphi$, we show that $\Gamma';\Delta'$ satisfies all clauses in Definition 4.24, which means that φ is an $\mathscr{R}$-safety property. Then we can conclude that, since $\varphi(\Gamma;\Delta)$ holds, $\Gamma;\Delta$ is $\mathscr{R}$-safe.

By definition of φ, there exists $\langle \mathscr{C}'; G'\rangle$ with $\langle \mathscr{C}; G\rangle \to^*_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$ and $\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$.

- [S-⊕&]: from the premise, we have

$$\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle \tag{A.99}$$
$$\Gamma'(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m_i}(B_i).T_i\}_{i\in I} \tag{A.100}$$
$$\Delta'(\mathtt{p},\mathtt{q}) \neq \oslash \tag{A.101}$$
$$\Delta'(\mathtt{p},\mathtt{q}) \neq \epsilon \tag{A.102}$$

Apply Item 2 of Lemma A.20 on (A.100), we have two cases.

– Case (1):

$$\mathrm{unf}(G') = \mathtt{p}^{\dagger} \rightsquigarrow \mathtt{q}{:}j\, \{\mathtt{m_i}(B'_i).G_i\}_{i\in I'} \text{ or } \mathrm{unf}(G') = \mathtt{p} \to \mathtt{q}^{\dagger}{:}\, \{\mathtt{m_i}(B'_i).G_i\}_{i\in I'} \tag{A.103}$$
$$I' \subseteq I \tag{A.104}$$
$$\forall i \in I' : \mathtt{m}_i = \mathtt{m}_i, T_i \leqslant (G_i \upharpoonright_{\mathscr{R}} \mathtt{q}), B'_i = B_i \tag{A.105}$$

First, we show that $\mathrm{unf}(G')$ cannot be the form of $\mathtt{p} \to \mathtt{q}^{\dagger}{:}\, \{\mathtt{m_i}(B'_i).G_i\}_{i\in I'}$. We prove this by contradiction on two subcases: $\mathtt{q}^{\dagger} = \mathtt{q}$ and $\mathtt{q}^{\dagger} = \mathtt{q}^{\lightning}$:

* $\mathtt{q}^{\dagger} = \mathtt{q}$: by association (A.99), we have $\Delta'(\mathtt{p},\mathtt{q}) = \epsilon$, a desired contradiction to (A.102).
* $\mathtt{q}^{\dagger} = \mathtt{q}^{\lightning}$: we have $\mathtt{q} \in \mathscr{C}$. Hence, by association (A.99), we have $\Delta'(\cdot,\mathtt{q}) = \oslash$, a desired contradiction to (A.101).

It follows that $\mathrm{unf}(G') = \mathtt{p}^{\dagger} \rightsquigarrow \mathtt{q}{:}j\, \{\mathtt{m_i}(B'_i).G_i\}_{i\in I'}$. We are left to show that $j \in I$, $B'_j = B_j$, and $\Delta'(\mathtt{p},\mathtt{q}) = \mathtt{m}_j(B'_j)\cdot\tau$, and then by applying [Γ-&], we can conclude with $\Gamma';\Delta' \xrightarrow{\mathtt{q}\&\mathtt{p}:\mathtt{m_j}(B_j)} \to$. We consider two subcases: $\mathtt{m}_j = \mathsf{crash}$ and $\mathtt{m}_j \neq \mathsf{crash}$:

* $\mathtt{m}_j = \mathsf{crash}$: by association, $\Delta'(\mathtt{p},\mathtt{q}) = \epsilon$, a desired contradiction to (A.102).
* $\mathtt{m}_j \neq \mathsf{crash}$: by association, $\Delta'(\mathtt{p},\mathtt{q}) = \mathtt{m}_j(B'_j)\cdot\tau$. Moreover, with (A.104) and (A.105), we obtain that $j \in I$, $B'_j = B_j$, and $\mathtt{m}_j = \mathtt{m}_j$, which follows that $\Delta'(\mathtt{p},\mathtt{q}) = \mathtt{m}_j(B'_j)\cdot\tau$, as desired.

– Case (2): $\mathrm{unf}(G') = \mathtt{s}{\to}\mathtt{t}^{\dagger}\colon \left\{\mathtt{m}_\mathtt{j}(B'_j).G_j\right\}_{j\in J}$, or $\mathrm{unf}(G') = \mathtt{s}^{\dagger}{\rightsquigarrow}\mathtt{t}\colon k\left\{\mathtt{m}_\mathtt{j}(B'_j).G_j\right\}_{j\in J}$, where for all $j \in J$: $\Gamma'(\mathtt{q}) \leqslant (G_j \upharpoonright_{\mathscr{R}} \mathtt{q})$, with $\mathtt{q} \neq \mathtt{s}$ and $\mathtt{q} \neq \mathtt{t}$.
We apply Lemma A.22 to get that there exists a global type $\langle \mathscr{C}'', G''\rangle$ and a queue environment Δ'' such that $\Gamma'(\mathtt{q}) \leqslant G'' \upharpoonright_{\mathscr{R}} \mathtt{q}$, $\mathrm{unf}(G'')$ is of the form $\mathtt{p}^{\dagger}{\rightsquigarrow}\mathtt{q}\colon j\,\{\mathtt{m}_\mathtt{i}(B'_i).G_i\}_{i\in I'}$ or $\mathtt{p}{\to}\mathtt{q}^{\dagger}\colon\{\mathtt{m}_\mathtt{i}(B'_i).G_i\}_{i\in I'}$, and Δ'' is associated with $\langle \mathscr{C}'', G''\rangle$ with $\Delta''(\mathtt{p},\mathtt{q}) = \Delta'(\mathtt{p},\mathtt{q})$.
The following proof argument is similar to that for Case (1).

- [S-↯&]: from the premise, we have $\Gamma'(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I}$, $\Gamma'(\mathtt{p}) = \mathsf{stop}$, and $\Delta'(\mathtt{p},\mathtt{q}) = \epsilon$. We just need to show that there exists $k \in I$ with $\mathtt{m}_k = \mathsf{crash}$, and then by applying [Γ-⊙], we can conclude with $\Gamma'; \Delta' \xrightarrow{\mathtt{q}\odot\mathtt{p}}$. By the association that $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, we have $\Gamma'(\mathtt{q}) = \mathtt{p}\&\{\mathtt{m}_\mathtt{i}(B_i).T_i\}_{i\in I} \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{q}$. Moreover, with $\mathtt{p} \notin \mathscr{R}$, which follows directly from $\Gamma'(\mathtt{p}) = \mathsf{stop}$, we can apply Lemma A.21 to get $\exists k \in I : \mathtt{m}_k = \mathsf{crash}$, as desired.
- [S-μ]: let $\Gamma''; \Delta''$ be constructed from $\Gamma'; \Delta'$ with $\Gamma'' = \Gamma'[\mathtt{p} \mapsto S\{{}^{\mu\mathbf{t}.S}/_{\mathbf{t}}\}]$ and $\Delta'' = \Delta'$. We want to show that $\Gamma''; \Delta'' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$. From the premise, we have $\Gamma'(\mathtt{p}) = \mu\mathbf{t}.S$. Then by association that $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, it holds that $\Gamma'(\mathtt{p}) = \mu\mathbf{t}.S \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{p}$. Hence, by inverting [Sub-μL], we have $\Gamma''(\mathtt{p}) = S\{{}^{\mu\mathbf{t}.S}/_{\mathbf{t}}\} \leqslant G' \upharpoonright_{\mathscr{R}} \mathtt{p}$. Therefore, by Definition 4.19, we conclude with $\Gamma''; \Delta'' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, as desired.
- [S-→↯]: from the premise, we have $\Gamma'; \Delta' \to_{\mathscr{R}} \Gamma''; \Delta''$. Hence, by Theorem 4.20, there exists $\langle \mathscr{C}''; G''\rangle$ with $\langle \mathscr{C}'; G'\rangle \xrightarrow{\alpha}_{\mathscr{R}} \langle \mathscr{C}''; G''\rangle$, where $\alpha \neq \mathtt{p}\not\downarrow$ for all $\mathtt{p} \in \mathscr{R}$, and $\Gamma''; \Delta'' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}''; G''\rangle$. By definition of φ, the configuration after transition $\Gamma''; \Delta''$ is in φ, as desired. □

A.7. **Deadlock Freedom by Projection.**

Lemma A.26. *If* $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$, *then* $\Gamma; \Delta$ *is* $\mathscr{R}$*-deadlock-free.*

