Secure Multiparty Sessions with Topics*

Ilaria Castellani
INRIA Sophia Antipolis, France

Mariangiola Dezani-Ciancaglini†
University of Turin, Italy

Ugo de'Liguoro‡
University of Turin, Italy

Multiparty session calculi have been recently equipped with security requirements, in order to guarantee properties such as access control and leak freedom. However, the proposed security requirements seem to be overly restrictive in some cases. In particular, a party is not allowed to communicate any kind of public information after receiving a secret information. This does not seem justified in case the two pieces of information are totally unrelated. The aim of the present paper is to overcome this restriction, by designing a type discipline for a simple multiparty session calculus, which classifies messages according to their topics and allows unrestricted sequencing of messages on independent topics.

1 Introduction

Today's distributed computing environment strongly relies on communication. Communication often takes place among multiple parties, which do not trust each other. This new scenario has spurred an active trend of research on safety and security properties for multiparty interactions. It is often the case that such interactions are "structured", i.e. they follow a specified protocol. Since their introduction in [5] (as an extension of binary session calculi), *multiparty session calculi* have been widely used to model structured communications among multiple parties. Session calculi are endowed with particular behavioural types called *session types*, which ensure that communications are not blocked and follow the expected protocol. Lately, multiparty session calculi have been enriched with security requirements, in order to ensure properties such as access control and leak freedom. An account of security analysis in multiparty session calculi and similar formalisms may be found in the recent survey [1].

A drawback of the existing security-enriched session calculi (such as those reviewed in [1]) is that the security requirements are overly restrictive in some cases. In particular, a party is not allowed to communicate any kind of public information after receiving a secret information. This does not seem justified in case the two pieces of information are totally unrelated. The aim of the present paper is to overcome this restriction, by designing a type discipline for a simple multiparty session calculus, which classifies messages according to their *topics* and allows unrestricted sequencing of messages on independent topics. In this way, we can safely type processes that are rejected by previous type systems.

We start by illustrating our approach with a familiar example.

Example 1.1. *A Programme Committee (PC) discussion may be described as a session whose participants are the PC members and whose main topics are the submitted papers. All papers are assumed to be unrelated unless they share some author. A further topic, unrelated to the papers, is constituted by a bibliographic database, which is public but possibly not easily accessible to all PC members; hence all PC members are allowed to ask other PC members to fetch a document in the database for them. Other topics, unrelated to the previous ones, are administrative data of interest to the PC, like email addresses.*

*Partly supported by the COST Action IC1201 BETTY.

†Partly supported by EU H2020-644235 Rephrase project, EU H2020-644298 HyVar project, ICT COST Actions IC1402 ARVI and Ateneo/CSP project RunVar.

‡Partly supported by EU H2020-644235 Rephrase project, EU H2020-644298 HyVar project, ICT COST Actions IC1402 ARVI and Ateneo/CSP project RunVar.

At the start of the session, all PC members receive a number of papers to review. During the discussion, PC members receive reviews and feedback on the papers in their lot, but possibly also on other papers for which they have not declared conflict. In this scenario, our typing will ensure the following properties:

1. *A PC member P_1 who received confidential information on paper φ can forward this information to another PC member P_2 if and only P_2 is not in conflict with paper φ nor with any related paper;*
2. *A PC member who received confidential information on some paper φ can subsequently send an email address to any other PC member, including those in conflict with paper φ;*
3. *The PC chair P_0 is allowed to request a document belonging to the bibliographic database to any PC member at any time, even after receiving confidential information on some paper φ. This could happen for instance if a PC member P_1 in charge of paper φ wishes to compare it with a previous paper by a PC member P_2 who is in conflict with paper φ. Suppose this paper is in the database but P_1 cannot access it; then P_1 will express her concerns about paper φ to the PC chair P_0 and ask him to retrieve the document from the database. The point is that P_0 himself may not have an easy access to the document; in this case P_0 will forward the request directly to P_2. Intuitively, this should be allowed because the requested document has the topic ψ of the database, which is not related to topic φ.*

In the above example, Property 1 is an access control (AC) property, which will be handled by assigning to each participant a reading level for each topic; Property 2 is a leak freedom (LF) property, where the usual "no write-down" condition is relaxed when the topic of the output is independent from that of the preceding input; finally, Property 3 involves both AC and LF issues. Our type system will ensure a *safety* property that is a combination of AC and of our relaxed LF property.

The next sections present the untyped calculus, the safety definition, the type system and the main properties of the typed calculus.

2 Synchronous Multiparty Session Calculus

We introduce here our synchronous multiparty session calculus, which is essentially the LTS version of the calculus considered in [4].

Syntax. A multiparty session is an abstraction for describing multiparty communication protocols [5]. It consists of a series of interactions between a fixed number of participants.

We use the following base sets: *security levels*, ranged over by $\ell, \ell', \ldots$; *topics*, ranged over by $\varphi, \psi, \ldots$; *values with levels and topics*, ranged over by $\mathsf{v}^{\ell,\varphi}, \mathsf{u}^{\ell',\psi}, \ldots$; *expressions*, ranged over by $e, e', \ldots$; *expression variables*, ranged over by $x, y, z \ldots$; *labels*, ranged over by $\lambda, \lambda', \ldots$; *session participants*, ranged over by $\mathtt{p}, \mathtt{q}, \ldots$; *process variables*, ranged over by $X, Y, \ldots$; *processes*, ranged over by $P, Q, \ldots$; and *multiparty sessions*, ranged over by $\mathscr{M}, \mathscr{M}', \ldots$.

Processes P are defined by:

$$P \quad ::= \quad \mathtt{q}!\lambda(e).P \;\mid\; \mathtt{p}?\lambda(x).Q \;\mid\; P \oplus P \;\mid\; P + P \;\mid\; \mu X.P \;\mid\; X \;\mid\; \mathbf{0}$$

The output process $\mathtt{q}!\lambda(e).P$ sends the value of expression e with label λ to participant $\mathtt{q}$. The input process $\mathtt{p}?\lambda(x).Q$ waits for the value of an expression with label λ from participant $\mathtt{p}$. The operators of internal and external choice, denoted $\oplus$ and $+$ respectively, are standard. We take an equi-recursive view of processes, not distinguishing between a process $\mu X.P$ and its unfolding $P\{\mu X.P/X\}$. We assume that the recursive processes are guarded, i.e. $\mu X.X$ is not a process.

A *multiparty session* $\mathscr{M}$ is a parallel composition of pairs (denoted by $\mathtt{p} \triangleleft P$) made of a participant and a process:

$$\mathscr{M} \quad ::= \quad \mathtt{p} \triangleleft P \;\mid\; \mathscr{M} \mid \mathscr{M}$$

We will use $\sum_{i\in I} P_i$ as short for $P_1 + \ldots + P_n$, and $\prod_{i\in I} \mathtt{p}_i \triangleleft P_i$ as short for $\mathtt{p}_1 \triangleleft P_1 \mid \ldots \mid \mathtt{p}_n \triangleleft P_n$, where $I = \{1,\ldots,n\}$.

Security levels and topics, which appear as superscripts of values, are used to classify values according to two criteria: their degree of confidentiality and their subject. The use of these two parameters will become clear in Section 3.

Our calculus is admittedly very simple, since processes are sequential and thus cannot be involved in more than one session at a time. As a consequence, it is not necessary to introduce explicit session channels: within a session, processes are identified as session participants and can directly communicate with each other, without ambiguity since the I/O operations mention the communicating partner.

Operational semantics The value $\mathtt{v}^{\ell,\varphi}$ of an expression e (notation $e \downarrow \mathtt{v}^{\ell,\varphi}$) is defined as expected, provided that all the values appearing in e have the same topic φ (this will be guaranteed by our typing) and the join of their security levels is ℓ. The semantics of processes and sessions is given by means of two separate LTS's. The actions of processes, ranged over by ϑ, are either the silent action τ or a visible I/O action α of the form $\mathtt{q}!\lambda(\mathtt{v}^{\ell,\varphi})$ or $\mathtt{p}?\lambda(\mathtt{v}^{\ell,\varphi})$. The actions of sessions, ranged over by κ, are either τ or a *message* of the form $\mathtt{p}(\lambda,\mathtt{v}^{\ell,\varphi})\mathtt{q}$.