Proof. Since $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$, by Lemma A.25, we have that $\Gamma; \Delta$ is $\mathscr{R}$-safe. By operational correspondence of global type $\langle \mathscr{C}; G\rangle$ and configuration $\Gamma; \Delta$ (Theorem 4.20 and Theorem 4.21), there exists a global type $\langle \mathscr{C}'; G'\rangle$ such that $\langle \mathscr{C}; G\rangle \to^*_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle \not\to_{\mathscr{R}}$, with associated configurations $\Gamma; \Delta \to^*_{\mathscr{R}} \Gamma'; \Delta' \not\to_{\mathscr{R}}$. Since no further reductions are possible for the global type, the global type $\langle \mathscr{C}'; G'\rangle$ must be of the form $\langle \mathscr{C}'; \mathsf{end}\rangle$. By applying Definition 4.19 on $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; \mathsf{end}\rangle$, we have $\Gamma' = \Gamma'_{\mathsf{end}}, \Gamma'_{\not\downarrow}$, where $\mathrm{dom}(\Gamma'_{\mathsf{end}}) = \{\mathtt{p} \mid \Gamma'(\mathtt{p}) = \mathsf{end}\}$ and $\mathrm{dom}\left(\Gamma'_{\not\downarrow}\right) = \mathscr{C}' = \{\mathtt{p} \mid \Gamma'(\mathtt{p}) = \mathsf{stop}\}$, as required. By association again, we have that, for any $\mathtt{p}, \mathtt{q}$, if $\mathtt{q} \in \mathscr{C}'(= \mathrm{dom}\left(\Gamma'_{\not\downarrow}\right))$, $\Delta'(\cdot, \mathtt{q}) = \oslash$, and otherwise, $\Delta'(\mathtt{p},\mathtt{q}) = \epsilon$, as required. □

A.8. **Liveness by Projection.**

Lemma A.27. *If* $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$, *then* $\Gamma; \Delta$ *is* $\mathscr{R}$*-live.*

Proof. Since $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$, by Lemma A.25, we have that $\Gamma; \Delta$ is $\mathscr{R}$-safe.

We are left to show that if $\Gamma; \Delta \to^*_{\mathscr{R}} \Gamma'; \Delta'$, then any non-crashing path starting with $\Gamma'; \Delta'$ which is fair is also live. By operational correspondence of global type $\langle \mathscr{C}; G\rangle$ and configuration $\Gamma; \Delta$ (Theorem 4.20 and Theorem 4.21), there exists a global type $\langle \mathscr{C}'; G'\rangle$ such that $\langle \mathscr{C}; G\rangle \to^* \langle \mathscr{C}'; G'\rangle$ and $\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$. We construct a non-crashing fair path $(\Gamma'_n; \Delta'_n)_{n\in N}$, where $N = \{0, 1, 2, \ldots\}$, $\Gamma'_0 = \Gamma'$, $\Delta'_0 = \Delta'$, and $\forall n \in N$, $\Gamma'_n; \Delta'_n \to \Gamma'_{n+1}; \Delta'_{n+1}$. Then we just need to show that $(\Gamma'_n; \Delta'_n)_{n\in N}$ is live.

(L1) Suppose that $\Delta'_n(\mathtt{p},\mathtt{q}) = \mathtt{m}(B)\cdot\tau \neq \oslash$ and $\mathtt{m} \neq \mathsf{crash}$. By operational correspondence of $\langle \mathscr{C}'; G'\rangle$ and configuration $\Gamma'_0;\Delta'_0$, and $\forall n \in N$, $\Gamma'_n;\Delta'_n \to \Gamma'_{n+1};\Delta'_{n+1}$, there exists $\langle \mathscr{C}'_{\mathtt{n}}; G'_n\rangle$ such that $\Gamma'_n;\Delta'_n \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'_{\mathtt{n}}; G'_n\rangle$. By Lemma A.22 and Definition 4.19, we only need to consider the case that $G'_n = \mathtt{p}^{\dagger} \rightsquigarrow \mathtt{q}{:}j\,\{\mathtt{m}_{\mathtt{i}}(B_i).G''_i\}_{i\in I}$ with $\mathtt{m} = \mathtt{m}_j \neq \mathsf{crash}$ and $B_j = B$. By applying [GR-&], $\langle \mathscr{C}'_{\mathtt{n}}; G'_n\rangle \xrightarrow{\mathtt{q}\&\mathtt{p}:\mathtt{m}_j(B_j)}_{\mathscr{R}} \langle \mathscr{C}'_{\mathtt{n}}; G''_j\rangle$. Hence, by the soundness of association, it holds that $\Gamma'_n;\Delta'_n \xrightarrow{\mathtt{q}\&\mathtt{p}:\mathtt{m}(B)}$. Finally, combining the fact that $(\Gamma'_n;\Delta'_n)_{n\in N}$ is fair with $\Gamma'_n;\Delta'_n \xrightarrow{\mathtt{q}\&\mathtt{p}:\mathtt{m}(B)}$, we can conclude that there exists k such that $n \leq k \in N$ and $\Gamma'_k;\Delta'_k \xrightarrow{\mathtt{q}\&\mathtt{p}:\mathtt{m}(B)} \Gamma'_{k+1};\Delta'_{k+1}$, as desired.

(L2) Suppose that $\Gamma'_n(\mathtt{p}) = \mathtt{q}\&\{\mathtt{m}_{\mathtt{i}}(B_i).T_i\}_{i\in I}$. By operational correspondence of $\langle \mathscr{C}'; G'\rangle$ and configuration $\Gamma'_0;\Delta'_0$, and $\forall n \in N$, $\Gamma'_n;\Delta'_n \to \Gamma'_{n+1};\Delta'_{n+1}$, there exists $\langle \mathscr{C}'_{\mathtt{n}}; G'_n\rangle$ such that $\Gamma'_n;\Delta'_n \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'_{\mathtt{n}}; G'_n\rangle$. By Definition 4.19, Lemma A.20, and Lemma A.22, we only have to consider that G'_n is of the form $\mathtt{q}{\to}\mathtt{p}^{\dagger}{:}\,\{\mathtt{m}_{\mathtt{i}}(B'_i).G''_i\}_{i\in I'}$ or $\mathtt{q}^{\dagger}\rightsquigarrow\mathtt{p}{:}j\,\{\mathtt{m}_{\mathtt{i}}(B'_i).G''_i\}_{i\in I'}$, where $I' \subseteq I$, and for all $i \in I'$: $\mathtt{m}_i = \mathtt{m}_i$, $B'_i = B_i$, and $\mathtt{q} \notin \mathscr{R}$ implies $\exists k \in I' : \mathtt{m}_k = \mathsf{crash}$.