The LTS's for processes and sessions are given by the rules in Table 2, defined up to a standard structural congruence denoted by $\equiv$ (by abuse of notation we use the same symbol for both processes and sessions), whose definition is in Table 1.

[S-INTCH 1] $P \oplus Q \equiv Q \oplus P$ [S-INTCH 2] $(P \oplus Q) \oplus R \equiv P \oplus (Q \oplus R)$

[S-EXTCH 1] $P + Q \equiv Q + P$ [S-EXTCH 2] $(P + Q) + R \equiv P + (Q + R)$

[S-REC] $\mu X.P \equiv P\{\mu X.P/X\}$ [S-MULTI] $P \equiv Q \Rightarrow \mathtt{p} \triangleleft P \equiv \mathtt{p} \triangleleft Q$ [S-PAR 1] $\mathtt{p} \triangleleft \mathbf{0} \mid \mathscr{M} \equiv \mathscr{M}$

[S-PAR 2] $\mathscr{M} \mid \mathscr{M}' \equiv \mathscr{M}' \mid \mathscr{M}$ [S-PAR 3] $(\mathscr{M} \mid \mathscr{M}') \mid \mathscr{M}'' \equiv \mathscr{M} \mid (\mathscr{M}' \mid \mathscr{M}'')$

Table 1: Structural congruence.

3 Safety

Our notion of safety for sessions has two facets: *access control* and information flow security or *leak-freedom*. We assume that security levels ℓ, ℓ' form a finite lattice, ordered by $\sqsubseteq$. We denote by $\sqcup$ and $\sqcap$ the join and meet operations on the lattice, and by $\bot$ and $\top$ its bottom and top elements. The partial ordering $\sqsubseteq$ is used to classify values according to their degree of confidentiality: a value of level $\bot$ is public, a value of level $\top$ is secret. The ordering also indicates the authorised direction for information flow: a flow from a value of level ℓ to a value of level ℓ' is allowed if and only if $\ell \sqsubseteq \ell'$.

Furthermore, each session participant $\mathtt{p}$ has a *reading level* for each topic φ, denoted by $\rho(\mathtt{p},\varphi)$. In a safe session, participant $\mathtt{p}$ will only be able to receive values of level $\ell \sqsubseteq \rho(\mathtt{p},\varphi)$ on topic φ. This requirement assures access control.

We also assume an irreflexive and symmetric relation of *independence* between topics: we denote

$$\text{[R-OUTPUT]}\ \frac{e \downarrow \mathsf{v}^{\ell,\varphi}}{\mathtt{q}!\lambda(e).P \xrightarrow{\mathtt{q}!\lambda(\mathsf{v}^{\ell,\varphi})} P} \qquad \text{[R-INPUT]}\ \mathtt{p}?\lambda(x).Q \xrightarrow{\mathtt{p}?\lambda(\mathsf{v}^{\ell,\varphi})} Q\{\mathsf{v}^{\ell,\varphi}/x\}$$

$$\text{[R-INT-CHOICE]}\ P \oplus Q \xrightarrow{\tau} P \qquad \text{[R-EXT-CHOICE]}\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \text{[R-STRUCT-PROC]}\ \frac{P_1' \equiv P_1 \quad P_1 \xrightarrow{\vartheta} P_2 \quad P_2 \equiv P_2'}{P_1' \xrightarrow{\vartheta} P_2'}$$

$$\text{[R-COMM]}\ \frac{P \xrightarrow{\mathtt{q}!\lambda(\mathsf{v}^{\ell,\varphi})} P' \quad Q \xrightarrow{\mathtt{p}?\lambda(\mathsf{v}^{\ell,\varphi})} Q'}{\mathtt{p} \triangleleft P \mid \mathtt{q} \triangleleft Q \xrightarrow{\mathtt{p}(\lambda,\mathsf{v}^{\ell,\varphi})\mathtt{q}} \mathtt{p} \triangleleft P' \mid \mathtt{q} \triangleleft Q'} \qquad \text{[R-TAU]}\ \frac{P \xrightarrow{\tau} P'}{\mathtt{p} \triangleleft P \xrightarrow{\tau} \mathtt{p} \triangleleft P'}$$

$$\text{[R-CONTEXT]}\ \frac{\mathscr{M} \xrightarrow{\kappa} \mathscr{M}'}{\mathscr{M} \mid \mathscr{M}'' \xrightarrow{\kappa} \mathscr{M}' \mid \mathscr{M}''} \qquad \text{[R-STRUCT-SESS]}\ \frac{\mathscr{M}_1' \equiv \mathscr{M}_1 \quad \mathscr{M}_1 \xrightarrow{\kappa} \mathscr{M}_2 \quad \mathscr{M}_2 \equiv \mathscr{M}_2'}{\mathscr{M}_1' \xrightarrow{\kappa} \mathscr{M}_2'}$$

Table 2: LTS rules for processes and sessions.

by $\varphi \curlyvee \psi$ the fact that φ and ψ are *independent* and by $\varphi \curlywedge \psi$ (defined as $\neg(\varphi \curlyvee \psi)$) the fact that φ and ψ are *correlated*. Neither of these two relations is transitive in general, as illustrated by Example 1.1, where $\curlywedge$ is the co-authorship relation between papers and $\curlyvee$ is its complement.

We say that a session is *leak-free* if, whenever a participant $\mathtt{p}$ receives a value of level ℓ on topic φ, then $\mathtt{p}$ can subsequently only send values of level $\ell' \sqsupseteq \ell$ on topics related to φ. For instance, the output of level ℓ' could be placed within an internal choice, and this choice could be resolved depending on the input of level ℓ, since this input is on a related topic. To formalise this requirement we need to look at the *traces* of multiparty sessions, ranged over by σ, σ' and defined as the sequences of actions that label a transition sequence. Formally, σ is a word on the alphabet containing τ and the messages $\mathtt{p}(\lambda, \mathsf{v}^{\ell,\varphi})\mathtt{q}$ for all participants $\mathtt{p},\mathtt{q}$, labels λ, values v, security levels ℓ and topics φ. Safety is now defined as follows, using the notion of relay trace:

Definition 3.1. *A* relay trace *is a trace of the form:*

$$\sigma \cdot \mathtt{p}(\lambda, \mathsf{v}^{\ell,\varphi})\mathtt{q} \cdot \sigma' \cdot \mathtt{q}(\lambda', \mathsf{u}^{\ell',\psi})\mathtt{r}$$

The middle participant $\mathtt{q}$ *is called the* mediator *between participants* $\mathtt{p}$ *and* $\mathtt{r}$.

Definition 3.2. *A multiparty session* $\mathscr{M}$ *is* safe *if it satisfies:*

1. Access control (AC): *whenever* $\sigma \cdot \mathtt{p}(\lambda, \mathsf{v}^{\ell,\varphi})\mathtt{q}$ *is a trace of* $\mathscr{M}$*, then* $\ell \sqsubseteq \rho(\mathtt{q}, \varphi)$*;*
2. Leak freedom (LF): *whenever* $\sigma \cdot \mathtt{p}(\lambda, \mathsf{v}^{\ell,\varphi})\mathtt{q} \cdot \sigma' \cdot \mathtt{q}(\lambda', \mathsf{u}^{\ell',\psi})\mathtt{r}$ *is a relay trace of* $\mathscr{M}$*, then either* $\ell \sqsubseteq \ell'$ *or* $\varphi \curlyvee \psi$.

For example the relay trace $\mathtt{p}(\lambda, \mathsf{true}^{\top,\varphi})\mathtt{q} \cdot \mathtt{q}(\lambda', \mathsf{false}^{\bot,\psi})\mathtt{r}$ satisfies the condition of the previous definition if $\rho(\mathtt{p}, \varphi) = \top$ and $\varphi \curlyvee \psi$. Intuitively, in spite of the "level drop" between the two messages, their sequencing is harmless because they belong to two different conversations.

Example 3.3. *The PC discussion described in Example 1.1 may be formalised as the session:*

$$\mathscr{M}_{PC} = \prod_{i \in I} \mathtt{p}_i \triangleleft P_i \quad \text{where } I = \{1, \ldots, n\}.$$

Here each participant $\mathtt{p}_i$ *represents a PC member, and* P_i *is the associated process. Let us see how the three properties discussed in Example 1.1 can be expressed in* $\mathscr{M}_{PC}$.

1. *(AC issue) Here we assume that* $\mathtt{p}_1$ *is entitled to receive a confidential value* $\mathtt{v}^{\ell,\varphi}$ *from some* $\mathtt{p}$. *Thus* $\ell \neq \bot$ *and* $\ell \sqsubseteq \rho(\mathtt{p}_1,\varphi)$. *Subsequently* $\mathtt{p}_1$ *forwards this information to* $\mathtt{p}_2$, *hence there is a relay trace of the form* $\mathtt{p}(\mathtt{v}^{\ell,\varphi})\mathtt{p}_1 \cdot \mathtt{p}_1(\mathtt{v}^{\ell,\varphi})\mathtt{p}_2$. *This trace trivially satisfies LF, and the second message satisfies AC if and only if* $\ell \sqsubseteq \rho(\mathtt{p}_2,\varphi)$. *Then, if we set* $\rho(\mathtt{p}_2,\varphi) = \bot$ *for any* $\mathtt{p}_2$ *in conflict with* φ *and* $\rho(\mathtt{p}_2,\varphi) = \top$ *for any other* $\mathtt{p}_2$, *Property 1 will be ensured by the safety of* $\mathscr{M}_{PC}$.

2. *(LF issue) Here the relay trace has the form* $\mathtt{p}(\mathtt{v}^{\ell,\varphi})\mathtt{p}_1 \cdot \mathtt{p}_1(\mathtt{v}_1^{\bot,\varphi_1})\mathtt{p}_2$, *where again* $\bot \neq \ell \sqsubseteq \rho(\mathtt{p}_1,\varphi)$. *This trace satisfies LF because* φ_1 *is independent from* φ. *The second message trivially satisfies AC because the email address* $\mathtt{v}_1^{\bot,\varphi_1}$ *has level* $\bot$ *and thus can be read by any participant.*

3. *(Combination of AC and LF) Here* $\mathtt{p}_1$ *sends to the PC Chair* $\mathtt{p}_0$ *a confidential value* $\mathtt{v}_1^{\ell,\varphi}$, *followed by a request for a public document of topic* ψ, *and then waits to receive this document from* $\mathtt{p}_0$. *The behaviour of* $\mathtt{p}_0$ *is dual for the first two steps, but then* $\mathtt{p}_0$ *asks* $\mathtt{p}_2$, *who is in conflict with paper* φ, *to fetch the document for him, before sending it back to* $\mathtt{p}_1$.

   Processes implementing the behaviour of the PC Chair and of the involved PC members are:

$$\begin{array}{lcl} P_0 & = & \mathtt{p}_1?(x).\mathtt{p}_1?(y).\mathtt{p}_2!(y).\mathtt{p}_2?(z).\mathtt{p}_1!(z).\mathbf{0} \\ P_1 & = & \mathtt{p}_0!(\mathtt{v}_1^{\ell,\varphi}).\mathtt{p}_0!(\mathtt{v}_2^{\bot,\psi}).\mathtt{p}_0?(x).\mathbf{0} \\ P_2 & = & \mathtt{p}_0?(x).\mathtt{p}_0!(\mathtt{v}_3^{\bot,\psi}).\mathbf{0} \end{array}$$

   Intuitively, the reading levels of $\mathtt{p}_0,\mathtt{p}_1$ *and* $\mathtt{p}_2$ *should be* $\rho(\mathtt{p}_0,\varphi) = \rho(\mathtt{p}_0,\psi) = \top$, $\rho(\mathtt{p}_1,\varphi) = \ell$, $\rho(\mathtt{p}_1,\psi) = \bot$, *and* $\rho(\mathtt{p}_2,\varphi) = \rho(\mathtt{p}_2,\psi) = \bot$. *Consider now the following trace of session* $\mathscr{M}_{PC}$:

$$\sigma = \mathtt{p}_1(\mathtt{v}_1^{\ell,\varphi})\mathtt{p}_0 \cdot \mathtt{p}_1(\mathtt{v}_2^{\bot,\psi})\mathtt{p}_0 \cdot \mathtt{p}_0(\mathtt{v}_2^{\bot,\psi})\mathtt{p}_2 \cdot \mathtt{p}_2(\mathtt{v}_3^{\bot,\psi})\mathtt{p}_0 \cdot \mathtt{p}_0(\mathtt{v}_3^{\bot,\psi})\mathtt{p}_1$$

   With the above reading levels, each message in trace σ *satisfies AC. Moreover, trace* σ *contains three relay traces:* $\mathtt{p}_1(\mathtt{v}_2^{\bot,\psi})\mathtt{p}_0 \cdot \mathtt{p}_0(\mathtt{v}_2^{\bot,\psi})\mathtt{p}_2$, $\mathtt{p}_0(\mathtt{v}_2^{\bot,\psi})\mathtt{p}_2 \cdot \mathtt{p}_2(\mathtt{v}_3^{\bot,\psi})\mathtt{p}_0$, *and* $\mathtt{p}_2(\mathtt{v}_3^{\bot,\psi})\mathtt{p}_0 \cdot \mathtt{p}_0(\mathtt{v}_3^{\bot,\psi})\mathtt{p}_1$, *which trivially satisfy LF since all values have level* $\bot$.

4 Type System

Our type system enriches the system of [4] with security levels and topics.

Types. *Sorts* are ranged over by S and defined by: $S \quad ::= \quad \mathtt{nat} \mid \mathtt{int} \mid \mathtt{bool} \mid \mathtt{string}$
Global types describe the whole conversation scenarios of multiparty sessions. They are generated by:

$$G \quad ::= \quad \mathtt{p} \to \mathtt{q} : \{\lambda_i(S_i^{\ell_i,\varphi_i}).G_i\}_{i\in I} \quad \mid \quad \mu\mathbf{t}.G \quad \mid \quad \mathbf{t} \quad \mid \quad \mathtt{end}$$

Session types correspond to the views of the individual participants. They can be either unions of outputs or intersections of inputs. The grammar of session types, ranged over by T, is then

$$T ::= \bigvee_{i\in I} \mathtt{q}!\lambda_i(S_i^{\ell_i,\varphi_i}).T_i \mid \bigwedge_{i\in I} \mathtt{p}?\lambda_i(S_i^{\ell_i,\varphi_i}).T_i \mid \mu\mathbf{t}.T \mid \mathbf{t} \mid \mathtt{end}$$

We require that $\lambda_i \neq \lambda_j$ with $i \neq j$ and $i,j \in I$.

We give now conditions on session types which will guarantee session safety.