- $G'_n = \mathtt{q}{\to}\mathtt{p}^{\dagger}{:}\,\{\mathtt{m}_{\mathtt{i}}(B'_i).G''_i\}_{i\in I'}$: we first show that $\mathtt{p}^{\dagger} = \mathtt{p}$. Since $\Gamma'_n(\mathtt{p}) \neq \mathsf{stop}$, by Definition 4.19, we have that $\mathtt{p} \notin \mathscr{C}'_{\mathtt{n}}$, and hence, $\mathtt{p}^{\dagger} \neq \mathtt{p}^{\not\downarrow}$. Given that $G'_n = \mathtt{q}{\to}\mathtt{p}{:}\,\{\mathtt{m}_{\mathtt{i}}(B'_i).G''_i\}_{i\in I'}$, by association, Definition 4.19, and inversion of subtyping, $\Gamma'_n(\mathtt{q})$ is of the form $\mathtt{p}{\oplus}\{\mathtt{m}_{\mathtt{i}}(B'_i).T'_i\}_{i\in I''}$ where $I'' \subseteq I'$. Then applying [Γ-⊕], $\Gamma'_n;\Delta'_n \xrightarrow{\mathtt{q}\oplus\mathtt{p}:\mathtt{m}_j(B'_j)}$ for some $j \in I''$. Then together with the fairness of $(\Gamma'_n;\Delta'_n)_{n\in N}$, we have that there exists $k, \mathtt{m}', B''$ such that $n \leq k \in N$ and $\Gamma'_k;\Delta'_k \xrightarrow{\mathtt{q}\oplus\mathtt{p}:\mathtt{m}'(B'')} \Gamma'_{k+1};\Delta'_{k+1}$, which follows that $\Delta'_{k+1}(\mathtt{q},\mathtt{p}) = \Delta'_k[\mathtt{q},\mathtt{p} \mapsto \Delta'_k(\mathtt{q},\mathtt{p})\cdot\mathtt{m}'(B'')]$. Finally, by the previous case (L1), we can conclude that there exists $k', \mathtt{m}'', B'''$ such that $k+1 \leq k' \in N$ and $\Gamma'_{k'};\Delta'_{k'} \xrightarrow{\mathtt{p}\&\mathtt{q}:\mathtt{m}''(B''')} \Gamma'_{k'+1};\Delta'_{k'+1}$, as desired.
- $G'_n = \mathtt{q}^{\dagger}\rightsquigarrow\mathtt{p}{:}j\,\{\mathtt{m}_{\mathtt{i}}(B'_i).G''_i\}_{i\in I'}$: we consider two subcases:
  - $\mathtt{m}_j \neq \mathsf{crash}$: by applying [GR-&], $\langle \mathscr{C}'_{\mathtt{n}}; G'_n\rangle \xrightarrow{\mathtt{p}\&\mathtt{q}:\mathtt{m}_j(B'_j)}_{\mathscr{R}} \langle \mathscr{C}'_{\mathtt{n}}; G''_j\rangle$. Hence, by the soundness association, we have that $\Gamma'_n;\Delta'_n \xrightarrow{\mathtt{p}\&\mathtt{q}:\mathtt{m}_j(B'_j)}$. Finally, combining the fact that $(\Gamma'_n;\Delta'_n)_{n\in N}$ is fair with $\Gamma'_n;\Delta'_n \xrightarrow{\mathtt{p}\&\mathtt{q}:\mathtt{m}_j(B'_j)}$, we can conclude that there exists k such that $n \leq k \in N$ and $\Gamma'_k;\Delta'_k \xrightarrow{\mathtt{p}\&\mathtt{q}:\mathtt{m}_j(B'_j)} \Gamma'_{k+1};\Delta'_{k+1}$, as desired.
  - $\mathtt{m}_j = \mathsf{crash}$: consider the following two subcases:
    - $\mathtt{q}^{\dagger} = \mathtt{q}^{\not\downarrow}$: $\Gamma'_n(\mathtt{q}) = \mathsf{stop}$ by $\mathtt{q} \in \mathscr{C}'_{\mathtt{n}}$. By Definition 4.19, $\Delta'_n(\mathtt{q},\mathtt{p}) = \epsilon$. Now applying [Γ-⊙] on $\Gamma'_n(\mathtt{p}) = \mathtt{q}\&\{\mathtt{m}_{\mathtt{i}}(B_i).T_i\}_{i\in I}, \Gamma'_n(\mathtt{q}) = \mathsf{stop}, \Delta'_n(\mathtt{q},\mathtt{p}) = \epsilon$ and $\mathtt{m}_j = \mathtt{m}_j = \mathsf{crash}$ with $j \in I' \subseteq I$, we have that $\Gamma'_n;\Delta'_n \xrightarrow{\mathtt{p}\odot\mathtt{q}}$. Then together with the fairness of $(\Gamma'_n;\Delta'_n)_{n\in N}$, we can conclude that there exists k such that $n \leq k \in N$ and $\Gamma'_k;\Delta'_k \xrightarrow{\mathtt{p}\odot\mathtt{q}} \Gamma'_{k+1};\Delta'_{k+1}$.
    - $\mathtt{q}^{\dagger} \neq \mathtt{q}^{\not\downarrow}$: since $\mathtt{m}_j = \mathsf{crash}$, together with $\langle \mathscr{C}; G\rangle \to^*_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, we know that $\mathtt{q}^{\dagger} \neq \mathtt{q}$ holds, a desired contradiction. □

Appendix B. Proofs for Section 5

Lemma B.1 (Typing Inversion). *Let $\Theta \vdash P : T'$. Then, there exists $T \leqslant T'$ such that:*

(1) $P = \mathbf{0}$ *implies* $T = \mathsf{end}$;
(2) $P = \lightning$ *implies* $T = \mathsf{stop}$;
(3) $P = \mathtt{q}!\mathtt{m}\langle e\rangle.P_1$ *implies*
  (a1) $T = \mathtt{q}\oplus\mathtt{m}(B).T_1$, *and*
  (a2) $\Theta \vdash P_1 : T_1$, *and*
  (a3) $\Theta \vdash e : B$;
(4) $P = \sum_{i\in I} \mathtt{q}?\mathtt{m}_i(x_i).P_i$ *implies*
  (a1) $T = \mathtt{q}\&\{\mathtt{m}_i(B_i).T_i\}_{i\in I}$, *and*
  (a2) $\forall i \in I \quad \Theta, x_i : B_i \vdash P_i : T_i$;
(5) $P = \mathsf{if}\, e\, \mathsf{then}\, P_1\, \mathsf{else}\, P_2$ *implies*
  (a1) $\Theta \vdash e : \mathsf{bool}$, *and*
  (a2) $\Theta \vdash P_1 : T$, *and*
  (a3) $\Theta \vdash P_2 : T$;
(6) $P = \mu X.P_1$ *implies* $\Theta, X : T \vdash P_1 : T$;

Let $\vdash h : \delta$. *Then:*

(7) $h = \epsilon$ *implies* $\delta = \epsilon$;
(8) $h = \oslash$ *implies* $\delta = \oslash$;
(9) $h = (\mathtt{q}, \mathtt{m}(v))$ *implies* $\vdash v : B$ *and* $\delta(\mathtt{q}) = \mathtt{m}(B)$;
(10) $h = h_1 \cdot h_2$ *implies* $\vdash h_1 : \delta_1$, *and* $\vdash h_2 : \delta_2$, *and* $\delta = \delta_1 \cdot \delta_2$;

Let $\langle \mathscr{C}; G\rangle \vdash \prod_{i\in I}(\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i)$. *Then:*

(a1) $\exists \Gamma; \Delta$ *such that* $\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle$, *and*
(a2) $\mathrm{dom}(\Gamma) \subseteq \{\mathtt{p}_i \mid i \in I\}$, *and*
(a3) $\forall i \in I : \vdash P_i : \Gamma(\mathtt{p}_i)$, *and*
(a4) $\forall i \in I : \vdash h_i : \Delta(-, \mathtt{p}_i)$.

Proof. By inverting the rules in Fig. 9. □

Lemma B.2 (Typing Precongruence)**.** (1) *If* $\Theta \vdash P : T$ *and* $P \Rrightarrow Q$, *then* $\Theta \vdash Q : T$.
(2) *If* $\vdash h_1 : \delta_1$ *and* $h_1 \Rrightarrow h_2$, *then there exists* δ_2 *such that* $\delta_1 \Rrightarrow \delta_2$ *and* $\vdash h_2 : \delta_2$.
(3) *If* $\langle \mathscr{C}; G\rangle \vdash \mathbb{M}$ *and* $\mathbb{M} \Rrightarrow \mathbb{M}'$, *then* $\langle \mathscr{C}; G\rangle \vdash \mathbb{M}'$.

Proof. (1) By induction on the precongruence rules for $P \Rrightarrow Q$.
(2) By induction on the precongruence rules for $h_1 \Rrightarrow h_2$.
(3) By induction on the precongruence rules for $\mathbb{M} \Rrightarrow \mathbb{M}'$. □

Lemma B.3 (Substitution)**.** *If* $\Theta, x : B \vdash P : T$ *and* $\Theta \vdash v : B$, *then* $\Theta \vdash P\{v/x\} : T$.

Proof. By induction on the structure of P. □

Lemma B.4. *If* $\emptyset \vdash e : B$ *and* $e \downarrow v$, *then* $\emptyset \vdash v : B$.

Proof. By induction on the derivation of $\emptyset \vdash e : B$. □

Lemma B.5. $\langle \mathscr{C}; \mu \mathbf{t}.G\rangle \vdash \mathbb{M}$ *if and only if* $\langle \mathscr{C}; G\{^{\mu \mathbf{t}.G}/_{\mathbf{t}}\}\rangle \vdash \mathbb{M}$.

Proof. The thesis follows directly by applying [T-SESS] and Proposition A.17. □

Theorem 5.2 (Subject Reduction)**.** *If* $\langle \mathscr{C}; G\rangle \vdash \mathbb{M}$ *and* $\mathbb{M} \to_{\mathscr{R}} \mathbb{M}'$, *then either* $\langle \mathscr{C}; G\rangle \vdash \mathbb{M}'$, *or there exists* $\langle \mathscr{C}'; G'\rangle$ *such that* $\langle \mathscr{C}; G\rangle \to_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$ *and* $\langle \mathscr{C}'; G'\rangle \vdash \mathbb{M}'$.

Proof. Let us recap the assumptions:

$$\langle \mathscr{C}; G\rangle \vdash \mathbb{M} \tag{B.1}$$
$$\mathbb{M} \to_{\mathscr{R}} \mathbb{M}' \tag{B.2}$$

The proof proceeds by induction on the derivation of $\mathbb{M} \to_{\mathscr{R}} \mathbb{M}'$. Most cases hold by typing inversion (Lemma B.1), and by applying the induction hypothesis.

- Case [R-SEND]: we have

$$\mathbb{M} = \mathtt{p} \triangleleft \mathtt{q}!\mathtt{m}\langle e\rangle.P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft Q \mid \mathtt{q} \triangleleft h_{\mathtt{q}} \mid \mathbb{M}_1 \tag{B.3}$$
$$\mathbb{M}' = \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft Q \mid \mathtt{q} \triangleleft h_{\mathtt{q}} \cdot (\mathtt{p}, \mathtt{m}(v)) \mid \mathbb{M}_1 \tag{B.4}$$
$$e \downarrow v \tag{B.5}$$
$$h_{\mathtt{q}} \neq \oslash \tag{B.6}$$
$$\mathbb{M}_1 = \prod_{i\in I} (\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i) \tag{B.7}$$

By (B.1) and Lemma B.1, we have that there exists $\Gamma; \Delta$ such that

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle \tag{B.8}$$
$$\vdash \mathtt{q}!\mathtt{m}\langle e\rangle.P : \Gamma(\mathtt{p}) \tag{B.9}$$
$$\vdash h_{\mathtt{p}} : \Delta(-, \mathtt{p}) \tag{B.10}$$
$$\vdash Q : \Gamma(\mathtt{q}) \tag{B.11}$$
$$\vdash h_{\mathtt{q}} : \Delta(-, \mathtt{q}) \tag{B.12}$$
$$\forall i \in I : \vdash P_i : \Gamma(\mathtt{p}_i) \tag{B.13}$$
$$\forall i \in I : \vdash h_i : \Delta(-, \mathtt{p}_i) \tag{B.14}$$

By (B.9) and 3 of Lemma B.1, we have that

$$\Gamma(\mathtt{p}) = \mathtt{q} \oplus \mathtt{m}(B).T \tag{B.15}$$
$$T_1 \leqslant T \tag{B.16}$$
$$\vdash P : T_1 \tag{B.17}$$
$$\vdash e : B \tag{B.18}$$

We now let

$$\Gamma' = \Gamma[\mathtt{p} \mapsto T] \tag{B.19}$$
$$\Delta' = \Delta[\mathtt{p}, \mathtt{q} \mapsto \Delta(\mathtt{p}, \mathtt{q}) \cdot \mathtt{m}(B)] \tag{B.20}$$

Then by [Γ-⊕] in Fig. 8, we have

$$\Gamma; \Delta \to_{\mathscr{R}} \Gamma'; \Delta' \tag{B.21}$$

Hence, using Theorem 4.20, we have that there exists $\langle \mathscr{C}'; G'\rangle$ such that

$$\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle \tag{B.22}$$
$$\langle \mathscr{C}; G\rangle \to_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle \tag{B.23}$$

Combine (B.22), (B.23) with

$$\vdash P : \Gamma'(\mathtt{p}) \qquad \text{(by (B.19), (B.16), (B.17) and [T-SUB])} \tag{B.24}$$

$$\vdash h_{\mathtt{p}} : \Delta'(-, \mathtt{p}) \quad \text{(by (B.20) and (B.10))} \tag{B.25}$$

$$\vdash Q : \Gamma'(\mathtt{q}) \qquad \text{(by (B.11) and (B.19))} \tag{B.26}$$

$$\vdash h_{\mathtt{q}} \cdot (\mathtt{p}, \mathtt{m}(v)) : \Delta'(-, \mathtt{q}) \quad \text{(by (B.20), (B.12), (B.5), (B.18), Lemma B.4, [T-MSG], and [T-$\cdot$])} \tag{B.27}$$

$$\forall i \in I : \vdash P_i : \Gamma'(\mathtt{p}_i) \quad \text{(by (B.13) and (B.19))} \tag{B.28}$$

$$\forall i \in I : \vdash h_i : \Delta'(-, \mathtt{p}_i) \quad \text{(by (B.14) and (B.20))} \tag{B.29}$$

We conclude that $\langle \mathscr{C}'; G' \rangle \vdash \mathbb{M}'$.

- Case [R-RCV]: similar to case [R-SEND] above, except that we proceed by [R-RCV], inversion of [T-EXT] (4 of Lemma B.1), and Lemma B.3.
- Case [R-SEND-↯]: we have

$$\mathbb{M} = \mathtt{p} \triangleleft \mathtt{q}!\mathtt{m}\langle e \rangle.P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M}_1 \tag{B.30}$$

$$\mathbb{M}' = \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M}_1 \tag{B.31}$$

$$\tag{B.32}$$

$$\mathbb{M}_1 = \prod_{i \in I} (\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i) \tag{B.33}$$

By (B.1) and Lemma B.1, we have that there exists $\Gamma; \Delta$ such that

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \tag{B.34}$$

$$\vdash \mathtt{q}!\mathtt{m}\langle e \rangle.P : \Gamma(\mathtt{p}) \tag{B.35}$$

$$\vdash h_{\mathtt{p}} : \Delta(-, \mathtt{p}) \tag{B.36}$$

$$\vdash \lightning : \Gamma(\mathtt{q}) \tag{B.37}$$

$$\vdash \oslash : \Delta(-, \mathtt{q}) \tag{B.38}$$

$$\forall i \in I : \vdash P_i : \Gamma(\mathtt{p}_i) \tag{B.39}$$

$$\forall i \in I : \vdash h_i : \Delta(-, \mathtt{p}_i) \tag{B.40}$$

It follows directly that

$$\Gamma(\mathtt{q}) = \mathsf{stop} \tag{B.41}$$

$$\Delta(-, \mathtt{q}) = \oslash \tag{B.42}$$

By (B.35) and 3 of Lemma B.1, we have that

$$\Gamma(\mathtt{p}) = \mathtt{q} \oplus \mathtt{m}(B).T \tag{B.43}$$

$$T_1 \leqslant T \tag{B.44}$$

$$\vdash P : T_1 \tag{B.45}$$

$$\vdash e : B \tag{B.46}$$

We now let

$$\Gamma' = \Gamma[\mathtt{p} \mapsto T] \tag{B.47}$$

$$\Delta' = \Delta[\mathtt{p}, \mathtt{q} \mapsto \Delta(\mathtt{p}, \mathtt{q}) \cdot \mathtt{m}(B)] \tag{B.48}$$

Then by [Γ-⊕] in Fig. 8, we have

$$\Gamma; \Delta \to_{\mathscr{R}} \Gamma'; \Delta' \tag{B.49}$$

Hence, using Theorem 4.20, we have that there exists $\langle \mathscr{C}'; G' \rangle$ such that

$$\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle \tag{B.50}$$

$$\langle \mathscr{C}; G \rangle \to_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle \tag{B.51}$$

Combine (B.50), (B.51) with

$$\vdash P : \Gamma'(\mathtt{p}) \quad \text{(by (B.47), (B.44), (B.45) and [T-SUB])} \tag{B.52}$$

$$\vdash h_{\mathtt{p}} : \Delta'(-, \mathtt{p}) \quad \text{(by (B.48) and (B.36))} \tag{B.53}$$

$$\vdash \lightning : \Gamma'(\mathtt{q}) \quad \text{(by (B.37), (B.41) and (B.47))} \tag{B.54}$$

$$\vdash \oslash : \Delta'(-, \mathtt{q}) \quad \text{(by (B.48), (B.38) and (B.42))} \tag{B.55}$$

$$\forall i \in I : \vdash P_i : \Gamma'(\mathtt{p}_i) \quad \text{(by (B.39) and (B.47))} \tag{B.56}$$

$$\forall i \in I : \vdash h_i : \Delta'(-, \mathtt{p}_i) \quad \text{(by (B.40) and (B.48))} \tag{B.57}$$

We conclude that $\langle \mathscr{C}'; G' \rangle \vdash \mathbb{M}'$.