Definition 4.1. *A pair of a security level* ℓ *and a topic* φ agrees *with a session type* T *(notation* $\langle \ell,\varphi\rangle \prec T$*) if* T *specifies that only values of level* $\ell' \sqsupseteq \ell$ *are sent on topics related with* φ:

$$\text{[AGR-END]}\quad \langle \ell, \varphi \rangle \prec \mathtt{end} \qquad \text{[AGR-OUT]}\quad \frac{\forall i \in I : \langle \ell, \varphi \rangle \prec T_i \text{ (either } \ell \sqsubseteq \ell'_i \text{ or } \varphi \curlyvee \psi_i)}{\langle \ell, \varphi \rangle \prec \bigvee_{i \in I} \mathtt{q}!\lambda_i(S^{\ell'_i, \psi_i}).T_i} \qquad \text{[AGR-IN]}\quad \frac{\forall i \in I : \langle \ell, \varphi \rangle \prec T_i}{\langle \ell, \varphi \rangle \prec \bigwedge_{i \in I} \mathtt{p}?\lambda_i(S_i^{\ell_i, \varphi_i}).T_i}$$

Definition 4.2. *A closed session type T is a* safe session type *if $\vdash T$ can be derived from the rules:*

$$\text{[SAFE-END]}\quad \vdash \mathtt{end} \qquad \text{[SAFE-OUT]}\quad \frac{\forall i \in I : \quad \vdash T_i \quad \ell_i \sqsubseteq \rho(\mathtt{q}, \varphi_i)}{\vdash \bigvee_{i \in I} \mathtt{q}!\lambda_i(S_i^{\ell_i, \varphi_i}).T_i} \qquad \text{[SAFE-IN]}\quad \frac{\forall i \in I : \quad \vdash T_i \quad \langle \ell_i, \varphi_i \rangle \prec T_i}{\vdash \bigwedge_{i \in I} \mathtt{p}?\lambda_i(S_i^{\ell_i, \varphi_i}).T_i}$$

The double line in the above rules means that they are *coinductive* [7, 21.1]. This is necessary since session types are recursive and under the equi-recursive approach the types in the premises can coincide with the types in the conclusion. For example $\mathtt{p}?\lambda(\mathtt{bool}^{\top,\varphi}).\mathtt{r}!\lambda'(\mathtt{bool}^{\bot,\psi}).\mathtt{end}$ is a safe type if $\rho(\mathtt{p}, \varphi) = \top$ and $\varphi \curlyvee \psi$.

We only allow safe types in the typing rules for processes and multiparty sessions. As will be established in Theorem 5.5, the conditions in rules [SAFE-OUT] and [SAFE-IN] of safe session types assure respectively access control and leak freedom, namely Properties 1 and 2 of session safety (Definition 3.2).

Typing rules. We distinguish three kinds of typing judgments. Expressions are typed by sorts with levels and topics, processes are typed by session types and multiparty sessions are typed by global types:

$$\Gamma \vdash e : S^{\ell, \varphi} \qquad \Gamma \vdash P \blacktriangleright T \qquad \mathscr{M} \blacktriangleright G$$

Here Γ is the *environment* that associates expression variables with sorts (decorated by levels and topics) and process variables with safe session types: $\Gamma ::= \emptyset \mid \Gamma, x : S^{\ell, \varphi} \mid \Gamma, X : T$.

The typing rules for expressions in Table 3 are almost standard, but for the treatment of topics. A value of level ℓ and topic φ is typed with the appropriate sort type decorated by ℓ and φ. Expressions cannot contain subexpressions of different topics. This limitation could be easily overcome by allowing sets of topics. In this way we could associate to an expression the set of topics of its subexpressions. The sets of topics would naturally build a lattice, where the order is given by subset inclusion.

$$\text{[EXP-VAR]}\quad \Gamma, x : S^{\ell, \varphi} \vdash x : S^{\ell, \varphi} \qquad \text{[EXP-VAL]}\quad \Gamma \vdash \mathtt{v}^{\ell, \varphi} : S^{\ell, \varphi} \qquad \text{[EXP-OP]}\quad \frac{\Gamma \vdash e_1 : S_1^{\ell_1, \varphi} \quad \Gamma \vdash e_2 : S_2^{\ell_2, \varphi} \quad \mathtt{op} : S_1, S_2 \to S_3}{\Gamma \vdash e_1 \,\mathtt{op}\, e_2 : S_3^{\ell_1 \sqcup \ell_2, \varphi}}$$

Table 3: Typing rules for expressions.

Processes have the expected types. Let us note that the syntax of session types only allows output processes in internal choices (typed by unions) and input processes in external choices (typed by intersections). Table 4 gives the typing rules for processes. For example, if $\rho(\mathtt{p}, \varphi) = \top$ and $\varphi \curlyvee \psi$ we can derive $\vdash \mathtt{p}?\lambda(x).\mathtt{r}!\lambda'(\mathsf{false}^{\bot,\psi}).\mathbf{0} \blacktriangleright \mathtt{p}?\lambda(\mathtt{bool}^{\top,\varphi}).\mathtt{r}!\lambda'(\mathtt{bool}^{\bot,\psi}).\mathtt{end}$, while this process cannot be typed otherwise. Notice that the process obtained by erasing topics is not typable in the system of [2], where the typing rule for input requires that the level of the input be lower than or equal to the level of the following output. Similarly, in the monitored semantics of [3], this input would raise the monitor level to $\top$ and then the monitor would produce an error when applied to the output of level $\bot$.

A session is typable when its parallel components can play as participants of a whole communication protocol or they are terminated. To formalise this we need some definitions.

$$\frac{\Gamma \vdash e : S^{\ell,\varphi} \quad \Gamma \vdash P \blacktriangleright T}{\Gamma \vdash \mathtt{q}!\lambda(e).P \blacktriangleright \mathtt{q}!\lambda(S^{\ell,\varphi}).T}\ [\text{T-OUT}] \qquad \frac{\Gamma, x : S^{\ell,\varphi} \vdash Q \blacktriangleright T}{\Gamma \vdash \mathtt{p}?\lambda(x).Q \blacktriangleright \mathtt{p}?\lambda(S^{\ell,\varphi}).T}\ [\text{T-IN}]$$

$$\frac{\Gamma \vdash P_1 \blacktriangleright T_1 \quad \Gamma \vdash P_2 \blacktriangleright T_2}{\Gamma \vdash P_1 \oplus P_2 \blacktriangleright T_1 \vee T_2}\ [\text{T-I-CHOICE}] \qquad \frac{\Gamma \vdash P_1 \blacktriangleright T_1 \quad \Gamma \vdash P_2 \blacktriangleright T_2}{\Gamma \vdash P_1 + P_2 \blacktriangleright T_1 \wedge T_2}\ [\text{T-E-CHOICE}]$$

$$\frac{\Gamma, X : T \vdash P \blacktriangleright T}{\Gamma \vdash \mu X.P \blacktriangleright T}\ [\text{T-REC}] \quad \Gamma, X : T \vdash X \blacktriangleright T\ [\text{T-VAR}] \qquad \Gamma \vdash \mathbf{0} \blacktriangleright \mathtt{end}\ [\text{T-}\mathbf{0}]$$

Table 4: Typing rules for processes.

The *subtyping* relation $\leq$ between session types as defined in Table 5 is simply the set-theoretic inclusion between intersections and unions. The double line in these rules means that subtyping is co-inductively defined.

$$[\text{SUB-END}] \quad \mathtt{end} \leqslant \mathtt{end} \qquad \frac{[\text{SUB-IN}] \quad \forall i \in I : \quad T_i \leqslant T_i'}{\bigwedge_{i \in I \cup J} \mathtt{p}?\lambda_i(S_i^{\ell_i,\varphi_i}).T_i \leqslant \bigwedge_{i \in I} \mathtt{p}?\lambda_i(S_i^{\ell_i,\varphi_i}).T_i'} \qquad \frac{[\text{SUB-OUT}] \quad \forall i \in I : \quad T_i \leqslant T_i'}{\bigvee_{i \in I} \mathtt{p}!\lambda_i(S_i^{\ell_i,\varphi_i}).T_i \leqslant \bigvee_{i \in I \cup J} \mathtt{p}!\lambda_i(S_i^{\ell_i,\varphi_i}).T_i'}$$

Table 5: Subtyping rules.