- Case [R-RCV-⊙]: we have

$$\mathbb{M} = \mathtt{p} \triangleleft \sum_{i \in I} \mathtt{q}?\mathtt{m}_i(x_i).P_i \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M}_1 \tag{B.58}$$

$$\mathbb{M}' = \mathtt{p} \triangleleft P_k \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M}_1 \tag{B.59}$$

$$\mathbb{M}_1 = \prod_{i \in I} (\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i) \tag{B.60}$$

$$k \in I \tag{B.61}$$

$$\mathtt{m}_k = \mathsf{crash} \tag{B.62}$$

$$\nexists \mathtt{m}, v : (\mathtt{q}, \mathtt{m}(v)) \in h_{\mathtt{p}} \tag{B.63}$$

By (B.1) and Lemma B.1, we have that there exists $\Gamma; \Delta$ such that

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \tag{B.64}$$

$$\vdash \sum_{i \in I} \mathtt{q}?\mathtt{m}_i(x_i).P_i : \Gamma(\mathtt{p}) \tag{B.65}$$

$$\vdash h_{\mathtt{p}} : \Delta(-, \mathtt{p}) \tag{B.66}$$

$$\vdash \lightning : \Gamma(\mathtt{q}) \tag{B.67}$$

$$\vdash \oslash : \Delta(-, \mathtt{q}) \tag{B.68}$$

$$\forall i \in I : \vdash P_i : \Gamma(\mathtt{p}_i) \tag{B.69}$$

$$\forall i \in I : \vdash h_i : \Delta(-, \mathtt{p}_i) \tag{B.70}$$

It follows directly that

$$\Gamma(\mathtt{q}) = \mathsf{stop} \tag{B.71}$$

$$\Delta(-, \mathtt{q}) = \oslash \tag{B.72}$$

By (B.65), 4 of Lemma B.1, and [Sub-&], we have that

$$\Gamma(\mathtt{p}) = \mathtt{q}\&\{\mathtt{m_i}(B_i).T'_i\}_{i\in J} \tag{B.73}$$
$$J \subseteq I \tag{B.74}$$
$$\forall i \in J : T_i \leqslant T'_i \tag{B.75}$$
$$\{\mathtt{m}_l \mid l \in I\} \neq \{\mathsf{crash}\} \tag{B.76}$$
$$\nexists j \in I \setminus J : \mathtt{m}_j = \mathsf{crash} \tag{B.77}$$
$$\forall i \in I : x_i : B_i \vdash P_i : T_i \tag{B.78}$$

From (B.77) and (B.62), we get

$$k \in J \tag{B.79}$$

By (B.66) and (B.63), we also know

$$\Delta(\mathtt{q},\mathtt{p}) = \epsilon \tag{B.80}$$

We now let

$$\Gamma' = \Gamma[\mathtt{p} \mapsto T'_k] \tag{B.81}$$
$$\Delta' = \Delta \tag{B.82}$$

Then by [Γ-⊙] in Fig. 8, we have

$$\Gamma;\Delta \to_{\mathscr{R}} \Gamma';\Delta' \tag{B.83}$$

Hence, using Theorem 4.20, we have that there exists $\langle \mathscr{C}';G'\rangle$ such that

$$\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}';G'\rangle \tag{B.84}$$
$$\langle \mathscr{C};G\rangle \to_{\mathscr{R}} \langle \mathscr{C}';G'\rangle \tag{B.85}$$

Combine (B.84), (B.85) with

$$\vdash P_k : \Gamma'(\mathtt{p}) \quad \text{(by (B.81), (B.75), (B.78) and [T-sub])} \tag{B.86}$$
$$\vdash h_{\mathtt{p}} : \Delta'(-,\mathtt{p}) \quad \text{(by (B.82) and (B.66))} \tag{B.87}$$
$$\vdash \lightning : \Gamma'(\mathtt{q}) \quad \text{(by (B.67), (B.71) and (B.81))} \tag{B.88}$$
$$\vdash \oslash : \Delta'(-,\mathtt{q}) \quad \text{(by (B.82), (B.68) and (B.72))} \tag{B.89}$$
$$\forall i \in I : \vdash P_i : \Gamma'(\mathtt{p}_i) \quad \text{(by (B.69) and (B.81))} \tag{B.90}$$
$$\forall i \in I : \vdash h_i : \Delta'(-,\mathtt{p}_i) \quad \text{(by (B.70) and (B.82))} \tag{B.91}$$

We conclude that $\langle \mathscr{C}';G'\rangle \vdash \mathbb{M}'$.

- Case [R-↯]: we have

$$\mathbb{M} = \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathbb{M}_1 \tag{B.92}$$
$$\mathbb{M}' = \mathtt{p} \triangleleft \lightning \mid \mathtt{p} \triangleleft \oslash \mid \mathbb{M} \tag{B.93}$$
$$P \neq \mathbf{0} \tag{B.94}$$
$$\mathtt{p} \notin \mathscr{R} \tag{B.95}$$
$$\mathbb{M}_1 = \prod_{i\in I}(\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i) \tag{B.96}$$

By (B.1) and Lemma B.1, we have that there exists $\Gamma; \Delta$ such that

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \tag{B.97}$$
$$\vdash P : \Gamma(\mathtt{p}) \tag{B.98}$$
$$\vdash h_{\mathtt{p}} : \Delta(-, \mathtt{p}) \tag{B.99}$$
$$\forall i \in I : \vdash P_i : \Gamma(\mathtt{p}_i) \tag{B.100}$$
$$\forall i \in I : \vdash h_i : \Delta(-, \mathtt{p}_i) \tag{B.101}$$

By (B.94), (B.95), and (B.98), we have that

$$\Gamma(\mathtt{p}) \neq \mathsf{stop} \tag{B.102}$$
$$\Gamma(\mathtt{p}) \neq \mathsf{end} \tag{B.103}$$

We now let

$$\Gamma' = \Gamma[\mathtt{p} \mapsto \mathsf{stop}] \tag{B.104}$$
$$\Delta' = \Delta[\cdot, \mathtt{p} \mapsto \oslash] \tag{B.105}$$

Then by [Γ-↯] in Fig. 8, we have

$$\Gamma; \Delta \rightarrow_{\mathscr{R}} \Gamma'; \Delta' \tag{B.106}$$

Hence, using Theorem 4.20, we have that there exists $\langle \mathscr{C}'; G' \rangle$ such that

$$\Gamma'; \Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle \tag{B.107}$$
$$\langle \mathscr{C}; G \rangle \rightarrow_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle \tag{B.108}$$

Combine (B.107), (B.108) with

$$\vdash \lightning : \Gamma'(\mathtt{p}) \qquad \text{(by (B.104))} \tag{B.109}$$
$$\vdash \oslash : \Delta'(-, \mathtt{p}) \quad \text{(by (B.105))} \tag{B.110}$$
$$\forall i \in I : \vdash P_i : \Gamma'(\mathtt{p}_i) \quad \text{(by (B.100) and (B.104))} \tag{B.111}$$
$$\forall i \in I : \vdash h_i : \Delta'(-, \mathtt{p}_i) \quad \text{(by (B.101) and (B.105))} \tag{B.112}$$

We conclude that $\langle \mathscr{C}'; G' \rangle \vdash \mathbb{M}'$.

- Case [R-COND-T]: we have

$$\mathbb{M} = \mathtt{p} \triangleleft \mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q \mid \mathtt{p} \triangleleft h \mid \mathbb{M}_1 \tag{B.113}$$
$$\mathbb{M}' = \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h \mid \mathbb{M}_1 \tag{B.114}$$
$$e \downarrow \mathtt{true} \tag{B.115}$$
$$\mathbb{M}_1 = \prod_{i \in I} (\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i) \tag{B.116}$$

By (B.1) and Lemma B.1, we have that there exists $\Gamma; \Delta$ such that

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \tag{B.117}$$
$$\vdash \mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q : \Gamma(\mathtt{p}) \tag{B.118}$$
$$\vdash h : \Delta(-, \mathtt{p}) \tag{B.119}$$
$$\forall i \in I : \vdash P_i : \Gamma(\mathtt{p}_i) \tag{B.120}$$
$$\forall i \in I : \vdash h_i : \Delta(-, \mathtt{p}_i) \tag{B.121}$$

By (B.115), (B.118), 5 of Lemma B.1 and [T-SUB], we have that

$$\vdash e : \mathsf{bool} \tag{B.122}$$
$$\vdash P : \Gamma(\mathtt{p}) \tag{B.123}$$
$$\vdash Q : \Gamma(\mathtt{p}) \tag{B.124}$$

Combine (B.123) with (B.119), (B.120), and (B.121), we can conclude that $\langle \mathscr{C}; G\rangle \vdash \mathbb{M}'$, as desired.

- Case [R-COND-F]: similar to the case [R-COND-T].
- Case [R-STRUCT]: assume that $\mathbb{M} \to \mathbb{M}'$ is derived from

$$\mathbb{M} \Rrightarrow \mathbb{M}_1 \tag{B.125}$$
$$\mathbb{M}_1 \to \mathbb{M}_1' \tag{B.126}$$
$$\mathbb{M}_1' \Rrightarrow \mathbb{M}' \tag{B.127}$$

From (B.125), (B.1), by (3) of Lemma B.2, we have that $\langle \mathscr{C}; G\rangle \vdash \mathbb{M}_1$. By induction hypothesis, either $\langle \mathscr{C}; G\rangle \vdash \mathbb{M}_1'$ or there exists $\langle \mathscr{C}'; G'\rangle$ such that $\langle \mathscr{C}; G\rangle \to_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$ and $\langle \mathscr{C}'; G'\rangle \vdash \mathbb{M}_1'$. Now by (B.127) and (3) of Lemma B.2, we have that either $\langle \mathscr{C}; G\rangle \vdash \mathbb{M}'$ or $\langle \mathscr{C}'; G'\rangle \vdash \mathbb{M}'$, as desired. □

Theorem 5.3 (Session Fidelity)**.** *If* $\langle \mathscr{C}; G\rangle \vdash \mathbb{M}$ *and* $\langle \mathscr{C}; G\rangle \to_{\mathscr{R}}$, *then there exists* $\mathbb{M}'$ *and* $\langle \mathscr{C}'; G'\rangle$ *such that* $\langle \mathscr{C}; G\rangle \to_{\mathscr{R}} \langle \mathscr{C}'; G'\rangle$, $\mathbb{M} \to_{\mathscr{R}}^* \mathbb{M}'$ *and* $\langle \mathscr{C}'; G'\rangle \vdash \mathbb{M}'$.

Proof. Let us recap the assumptions:

$$\langle \mathscr{C}; G\rangle \vdash \mathbb{M} \tag{B.128}$$
$$\langle \mathscr{C}; G\rangle \to_{\mathscr{R}} \tag{B.129}$$

The proof proceeds by induction on the derivation of $\langle \mathscr{C}; G\rangle \to_{\mathscr{R}}$.

- Case [GR-↯]: by inversion of [GR-↯], we have

$$\mathtt{p} \notin \mathscr{R} \tag{B.130}$$
$$\mathtt{p} \in \mathrm{roles}(G) \tag{B.131}$$
$$\langle \mathscr{C}; G\rangle \xrightarrow{\mathtt{p}\lightning}_{\mathscr{R}} \langle \mathscr{C} \cup \{\mathtt{p}\}; G \lightning \mathtt{p}\rangle \tag{B.132}$$

We can assume that $\mathbb{M}$ is of the form

$$\mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathbb{M}_1 \tag{B.133}$$
$$\mathbb{M}_1 = \prod_{i \in I} (\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i) \tag{B.134}$$
$$P \neq \mathbf{0} \tag{B.135}$$
$$\mathtt{p} \notin \mathscr{R} \tag{B.136}$$

Then by (B.128) and Lemma B.1, there exists $\Gamma;\Delta$ such that

$$\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle \tag{B.137}$$
$$\vdash P : \Gamma(\mathtt{p}) \tag{B.138}$$
$$\vdash h_{\mathtt{p}} : \Delta(-,\mathtt{p}) \tag{B.139}$$
$$\forall i \in I : \vdash P_i : \Gamma(\mathtt{p}_i) \tag{B.140}$$
$$\forall i \in I : \vdash h_i : \Delta(-,\mathtt{p}_i) \tag{B.141}$$

From (B.132), by Theorem 4.21, there is $\Gamma';\Delta'$ such that

$$\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C} \cup \{\mathtt{p}\}; G\!\upharpoonright\!\lightning\,\mathtt{p}\rangle \tag{B.142}$$
$$\Gamma;\Delta \xrightarrow{\mathtt{p}\lightning} \Gamma';\Delta' \tag{B.143}$$

Using (B.135), (B.136), (B.138) and [Γ-↯] in Fig. 8, we get

$$\Gamma' = \Gamma[\mathtt{p} \mapsto \mathsf{stop}] \tag{B.144}$$
$$\Delta' = \Delta[\cdot,\mathtt{p} \mapsto \oslash] \tag{B.145}$$

It follows that

$$\vdash \lightning : \Gamma'(\mathtt{p}) \qquad \text{(by (B.144))} \tag{B.146}$$
$$\vdash \oslash : \Delta'(-,\mathtt{p}) \quad \text{(by (B.145))} \tag{B.147}$$
$$\forall i \in I : \vdash P_i : \Gamma'(\mathtt{p}_i) \quad \text{(by (B.140) and (B.144))} \tag{B.148}$$
$$\forall i \in I : \vdash h_i : \Delta'(-,\mathtt{p}_i) \quad \text{(by (B.141) and (B.145))} \tag{B.149}$$

Therefore, by [R-↯] and [T-SESS], we can conclude that there exists $\mathbb{M}' = \mathtt{p} \triangleleft \lightning \mid \mathtt{p} \triangleleft \oslash \mid \mathbb{M}_1$ and $\langle \mathscr{C}';G'\rangle = \langle \mathscr{C} \cup \{\mathtt{p}\}; G\!\upharpoonright\!\lightning\,\mathtt{p}\rangle$ such that $\langle \mathscr{C};G\rangle \xrightarrow{\mathtt{p}\lightning}_{\mathscr{R}} \langle \mathscr{C}';G'\rangle$, $\mathbb{M} \not\to_{\mathscr{R}}$, and $\langle \mathscr{C}';G'\rangle \vdash \mathbb{M}'$.

- Case [GR-⊕]: by inversion of [GR-⊕], we have

$$j \in I \tag{B.150}$$
$$\mathtt{m}_j \neq \mathsf{crash} \tag{B.151}$$
$$\langle \mathscr{C}; \mathtt{p}\to\mathtt{q}\colon \{\mathtt{m}_i(B_i).G'_i\}_{i\in I}\rangle \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m}_j(B_j)}_{\mathscr{R}} \langle \mathscr{C}; \mathtt{p}\rightsquigarrow\mathtt{q}:j\ \{\mathtt{m}_i(B_i).G'_i\}_{i\in I}\rangle \tag{B.152}$$

We can assume that $\mathbb{M}$ is of the form

$$\mathtt{p} \triangleleft \mathtt{q}!\mathtt{m}\langle e\rangle.P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft Q \mid \mathtt{q} \triangleleft h_{\mathtt{q}} \mid \mathbb{M}_1 \tag{B.153}$$
$$\mathbb{M}_1 = \prod_{i\in I} (\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i) \tag{B.154}$$
$$e \downarrow v \tag{B.155}$$
$$h_{\mathtt{q}} \neq \oslash \tag{B.156}$$

Then by (B.128) and Lemma B.1, there exists $\Gamma;\Delta$ such that

$$\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle \tag{B.157}$$
$$\vdash \mathtt{q}!\mathtt{m}\langle e\rangle.P : \Gamma(\mathtt{p}) \tag{B.158}$$
$$\vdash h_{\mathtt{p}} : \Delta(-,\mathtt{p}) \tag{B.159}$$
$$\vdash Q : \Gamma(\mathtt{q}) \tag{B.160}$$
$$\vdash h_{\mathtt{q}} : \Delta(-,\mathtt{q}) \tag{B.161}$$
$$\forall i \in I : \vdash P_i : \Gamma(\mathtt{p}_i) \tag{B.162}$$
$$\forall i \in I : \vdash h_i : \Delta(-,\mathtt{p}_i) \tag{B.163}$$

By (B.158) and 3 of Lemma B.1, we have that

$$\Gamma(\mathtt{p}) = \mathtt{q}\oplus\mathtt{m}(B).T \tag{B.164}$$
$$T_1 \leqslant T \tag{B.165}$$
$$\vdash P : T_1 \tag{B.166}$$
$$\vdash e : B \tag{B.167}$$

From (B.152), by Theorem 4.21, there are $\Gamma';\Delta'$ and $\alpha = \mathtt{p}\oplus\mathtt{q} : \mathtt{m}_{\mathtt{k}}(B_k)$ such that

$$k \in I \tag{B.168}$$
$$\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; \mathtt{p}\rightsquigarrow\mathtt{q}{:}j\,\{\mathtt{m}_{\mathtt{i}}(B_i).G'_i\}_{i\in I}\rangle \tag{B.169}$$
$$\Gamma;\Delta \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m}_{\mathtt{k}}(B_k)} \Gamma';\Delta' \tag{B.170}$$