The *projection* of the global type G on participant $\mathtt{p}$, notation $G \upharpoonright \mathtt{p}$, is as usual [5], and it is reported in Table 6. We shall consider projectable global types only.

$$\mathtt{p} \to \mathtt{q} : \{\lambda_i(S_i^{\ell_i,\varphi_i}).G_i\}_{i \in I} \upharpoonright \mathtt{r} = \begin{cases} \bigvee_{i \in I} \mathtt{q}!\lambda_i(S_i^{\ell_i,\varphi_i}).G_i \upharpoonright \mathtt{r} & \text{if } \mathtt{r} = \mathtt{p}, \\ \bigwedge_{i \in I} \mathtt{p}?\lambda_i(S_i^{\ell_i,\varphi_i}).G_i \upharpoonright \mathtt{r} & \text{if } \mathtt{r} = \mathtt{q}, \\ G_i \upharpoonright \mathtt{r} & \text{if } \mathtt{r} \neq \mathtt{p}, \mathtt{r} \neq \mathtt{q} \\ & \text{and } G_i \upharpoonright \mathtt{r} = G_j \upharpoonright \mathtt{r} \text{ for all } i, j \in I. \end{cases}$$

$$\mu\mathbf{t}.G \upharpoonright \mathtt{r} = \begin{cases} G \upharpoonright \mathtt{r} & \text{if } \mathtt{r} \text{ occurs in } G, \\ \mathtt{end} & \text{otherwise.} \end{cases} \qquad \mathbf{t} \upharpoonright \mathtt{r} = \mathbf{t} \qquad \mathtt{end} \upharpoonright \mathtt{r} = \mathtt{end}$$

Table 6: Projection of global types onto participants.

We define the set $\mathtt{pt}\{G\}$ of participants of a global type G as expected:

$$\mathtt{pt}\{\mathtt{p} \to \mathtt{q} : \{\lambda_i(S_i^{\ell_i,\varphi_i}).G_i\}_{i \in I}\} = \{\mathtt{p}, \mathtt{q}\} \cup \mathtt{pt}\{G_i\}\ (i \in I)^1$$
$$\mathtt{pt}\{\mu\mathbf{t}.G\} = \mathtt{pt}\{G\} \qquad \mathtt{pt}\{\mathbf{t}\} = \emptyset \qquad \mathtt{pt}\{\mathtt{end}\} = \emptyset$$

We can now explain the typing rule for sessions:

$$\frac{\forall i \in \{1, \ldots, n\} : \quad \vdash P_i \blacktriangleright T_i \qquad T_i \leq G \upharpoonright \mathtt{p}_i \qquad \mathtt{pt}\{G\} \subseteq \{\mathtt{p}_1, \ldots, \mathtt{p}_n\}}{\mathtt{p}_1 \triangleleft P_1 \mid \ldots \mid \mathtt{p}_n \triangleleft P_n \blacktriangleright G}\ [\text{T-SESS}]$$

Note that all $\mathtt{p}_i$ must be distinct, since the premise assumes $\{\mathtt{p}_1, \ldots, \mathtt{p}_n\}$ to be a set of n elements. The condition $T_i \leq G \upharpoonright \mathtt{p}_i$ assures that the type of the process paired with participant $\mathtt{p}_i$ is "better" than the projection of the global type G on $\mathtt{p}_i$. The inclusion of $\mathtt{pt}\{G\}$ in the set $\{\mathtt{p}_1, \ldots, \mathtt{p}_n\}$ allows sessions containing $\mathtt{p} \triangleleft \mathbf{0}$ to be typed, a property needed to assure invariance of types under structural congruence.

[1] The projectability of G assures $\mathtt{pt}\{G_i\} = \mathtt{pt}\{G_j\}$ for all $i, j \in I$.

Example 4.3. *The communication protocol described in Examples 1.1 and 3.3, Item 3, can be formalised (omitting labels) by the global type:*

$$\mathtt{p}_1 \to \mathtt{p}_0 : \mathtt{str}^{\ell,\varphi}.\mathtt{p}_1 \to \mathtt{p}_0 : \mathtt{str}^{\perp,\psi}.\mathtt{p}_0 \to \mathtt{p}_2 : \mathtt{str}^{\perp,\psi}.\mathtt{p}_2 \to \mathtt{p}_0 : \mathtt{str}^{\perp,\psi}.\mathtt{p}_0 \to \mathtt{p}_1 : \mathtt{str}^{\perp,\psi}.\mathtt{end}$$

where `str` *is short for* `string`.

The session type of the PC chair $\mathtt{p}_0$ *is then:*

$$\mathtt{p}_1?\mathtt{str}^{\ell,\varphi}.\mathtt{p}_1?\mathtt{str}^{\perp,\psi}.\mathtt{p}_2!\mathtt{str}^{\perp,\psi}.\mathtt{p}_2?\mathtt{str}^{\perp,\psi}.\mathtt{p}_1!\mathtt{str}^{\perp,\psi}.\mathtt{end}$$

This type is safe, since φ *and* ψ *are unrelated. In fact we can check that*

$$\langle \ell, \varphi \rangle \prec \mathtt{p}_1?\mathtt{str}^{\perp,\psi}.\mathtt{p}_2!\mathtt{str}^{\perp,\psi}.\mathtt{p}_2?\mathtt{str}^{\perp,\psi}.\mathtt{p}_1!\mathtt{str}^{\perp,\psi}.\mathtt{end}.$$

5 Main Properties

The basic soundness property of the typing system w.r.t. operational semantics is subject reduction. As usual with types expressing communications, the reduction of sessions "consumes" the types. This consumption can be formalised by means of a reduction. In our system we need to reduce both session types and global types.

The reduction of session types is the smallest pre-order relation closed under the rules:

$$T \vee T' \Longrightarrow T \qquad \mathtt{p}!\lambda(S^{\ell,\varphi}).T \Longrightarrow T \qquad \textstyle\bigwedge_{i\in I} \mathtt{p}?\lambda_i(S_i^{\ell_i,\varphi_i}).T_i \Longrightarrow T_i$$

These rules mimic respectively internal choice, output and external choice among inputs.

The reduction of global types is the smallest pre-order relation closed under the rule:

$$G \Longrightarrow G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}$$

where $G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}$ is the global type obtained from G by executing the communication $\mathtt{p} \xrightarrow{\lambda} \mathtt{q}$. We dub $G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}$ the *residual* after the communication $\mathtt{p} \xrightarrow{\lambda} \mathtt{q}$ in the global type G, whose definition is given in Table 7. Notice that $G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}$ is defined only if $\mathtt{p} \xrightarrow{\lambda} \mathtt{q}$ occurs in G, since both $\mathtt{end} \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}$ and $\mathbf{t} \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}$ are undefined. For example, if $G = \mathtt{r} \to s : \lambda'(\mathtt{nat}^{\perp,\varphi}).\mathtt{p} \to \mathtt{q} : \lambda(\mathtt{bool}^{\top,\psi})$, then

$$G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q} = \mathtt{r} \to s : \lambda'(\mathtt{nat}^{\perp,\varphi}).$$

The reduction rule for global types is more involved than that for session types, since the global types do not prescribe an order on communications between disjoint pairs of participants.

We can now show that session reduction transforms the global type of a session into its residual, and the session types of the processes into their reducts. Besides substitution and inversion lemmas, the proof of subject reduction is based on the relations between subtyping, projection and erasure of communications.

$$(\mathtt{r} \to \mathtt{s} : \{\lambda_i(S_i^{\ell_i,\varphi_i}).G_i\}_{i\in I}) \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q} = \begin{cases} G_{i_0} & \text{if } \mathtt{r} = \mathtt{p}, \\ & \mathtt{s} = \mathtt{q}, \\ & \lambda_{i_0} = \lambda \quad i_0 \in I \\ \mathtt{r} \to \mathtt{s} : \{\lambda_i(S_i^{\ell_i,\varphi_i}).G_i \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}\}_{i\in I} & \text{otherwise} \end{cases}$$

$$(\mu \mathbf{t}.G) \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q} = \mu \mathbf{t}.G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}$$

Table 7: Residual after a communication.

Lemma 5.1. *If* $\Gamma \vdash e : S^{\ell,\varphi}$ *and* $e \downarrow \mathsf{v}^{\ell,\varphi}$ *and* $\Gamma, x : S^{\ell,\varphi} \vdash P \blacktriangleright T$, *then* $\Gamma \vdash P\{\mathsf{v}^{\ell,\varphi}/x\} \blacktriangleright T$.

Proof. Standard. □

Lemma 5.2.

1. *If* $\Gamma \vdash \mathtt{p}!\lambda(e).P \blacktriangleright T$, *then* $T = \mathtt{p}!\lambda(S^{\ell,\varphi}).T'$ *and* $\Gamma \vdash e : S^{\ell,\varphi}$ *and* $\Gamma \vdash P \blacktriangleright T'$.
2. *If* $\Gamma \vdash \mathtt{p}?\lambda(x).P \blacktriangleright T$, *then* $T = \mathtt{p}?\lambda(S^{\ell,\varphi}).T'$ *and* $\Gamma, x : S^{\ell,\varphi} \vdash P \blacktriangleright T'$.
3. *If* $\Gamma \vdash P \oplus Q \blacktriangleright T$, *then* $T = T_1 \vee T_2$ *and* $\Gamma \vdash P \blacktriangleright T_1$ *and* $\Gamma \vdash Q \blacktriangleright T_2$.
4. *If* $\Gamma \vdash P + Q \blacktriangleright T$, *then* $T = T_1 \wedge T_2$ *and* $\Gamma \vdash P \blacktriangleright T_1$ *and* $\Gamma \vdash Q \blacktriangleright T_2$.
5. *If* $\mathtt{p}_1 \triangleleft P_1 \mid \ldots \mid \mathtt{p}_n \triangleleft P_n \blacktriangleright G$, *then* $\vdash P_i \blacktriangleright T_i$ *and* $T_i \leq G \upharpoonright \mathtt{p}_i$ *for* $1 \leq i \leq n$ *and* $\mathtt{pt}\{G\} \subseteq \{\mathtt{p}_1, \ldots, \mathtt{p}_n\}$.

Proof. By observing that the type assignment system for processes and multiparty sessions is syntax directed. □

Lemma 5.3. *If* $\mathtt{q}!\lambda(S^{\ell,\varphi}).T \leq G \upharpoonright \mathtt{p}$ *and* $\mathtt{p}?\lambda(S^{\ell,\varphi}).T' \wedge T'' \leq G \upharpoonright \mathtt{q}$, *then* $T \leq (G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{p}$ *and* $T' \leq (G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{q}$. *Moreover* $G \upharpoonright \mathtt{r} = (G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{r}$ *for* $\mathtt{r} \neq \mathtt{p}$, $\mathtt{r} \neq \mathtt{q}$.

Proof. By induction on G and by cases on the definition of $G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}$. Notice that G can only be $\mathtt{s}_1 \to \mathtt{s}_2 : \{\lambda_i(S_i^{\ell_i,\varphi_i}).G_i\}_{i\in I}$ with either $\mathtt{s}_1 = \mathtt{p}$ and $\mathtt{s}_2 = \mathtt{q}$ or $\{\mathtt{s}_1, \mathtt{s}_2\} \cap \{\mathtt{p}, \mathtt{q}\} = \emptyset$, since otherwise the types in the statement of the lemma could not be subtypes of the given projections of G.

If $G = \mathtt{p} \to \mathtt{q} : \{\lambda_i(S_i^{\ell_i,\varphi_i}).G_i\}_{i\in I}$, then $G \upharpoonright \mathtt{p} = \bigvee_{i\in I} \mathtt{q}!\lambda_i(S_i^{\ell_i,\varphi_i}).G_i \upharpoonright \mathtt{p}$ and $G \upharpoonright \mathtt{q} = \bigwedge_{i\in I} \mathtt{p}?\lambda_i(S_i^{\ell_i,\varphi_i}).G_i \upharpoonright \mathtt{q}$. From $\mathtt{q}!\lambda(S^{\ell,\varphi}).T \leq \bigvee_{i\in I} \mathtt{q}!\lambda_i(S_i^{\ell_i,\varphi_i}).G_i \upharpoonright \mathtt{p}$ we get $\lambda = \lambda_{i_0}$ and $T \leq G_{i_0} \upharpoonright \mathtt{p}$ for some $i_0 \in I$. From $\mathtt{p}?\lambda(S^{\ell,\varphi}).T' \wedge T'' \leq \bigwedge_{i\in I} \mathtt{p}?\lambda_i(S_i^{\ell_i,\varphi_i}).G_i \upharpoonright \mathtt{q}$ and $\lambda = \lambda_{i_0}$ we get $T' \leq G_{i_0} \upharpoonright \mathtt{q}$. This implies

$$T \leq (G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{p} \text{ and } T' \leq (G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{q},$$

since $(G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{p} = G_{i_0} \upharpoonright \mathtt{p}$ and $(G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{q} = G_{i_0} \upharpoonright \mathtt{q}$. If $\mathtt{r} \neq \mathtt{p}$, $\mathtt{r} \neq \mathtt{q}$, then by definition of projection $G \upharpoonright \mathtt{r} = G_{i_0} \upharpoonright \mathtt{r}$ for an arbitrary $i_0 \in I$, and then $G \upharpoonright \mathtt{r} = (G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{r}$ by definition of residual.

If $G = \mathtt{s}_1 \to \mathtt{s}_2 : \{\lambda_i(S_i^{\ell_i,\varphi_i}).G_i\}_{i\in I}$ and $\{\mathtt{s}_1, \mathtt{s}_2\} \cap \{\mathtt{p}, \mathtt{q}\} = \emptyset$, then $G \upharpoonright \mathtt{p} = G_{i_0} \upharpoonright \mathtt{p}$ and $G \upharpoonright \mathtt{q} = G_{i_0} \upharpoonright \mathtt{q}$ for an arbitrary $i_0 \in I$. By definition of residual

$$G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q} = \mathtt{s}_1 \to \mathtt{s}_2 : \{\lambda_i(S_i^{\ell_i,\varphi_i}).G_i \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}\}_{i\in I},$$

which implies $(G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{p} = (G_{i_0} \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{p}$ and $(G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{q} = (G_{i_0} \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{q}$.
Notice that the choice of i_0 does not modify the projection, by definition of projectability. We get $\mathtt{q}!\lambda(S^{\ell,\varphi}).T \leq G_{i_0} \upharpoonright \mathtt{p}$ and $\mathtt{p}?\lambda(S^{\ell,\varphi}).T' \wedge T'' \leq G_{i_0} \upharpoonright \mathtt{q}$, which imply by induction $T \leq (G_{i_0} \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{p}$ and $T' \leq (G_{i_0} \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{q}$.

Let $\mathtt{r} \neq \mathtt{p}$, $\mathtt{r} \neq \mathtt{q}$.
If $\mathtt{r} = \mathtt{s}_1$, then $G \upharpoonright \mathtt{r} = \bigvee_{i\in I} \mathtt{s}_2!\lambda_i(S_i^{\ell_i,\varphi_i}).G_i \upharpoonright \mathtt{r}$ and

$$(G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{r} = \bigvee_{i\in I} \mathtt{s}_2!\lambda_i(S_i^{\ell_i,\varphi_i}).(G_i \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{r},$$

so we may conclude, since by induction $G_i \upharpoonright \mathtt{r} = (G_i \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{r}$ for all $i \in I$.
If $\mathtt{r} = \mathtt{s}_2$, then $G \upharpoonright \mathtt{r} = \bigwedge_{i\in I} \mathtt{s}_1?\lambda_i(S_i^{\ell_i,\varphi_i}).G_i \upharpoonright \mathtt{r}$ and

$$(G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{r} = \bigwedge_{i\in I} \mathtt{s}_1?\lambda_i(S_i^{\ell_i,\varphi_i}).(G_i \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{r},$$

so we may conclude using induction as in the previous case.
If $\mathtt{r} \notin \{\mathtt{s}_1, \mathtt{s}_2\}$, then $G \upharpoonright \mathtt{r} = G_{i_0} \upharpoonright \mathtt{r}$ and $(G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{r} = (G_{i_0} \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{r}$ for an arbitrary $i_0 \in I$. Again, we can conclude using induction. □

Theorem 5.4. (Subject reduction) *If* $\mathtt{p} \triangleleft P \mid \mathscr{M} \xrightarrow{\kappa} \mathtt{p} \triangleleft P' \mid \mathscr{M}'$, $\mathtt{p} \triangleleft P \mid \mathscr{M} \blacktriangleright G$ *and* $\vdash P \blacktriangleright T$, *then:*

1. $\mathtt{p} \triangleleft P' \mid \mathscr{M}' \blacktriangleright G'$ *for some* G' *such that* $G \Longrightarrow^* G'$;
2. $\vdash P' \blacktriangleright T'$ *for some* T' *such that* $T \Longrightarrow^* T'$.