Using (B.164), (B.168), and [Γ-⊕] in Fig. 8, we get

$$\Gamma' = \Gamma[\mathtt{p} \mapsto T] \tag{B.171}$$
$$\Delta' = \Delta[\mathtt{p},\mathtt{q} \mapsto \Delta(\mathtt{p},\mathtt{q})\cdot\mathtt{m}(B)] \tag{B.172}$$

It follows that

$$\vdash P : \Gamma'(\mathtt{p}) \quad \text{(by (B.171), (B.165), (B.166) and [T-SUB])} \tag{B.173}$$
$$\vdash h_{\mathtt{p}} : \Delta'(-,\mathtt{p}) \quad \text{(by (B.172) and (B.159))} \tag{B.174}$$
$$\vdash Q : \Gamma'(\mathtt{q}) \quad \text{(by (B.160) and (B.171))} \tag{B.175}$$
$$\vdash h_{\mathtt{q}} : \Delta'(-,\mathtt{q}) \quad \text{(by (B.172), (B.161) and 9, 10 of Lemma B.1)} \tag{B.176}$$
$$\forall i \in I : \vdash P_i : \Gamma'(\mathtt{p}_i) \quad \text{(by (B.162) and (B.171))} \tag{B.177}$$
$$\forall i \in I : \vdash h_i : \Delta'(-,\mathtt{p}_i) \quad \text{(by (B.163) and (B.172))} \tag{B.178}$$

Therefore, by [R-SEND] and [T-SESS], we can conclude that there exists $\mathbb{M}' = \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft Q \mid \mathtt{q} \triangleleft h_{\mathtt{q}} \cdot (\mathtt{p},\mathtt{m}(v)) \mid \mathbb{M}_1$ and $\langle \mathscr{C}';G'\rangle = \langle \mathscr{C}; \mathtt{p}\rightsquigarrow\mathtt{q}{:}j\,\{\mathtt{m}_{\mathtt{i}}(B_i).G'_i\}_{i\in I}\rangle$ such that $\langle \mathscr{C};G\rangle \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m}_{\mathtt{j}}(B_j)}_{\mathscr{R}} \langle \mathscr{C}';G'\rangle$, $\mathbb{M} \to \mathbb{M}'$, and $\langle \mathscr{C}';G'\rangle \vdash \mathbb{M}'$.

- Case [GR-&]: similar to case [GR-⊕] above, except that we proceed by [GR-&], [R-RCV], inversion of [T-EXT] (4 of Lemma B.1), and Lemma B.3.

- Case [GR-↯m]: by inversion of [GR-⊕], we have

$$j \in I \tag{B.179}$$
$$\mathtt{m}_j \neq \mathsf{crash} \tag{B.180}$$
$$\langle \mathscr{C}; \mathtt{p} \to \mathtt{q}^{\lightning} : \{\mathtt{m_i}(B_i).G'_i\}_{i\in I}\rangle \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m_j}(B_j)}_{\mathscr{R}} \langle \mathscr{C}; G'_j\rangle \tag{B.181}$$

We can assume that $\mathbb{M}$ is of the form

$$\mathtt{p} \triangleleft \mathtt{q}!\mathtt{m}\langle e\rangle.P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M}_1 \tag{B.182}$$
$$\mathbb{M}_1 = \prod_{i\in I} (\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i) \tag{B.183}$$

Then by (B.128) and Lemma B.1, there exists $\Gamma;\Delta$ such that

$$\Gamma;\Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G\rangle \tag{B.184}$$
$$\vdash \mathtt{q}!\mathtt{m}\langle e\rangle.P : \Gamma(\mathtt{p}) \tag{B.185}$$
$$\vdash h_{\mathtt{p}} : \Delta(-,\mathtt{p}) \tag{B.186}$$
$$\vdash \lightning : \Gamma(\mathtt{q}) \tag{B.187}$$
$$\vdash \oslash : \Delta(-,\mathtt{q}) \tag{B.188}$$
$$\forall i \in I : \vdash P_i : \Gamma(\mathtt{p}_i) \tag{B.189}$$
$$\forall i \in I : \vdash h_i : \Delta(-,\mathtt{p}_i) \tag{B.190}$$

By (B.185) and 3 of Lemma B.1, we have that

$$\Gamma(\mathtt{p}) = \mathtt{q}\oplus\mathtt{m}(B).T \tag{B.191}$$
$$T_1 \leqslant T \tag{B.192}$$
$$\vdash P : T_1 \tag{B.193}$$

From (B.181), by Theorem 4.21, there are $\Gamma';\Delta'$ and $\alpha = \mathtt{p}\oplus\mathtt{q} : \mathtt{m_k}(B_k)$ such that

$$k \in I \tag{B.194}$$
$$\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G'_j\rangle \tag{B.195}$$
$$\Gamma;\Delta \xrightarrow{\mathtt{p}\oplus\mathtt{q}:\mathtt{m_k}(B_k)} \Gamma';\Delta' \tag{B.196}$$

Using (B.191), (B.194), and [Γ-⊕] in Fig. 8, we get

$$\Gamma' = \Gamma[\mathtt{p} \mapsto T] \tag{B.197}$$
$$\Delta' = \Delta[\mathtt{p},\mathtt{q} \mapsto \Delta(\mathtt{p},\mathtt{q})\cdot\mathtt{m}(B)] \tag{B.198}$$

It follows that

$$\vdash P : \Gamma'(\mathtt{p}) \quad \text{(by (B.197), (B.192), (B.193) and [T-SUB])} \tag{B.199}$$
$$\vdash h_{\mathtt{p}} : \Delta'(-,\mathtt{p}) \quad \text{(by (B.198) and (B.186))} \tag{B.200}$$
$$\vdash \lightning : \Gamma'(\mathtt{q}) \quad \text{(by (B.187) and (B.197))} \tag{B.201}$$
$$\vdash \oslash : \Delta'(-,\mathtt{q}) \quad \text{(by (B.198) and (B.188))} \tag{B.202}$$
$$\forall i \in I : \vdash P_i : \Gamma'(\mathtt{p}_i) \quad \text{(by (B.189) and (B.197))} \tag{B.203}$$
$$\forall i \in I : \vdash h_i : \Delta'(-,\mathtt{p}_i) \quad \text{(by (B.190) and (B.198))} \tag{B.204}$$

Therefore, by [R-SEND-↯] and [T-SESS], we can conclude that there exists $\mathbb{M}' = \mathtt{p} \triangleleft P \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M}_1$ and $\langle \mathscr{C}'; G' \rangle = \langle \mathscr{C}; G'_j \rangle$ such that $\langle \mathscr{C}; G \rangle \xrightarrow{\mathtt{p} \oplus \mathtt{q}:\mathtt{m}_j(B_j)}_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle$, $\mathbb{M} \to \mathbb{M}'$, and $\langle \mathscr{C}'; G' \rangle \vdash \mathbb{M}'$.

- Case [GR-⊙]: by inversion of [GR-⊙], we have

$$j \in I \tag{B.205}$$
$$\mathtt{m}_j = \mathsf{crash} \tag{B.206}$$
$$\langle \mathscr{C}; \mathtt{q}^{\lightning} \rightsquigarrow \mathtt{p}{:}j \left\{\mathtt{m}_{\mathtt{i}}(B_i).G'_i\right\}_{i \in I} \rangle \xrightarrow{\mathtt{p} \odot \mathtt{q}}_{\mathscr{R}} \langle \mathscr{C}; G'_j \rangle \tag{B.207}$$

We can assume that $\mathbb{M}$ is of the form

$$\mathtt{p} \triangleleft \sum_{i \in I} \mathtt{q}?\mathtt{m}_i(x_i).P_i \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M}_1 \tag{B.208}$$
$$\mathbb{M}_1 = \prod_{i \in I} (\mathtt{p}_i \triangleleft P_i \mid \mathtt{p}_i \triangleleft h_i) \tag{B.209}$$
$$k \in I \tag{B.210}$$
$$\mathtt{m}_k = \mathsf{crash} \tag{B.211}$$
$$\nexists \mathtt{m}, v : (\mathtt{q}, \mathtt{m}(v)) \in h_{\mathtt{p}} \tag{B.212}$$