Proof. We only consider the more interesting reduction, i.e., when P is reduced. We distinguish three cases according to the shape of κ.

Case $\kappa = \tau$*:* then $P \equiv P_1 \oplus P_2$ and $P' \equiv P_1$ and $\mathscr{M}' \equiv \mathscr{M}$. By Lemma 5.2(5) and (3) $T \leq G \upharpoonright \mathtt{p}$ and $T = T_1 \vee T_2$ and $\vdash P_1 \blacktriangleright T_1$. We can then choose $G' = G$ and $T' = T_1$.

Case $\kappa = \mathtt{p}(\lambda, \mathtt{v}^{\ell,\varphi})\mathtt{q}$*:* then $P \equiv \mathtt{q}!\lambda(e).P'$ and $\mathscr{M} \equiv \mathtt{q} \triangleleft \mathtt{p}?\lambda(x).Q_1 + Q_2 \mid \mathscr{M}''$ and

$$\mathscr{M}' \equiv \mathtt{q} \triangleleft Q_1\{\mathtt{v}^{\ell,\varphi}/x\} \mid \mathscr{M}'',$$

where $e \downarrow \mathtt{v}^{\ell,\varphi}$. By Lemma 5.2(5) and (1) $T \leq G \upharpoonright \mathtt{p}$ and $T = \mathtt{q}!\lambda(S^{\ell,\varphi}).T'$ and $\vdash e : S^{\ell,\varphi}$ and $\vdash P' \blacktriangleright T'$. By Lemma 5.2(5) and (4) and (2) $T_1 \wedge T_2 \leq G \upharpoonright \mathtt{q}$ and $\vdash \mathtt{p}?\lambda(x).Q_1 \blacktriangleright T_1$ and $\vdash Q_2 \blacktriangleright T_2$ and $T_1 = \mathtt{p}?\lambda(S_1^{\ell',\psi}).T_1'$ and $x : S_1^{\ell',\psi} \vdash Q_1 \blacktriangleright T_1'$. From $\mathtt{q}!\lambda(S^{\ell,\varphi}).T' \leq G \upharpoonright \mathtt{p}$ and $\mathtt{p}?\lambda(S_1^{\ell',\psi}).T_1' \wedge T_2 \leq G \upharpoonright \mathtt{q}$ we get $S = S_1$ and $\ell = \ell'$ and $\varphi = \psi$. By Lemma 5.1 $\vdash e : S^{\ell,\varphi}$ and $x : S^{\ell,\varphi} \vdash Q_1 \blacktriangleright T_1'$ imply $\vdash Q_1\{\mathtt{v}^{\ell,\varphi}/x\} \blacktriangleright T_1'$. Then we choose $G' = G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}$, since Lemma 5.3 gives $T' \leq (G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{p}$ and $T_1' \leq (G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{q}$ and the same projections for all other participants of G.

Case $\kappa = \mathtt{q}(\lambda, \mathtt{v}^{\ell,\varphi})\mathtt{p}$*:* then $P \equiv \mathtt{q}?\lambda(x).P_1 + P_2$ and $\mathscr{M} \equiv \mathtt{q} \triangleleft \mathtt{p}!\lambda(e).Q \mid \mathscr{M}''$ and $P' = P_1\{\mathtt{v}^{\ell,\varphi}/x\}$ and $\mathscr{M}' \equiv \mathtt{q} \triangleleft Q \mid \mathscr{M}''$, where $e \downarrow \mathtt{v}^{\ell,\varphi}$. By Lemma 5.2(5) and (4) and (2) $T = T_1 \wedge T_2 \leq G \upharpoonright \mathtt{p}$ and $\vdash \mathtt{q}?\lambda(x).P_1 \blacktriangleright T_1$ and $\vdash P_2 \blacktriangleright T_2$ and $T_1 = \mathtt{q}?\lambda(S^{\ell,\varphi}).T'$ and $x : S^{\ell,\varphi} \vdash P_1 \blacktriangleright T'$. By Lemma 5.2(5) and (1) $T_3 \leq G \upharpoonright \mathtt{q}$ and $\vdash \mathtt{p}!\lambda(e).Q \blacktriangleright T_3$ and $T_3 = \mathtt{p}!\lambda(S_1^{\ell',\psi}).T_3'$ and $\vdash e : S_1^{\ell',\psi}$ and $\vdash Q \blacktriangleright T_3'$. From $\mathtt{q}?\lambda(S^{\ell,\varphi}).T' \wedge T_2 \leq G \upharpoonright \mathtt{p}$ and $\mathtt{p}!\lambda(S_1^{\ell',\psi}).T_3' \leq G \upharpoonright \mathtt{q}$ we get $S = S_1$ and $\ell = \ell'$ and $\varphi = \psi$. By Lemma 5.1 $\vdash e : S^{\ell,\varphi}$ and $x : S^{\ell,\varphi} \vdash P_1 \blacktriangleright T'$ imply $\vdash P_1\{\mathtt{v}^{\ell,\varphi}/x\} \blacktriangleright T'$. Then we take $G' = G \setminus \mathtt{q} \xrightarrow{\lambda} \mathtt{p}$, since Lemma 5.3 gives $T' \leq (G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{p}$ and $T_3' \leq (G \setminus \mathtt{p} \xrightarrow{\lambda} \mathtt{q}) \upharpoonright \mathtt{q}$ and the same projections for all other participants of G. □

We may now prove our main result:

Theorem 5.5. (Soundness) *If* $\mathscr{M}$ *is typable, then* $\mathscr{M}$ *is safe.*

Proof. Suppose that $\mathscr{M}$ is safely typed. If $\mathscr{M}$ generates the trace $\sigma \cdot \mathtt{p}(\lambda, \mathtt{v}^{\ell,\varphi})\mathtt{q}$, then

$$\mathscr{M} \xrightarrow{\quad\sigma\quad} \mathtt{p} \triangleleft P \mid \mathtt{q} \triangleleft Q \mid \mathscr{M}' \xrightarrow{\mathtt{p}(\lambda,\mathtt{v}^{\ell,\varphi})\mathtt{q}} \mathtt{p} \triangleleft P' \mid \mathtt{q} \triangleleft Q' \mid \mathscr{M}'.$$

From $\mathtt{p} \triangleleft P \mid \mathtt{q} \triangleleft Q \mid \mathscr{M}' \xrightarrow{\mathtt{p}(\lambda,\mathtt{v}^{\ell,\varphi})\mathtt{q}} \mathtt{p} \triangleleft P' \mid \mathtt{q} \triangleleft Q' \mid \mathscr{M}'$ we get that $P \equiv \mathtt{q}!\lambda(e).P'$ for some e such that $e \downarrow \mathtt{v}^{\ell,\varphi}$, and $Q \equiv \mathtt{p}?\lambda(x).Q_1 + Q_2$. By Lemma 5.2(5), there are types T_P and T_Q such that $\vdash P \blacktriangleright T_P$ and $\vdash Q \blacktriangleright T_Q$. By Lemma 5.2(1), T_P must be of the form $T_P = \mathtt{q}!\lambda(S^{\ell,\varphi}).T_P'$. Then the safety of $\mathtt{q}!\lambda(S^{\ell,\varphi}).T_P'$ (more specifically, the premise of Rule [SAFE-OUT]) implies that $\ell \sqsubseteq \rho(\mathtt{q}, \varphi)$. This concludes the proof of Property 1 of session safety (AC).

Suppose now that the above computation continues as follows:

$$\mathtt{p} \triangleleft P' \mid \mathtt{q} \triangleleft Q' \mid \mathscr{M}' \xrightarrow{\quad\sigma'\quad} \mathtt{q} \triangleleft Q_2 \mid \mathscr{M}'' \xrightarrow{\mathtt{q}(\lambda',\mathtt{u}^{\ell',\psi})\mathtt{r}} \mathtt{q} \triangleleft Q''' \mid \mathscr{M}'''$$

namely, that the trace $\sigma \cdot \mathtt{p}(\lambda, \mathtt{v}^{\ell,\varphi})\mathtt{q}$ is extended to the relay trace $\sigma \cdot \mathtt{p}(\lambda, \mathtt{v}^{\ell,\varphi})\mathtt{q} \cdot \sigma' \cdot \mathtt{q}(\lambda', \mathtt{u}^{\ell',\psi})\mathtt{r}$. From $Q \equiv \mathtt{p}?\lambda(x).Q_1 + Q_2$, by Lemma 5.2(4) we get $T_Q = T_1 \wedge T_2$ and $\vdash \mathtt{p}?\lambda(x).Q_1 \blacktriangleright T_1$. By applying now Lemma 5.2(2), we obtain $T_1 = \mathtt{p}?\lambda(S^{\ell,\varphi}).T_1'$ and $x : S^{\ell,\varphi} \vdash Q_1 \blacktriangleright T_1'$. From this, since $Q' = Q_1\{\mathtt{v}^{\ell,\varphi}/x\}$, we infer $\vdash Q' \blacktriangleright T_1'$. By Theorem 5.4, $\vdash Q' \blacktriangleright T_1'$ implies $\vdash Q'' \blacktriangleright T_1''$ for some T_1'' such that $T_1' \Longrightarrow T_1''$. Now, since $\mathtt{q} \triangleleft Q'' \mid \mathscr{M}'' \xrightarrow{\mathtt{q}(\lambda', \mathtt{u}^{\ell',\psi})\mathtt{r}} \mathtt{q} \triangleleft Q''' \mid \mathscr{M}'''$, we have $Q'' \equiv \mathtt{r}!\lambda'(e').Q'''$ for some e' such that $e' \downarrow \mathtt{u}^{\ell',\psi}$. By Lemma 5.2(1), $T_1'' = \mathtt{r}!\lambda'(S_1^{\ell',\psi}).T_1'''$. Now, the safety of $T_1 = \mathtt{p}?\lambda(S^{\ell,\varphi}).T_1'$ (and more specifically, the premise of Rule [SAFE-IN]) implies that $\langle \ell, \varphi \rangle \prec T_1'$ and therefore also $\langle \ell, \varphi \rangle \prec T_1'' = \mathtt{r}!\lambda'(S_1^{\ell',\psi}).T_1'''$, since T_1'' is obtained by reducing T_1' (and therefore T_1'' is a subterm of T_1'). Then $\ell \sqsubseteq \ell'$ or $\varphi \curlyvee \psi$ by definition of agreement (Rule [AGR-OUT]). This concludes the proof of Property 2 of session safety (LF). □

6 Related and Future Work

We introduced the notion of topic as a way to relax security type systems for session calculi. We focussed on multiparty rather than binary sessions, as security issues appear to be less relevant for binary sessions. Indeed, binary sessions may often be viewed as client-server interactions, where one can assume that the client chooses the server (and thus to some extent trusts it) and that the server is protected against malicious clients. On the other hand, in a multiparty session the parties are symmetric peers which may not know each other and thus require to be protected against each other.

The first multiparty session calculus with synchronous communication was presented in [**?**]. Here we considered an enrichment of the calculus of [4] with security and types. The base calculus is admittedly very simple, as it cannot describe parallel and interleaved sessions, and its type system only allows internal choices among outputs and external choices among inputs. Our version is even simpler than that of [4] since the syntax does not include the conditional construct. The advantage of this minimal setting is that the safety property, which covers both access control and leak freedom, enjoys a particularly simple definition. In particular, leak freedom amounts to a condition on *mediators*, which are participants acting as a bridge between a sender and a receiver. This condition says that after receiving high information by the sender on some topic, the mediator should not send low information to the receiver on a related topic.

It can be argued that topics are orthogonal to structured communication features, and could therefore be studied in a more general setting. However, within a structured communication the set of topics is delimited a priori, as specified by the global type, so the notion becomes more effective.

One further issue is that of expressiveness of topics. One may wonder whether the use of topics could be simulated by using other ingredients of our calculus, such as security levels, labels and base types. Clearly, the independence of topics in the two end messages of a relay trace cannot be represented by the incomparability of their security levels: a safe relay trace $m_1 \cdot \sigma \cdot m_2$ where m_1 and m_2 have unrelated topics could not be mimicked by the same trace with incomparable security levels for m_1 and m_2, since the latter is insecure in a classical LF approach. As for labels, they are meant to represent different options in the choice operators, so they are conceptually quite different from topics.

Related work. Compared to previous work on security-enriched multiparty session calculi [2, 3], our definition of leak freedom is more permissive in two respects:

1. A sequence of messages directed to the same participant is always allowed. In the calculi of [2, 3], where deadlocks could arise, it was necessary to prevent any low communication after a high communication (because the mere fact that the high communication could fail to occur would cause a leak). For instance the trace (omitting labels) $\mathtt{p}(v^{\top,\varphi})\mathtt{q} \cdot \mathtt{p}'(u^{\bot,\varphi})\mathtt{q}$ was rejected in those calculi, while it is allowed in the present one, which is deadlock-free. In our case it is only the

content of a message that can be leaked, and therefore it is enough to focus on relay sequences made of a message *to* a participant, followed by a message *from* the same participant.

2. Thanks to the introduction of topics, the standard leak-freedom requirement can be relaxed also on relay sequences, by forbidding only downward flows between messages on correlated topics.

One could see the use of topics as a way of implementing declassification (see [**?**] for a survey). For instance, a relay trace whose end messages carry values $v_1^{\top,\varphi_1}$ and $v_2^{\bot,\varphi_2}$ with independent topics φ_1 and φ_2 could be interpreted as the application of a *trusted function* (such as encryption [**?**]) to transform a secret value v_1 into a public value v_2.

Future work. We intend to explore further the relationship between topics and declassification. Also, inspired by [6], we plan to enrich the present calculus by allowing levels and topics to depend on exchanged values. Indeed, it seems reasonable to expect that a server should conform the levels and topics of its messages to its different kinds of clients. For example an ATM should receive credit card numbers with personalised topics.

Acknowledgments. We are grateful to the anonymous reviewers for their useful remarks.

References

[1] M. Bartoletti, I. Castellani, P.-M. Deniélou, M. Dezani-Ciancaglini, S. Ghilezan, J. Pantovic, J. A. Pérez, P. Thiemann, B. Toninho, and H. T. Vieira. Combining Behavioural Types with Security Analysis. *Journal of Logical and Algebraic Methods in Programming*, 84(6):763 – 780, 2015. Special Issue on Open Problems in Concurrency Theory.

[2] S. Capecchi, I. Castellani, and M. Dezani-Ciancaglini. Typing Access Control and Secure Information Flow in Sessions. *Information and Computation*, 238:68–105, 2014.

[3] S. Capecchi, I. Castellani, and M. Dezani-Ciancaglini. Information Flow Safety in Multiparty Sessions. *Mathematical Structures in Computer Science*, 2016. to appear.

[4] M. Dezani-Ciancaglini, S. Ghilezan, J. P. Svetlana Jaksic, and N. Yoshida. Precise Subtyping for Synchronous Multiparty Sessions. In *Proc. PLACES*, EPTCS, 2016. To appear.

[5] K. Honda, N. Yoshida, and M. Carbone. Multiparty Asynchronous Session Types. In *Proc. POPL'08*, pages 273–284. ACM Press, 2008.

[6] L. Lourenço and L. Caires. Dependent Information Flow Types. In *Proc. POPL'15*, pages 317–328. ACM Press, 2015.

[7] B. C. Pierce. *Types and Programming Languages*. MIT Press, 2002.