Then by (B.128) and Lemma B.1, there exists $\Gamma; \Delta$ such that

$$\Gamma; \Delta \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G \rangle \tag{B.213}$$
$$\vdash \sum_{i \in I} \mathtt{q}?\mathtt{m}_i(x_i).P_i : \Gamma(\mathtt{p}) \tag{B.214}$$
$$\vdash h_{\mathtt{p}} : \Delta(-, \mathtt{p}) \tag{B.215}$$
$$\vdash \lightning : \Gamma(\mathtt{q}) \tag{B.216}$$
$$\vdash \oslash : \Delta(-, \mathtt{q}) \tag{B.217}$$
$$\forall i \in I : \vdash P_i : \Gamma(\mathtt{p}_i) \tag{B.218}$$
$$\forall i \in I : \vdash h_i : \Delta(-, \mathtt{p}_i) \tag{B.219}$$

It follows directly that

$$\Gamma(\mathtt{q}) = \mathsf{stop} \tag{B.220}$$
$$\Delta(-, \mathtt{q}) = \oslash \tag{B.221}$$

By (B.214), 4 of Lemma B.1, and [SUB-&], we have that

$$\Gamma(\mathtt{p}) = \mathtt{q} \& \left\{\mathtt{m}_{\mathtt{i}}(B_i).T'_i\right\}_{i \in J} \tag{B.222}$$
$$J \subseteq I \tag{B.223}$$
$$\forall i \in J : T_i \leqslant T'_i \tag{B.224}$$
$$\{\mathtt{m}_l \mid l \in I\} \neq \{\mathsf{crash}\} \tag{B.225}$$
$$\nexists j \in I \setminus J : \mathtt{m}_j = \mathsf{crash} \tag{B.226}$$
$$\forall i \in I : x_i : B_i \vdash P_i : T_i \tag{B.227}$$

From (B.226) and (B.211), we get

$$k \in J \tag{B.228}$$

By (B.215) and (B.212), we also know

$$\Delta(\mathtt{q},\mathtt{p}) = \epsilon \tag{B.229}$$

From (B.207), by Theorem 4.21, there is $\Gamma';\Delta'$ such that

$$\Gamma';\Delta' \sqsubseteq_{\mathscr{R}} \langle \mathscr{C}; G'_j \rangle \tag{B.230}$$

$$\Gamma;\Delta \xrightarrow{\mathtt{p}\odot\mathtt{q}} \Gamma';\Delta' \tag{B.231}$$

Using (B.222), (B.220), (B.228), (B.211), (B.229), and [Γ-⊙] in Fig. 8, we get

$$\Gamma' = \Gamma[\mathtt{p} \mapsto T'_k] \tag{B.232}$$

$$\Delta' = \Delta \tag{B.233}$$

It follows that

$$\vdash P_k : \Gamma'(\mathtt{p}) \quad \text{(by (B.232), (B.224), (B.227) and [T-SUB])} \tag{B.234}$$

$$\vdash h_{\mathtt{p}} : \Delta'(-,\mathtt{p}) \quad \text{(by (B.233) and (B.215))} \tag{B.235}$$

$$\vdash \lightning : \Gamma'(\mathtt{q}) \quad \text{(by (B.216), (B.220) and (B.232))} \tag{B.236}$$

$$\vdash \oslash : \Delta'(-,\mathtt{q}) \quad \text{(by (B.233), (B.217) and (B.221))} \tag{B.237}$$

$$\forall i \in I : \vdash P_i : \Gamma'(\mathtt{p}_i) \quad \text{(by (B.218) and (B.232))} \tag{B.238}$$

$$\forall i \in I : \vdash h_i : \Delta'(-,\mathtt{p}_i) \quad \text{(by (B.219) and (B.233))} \tag{B.239}$$

Therefore, by [R-RCV-⊙] and [T-SESS], we can conclude that there exists $\mathbb{M}' = \mathtt{p} \triangleleft P_k \mid \mathtt{p} \triangleleft h_{\mathtt{p}} \mid \mathtt{q} \triangleleft \lightning \mid \mathtt{q} \triangleleft \oslash \mid \mathbb{M}_1$ and $\langle \mathscr{C}'; G' \rangle = \langle \mathscr{C}; G'_j \rangle$ such that $\langle \mathscr{C}; G \rangle \xrightarrow{\mathtt{p}\odot\mathtt{q}}_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle$, $\mathbb{M} \to \mathbb{M}'$, and $\langle \mathscr{C}'; G' \rangle \vdash \mathbb{M}'$.

- Case [GR-μ]: follows by Lemma B.5 and inductive hypothesis.
- Cases [GR-CTX-I] and [GR-CTX-II]: these two cases do not need to be considered, as the reduction $\langle \mathscr{C}; G \rangle \to_{\mathscr{R}}$ can always be triggered by the preceding cases. □

Theorem 5.5 (Session Deadlock-Freedom)**.** *If* $\langle \mathscr{C}; G \rangle \vdash \mathbb{M}$*, then* $\mathbb{M}$ *is deadlock-free.*

Proof. Assume that $\langle \mathscr{C}; G \rangle \vdash \mathbb{M}$ and $\mathbb{M} \to^*_{\mathscr{R}} \mathbb{M}' \not\to_{\mathscr{R}}$, we need to prove that either $\mathbb{M}' \Rrightarrow \mathtt{p} \triangleleft \mathbf{0} \mid \mathtt{p} \triangleleft \epsilon$ for some $\mathtt{p}$, or $\mathbb{M}' \Rrightarrow \prod_{i\in I}(\mathtt{p}_i \triangleleft \lightning \mid \mathtt{p}_i \triangleleft \oslash)$ with $I \neq \emptyset$.

By applying Theorem 5.2 (subject reduction) repeatedly as needed, there exists a global type $\langle \mathscr{C}'; G' \rangle$ such that $\langle \mathscr{C}; G \rangle \to^*_{\mathscr{R}} \langle \mathscr{C}'; G' \rangle$ and $\langle \mathscr{C}'; G' \rangle \vdash \mathbb{M}'$. Since no further reductions are possible for $\mathbb{M}'$, by Theorem 5.3 (session fidelity), the global type $\langle \mathscr{C}'; G' \rangle$ cannot be reduced further either, implying that $\langle \mathscr{C}'; G' \rangle$ must be of the form $\langle \mathscr{C}'; \mathsf{end} \rangle$.

Furthermore, using [T-SESS] and the proof for Lemma A.26, we can conclude that $\mathbb{M}'$ is of the form $\prod_{i\in I}(\mathtt{p}_i \triangleleft P'_i \mid \mathtt{p}_i \triangleleft h'_i)$ such that $\forall i \in I : \vdash P'_i : \Gamma'(\mathtt{p}_i)$, $\vdash h'_i : \Delta'(-,\mathtt{p}_i)$, where $\Gamma' = \Gamma'_{\mathsf{end}}, \Gamma'_{\lightning}$, with $\forall \mathtt{p} \in \mathrm{dom}(\Gamma'_{\mathsf{end}}) : \Gamma'(\mathtt{p}) = \mathsf{end}$, $\forall \mathtt{p} \in \mathrm{dom}\left(\Gamma'_{\lightning}\right) : \Gamma'(\mathtt{p}) = \mathsf{stop}$, and for any $\mathtt{p},\mathtt{q}$, if $\mathtt{q} \in \mathrm{dom}\left(\Gamma'_{\lightning}\right), \Delta'(\cdot,\mathtt{q}) = \oslash$, and otherwise, $\Delta'(\mathtt{p},\mathtt{q}) = \epsilon$.

Then, by applying [T-ϵ], [T-$\oslash$], [T-$\lightning$], [T-$\mathbf{0}$], and the precongruence rules $\mathtt{p} \triangleleft \mathbf{0} \mid \mathtt{p} \triangleleft \epsilon \mid \mathbb{M} \Rrightarrow \mathbb{M}$ and $\mathbb{M} \Rrightarrow \mathbb{M}'$ and $\mathbb{M}' \Rrightarrow \mathbb{M}'' \implies \mathbb{M} \Rrightarrow \mathbb{M}''$, the thesis follows. □

Theorem 5.7 (Session Liveness)**.** *If* $\langle \mathscr{C}; G \rangle \vdash \mathbb{M}$*, then* $\mathbb{M}$ *is live.*

Proof. The proof follows the same structure as Theorem 5.5, but instead uses the definition of live sessions (Definition 5.6), the proof for Lemma A.27, and the relevant typing rules in Fig. 9. □