Data-driven Adaptation for Smart Sessions

Viviana Bono[a], Mario Coppo[a], Mariangiola Dezani-Ciancaglini[a], Betti Venneri[b]

[a]*Dipartimento di Informatica, Università di Torino, Italy*
[b]*Dipartimento di Statistica, Informatica, Applicazioni, Università di Firenze, Italy*

Abstract

This paper presents a formal framework of self-adaptation for multiparty sessions. The adaptation function contains the dynamic evolution policy, by prescribing how the session needs to reconfigure itself, based on critical changes in global data. A global type prescribes the overall communication choreography; its projections onto participants generate the monitors, which set-up the communication protocols. The key technical novelty of the calculus is the parallel operator for building global types and monitors, which allows the adaptation procedure to be rather flexible. The smart session is able to minimise its adaptation, by partially reconfiguring some of the communications and leaving all others unchanged, in case a part of the whole behaviour only needs to be modified. Furthermore, new participants can be added and/or some of the old participants can be removed. As a main result, we prove that this adaptation mechanism is safe, in order to guarantee that the communications will continue to evolve in a correct way after reconfiguration.

1. Introduction

Modern, emerging software intensive systems, operating in different scenarios within highly dynamic environments, brought out the need of novel approaches to master the extreme complexity of component interactions and their sensitivity to environmental changes at runtime. In particular, the property of self-adaptation emerged as a key requirement in a variety of application areas, including the internet of things and the cyber-physical systems, where a system must be able to autonomously reconfigure its behaviour dynamically, in response to changing conditions and unexpected circumstances in the current environment.

In this paper we propose a model of *data-driven self-adaptivity* for *multiparty sessions* [1], by focusing on the formal property of correctness to guarantee that dynamic adaptations preserve the safety of interactions among the session participants.

Typical scenarios that can be fruitfully represented by our approach are those characterised by the following features:

- a community, established for a common mission, has many distributed participants which interact with each other according to an overall operational plan,
- a common set of data represent the global environment and are shared by all the participants,

- unforeseen events are dynamically revealed by crucial values in the global data, thus requiring an adaptation,
- the community promptly reacts to those critical events by reconfiguring itself at minimal cost, so reusing what has to be left unchanged while adding new behaviours or modifying few interactions only.

Our calculus comprises four active parties: *adaptation functions*, *global types*, *monitors*, and *processes*.

The adaptation function represents the overall operational plan to deal with exceptional events, revealed by critical values in global data. It contains the dynamic evolution policy, by prescribing how the session needs to reconfigure itself based on changes of data.

A global type represents the overall communication choreography [2]; its projections onto participants generate the monitors, which set-up their communication protocols. The association of a monitor with a compliant process, dubbed *monitored process*, incarnates a participant, where the process provides an implementation to the monitoring protocol.

Global data, dubbed *control data* as in [3], represent the dynamic environment, inside and through which the system components interact. They are accessed and modified by the participants. For example, participants update them by progress reports on their correct functioning and by valuable data for the whole system. Thus crucial changes in global data identify an internal criticality or an external emergency such as, for instance, the sudden inability of a participant or a glut of requests for a specific product by the market in a production context. The adaptation strategy is owned by the adaptation function and control data. As long as the control data do not yield critical values, the adaptation function is undefined. Thus participants communicate and read/write data according to their monitors and codes, without performing any reconfiguration. Instead, in case of critical data values, the adaptation function generates a new global type. As a consequence, the session updates itself autonomously, by reconfiguring its participants according to the new global type. The proactive role of data in triggering the adaptation is the hallmark of our *data-driven* approach.

Monitored processes have local data used in evaluating conditionals and exchanging informations. Our model leaves implicit these local data.

The key technical novelty of this calculus with respect to the current literature is the use of parallel composition for building global types, monitors and processes. It allows the adaptation procedure to be performed in a rather flexible and smooth way. Indeed, using monitors in parallel, the session is *smart* enough to minimise its self-reconfiguration, by updating only some of the communications while leaving the other ones as before, in case a part of the whole behaviour needs to be changed. Thus the adaptation of some participants can be transparent to the other ones, also in presence of communications between them, while new participants can be added and/or some of the old participants can be removed.

To exemplify our approach, let us consider a manufacturing Company. The adaptation function can be thought as a control software, possibly including data mining algorithms, which implements the company business strategy when applied to control

data. The session includes an Italian factory (IF), an Italian supplier (IS) and a store (Ro) in Rome. Control data hold information about factories, suppliers and stores. The supplier interacts in parallel with the factory and the store about marketing data (number of item requested, delivery date).

We denote by $\mathbb{G}$ a global type consisting of the parallel composition of single threaded global types G. The interactions are described by the following (single threaded) global types:

$$\mathsf{G}_1 = \mu \mathbf{t}.\mathsf{IS} \to \mathsf{IF} : SF(\text{Item,Amount}).\mathsf{IF} \to \mathsf{IS} : FS(\text{DeliveryDate}).\mathbf{t}$$

$$\mathsf{G}_2 = \mu \mathbf{t}.\mathsf{Ro} \to \mathsf{IS} : RS(\text{Item,Amount}).\mathsf{IS} \to \mathsf{Ro} : SR(\text{DeliveryDate}).\mathbf{t}$$

where SF, FS, RS, SR are *labels* and DeliveryDate, Item, Amount are sorts. In this paper labels are used both for identifying the branches in presence of multiple choices and to connect monitors and processes as in [4].

The global type of the system is then given by $\mathbb{G}_0 = \mathsf{G}_1 \mid \mathsf{G}_2$. We observe that we allow the parallel composition of global types with the same participants.

The monitors of the participants are obtained as the parallel composition of the projections of these global types. For instance, the monitor of participant IS is:

$$\mu \mathbf{t}.\mathsf{IF}!SF(\text{Item, Amount}).\mathsf{IF}?FS(\text{Date}).\mathbf{t} \mid \mu \mathbf{t}.\mathsf{Ro}?RS(\text{Item, Amount}).\mathsf{Ro}!SR(\text{Date}).\mathbf{t}$$

and a possible process code is[1]:

$$\mu X.!SF(\mathsf{item},\mathsf{amount}).?FS(date).X \mid \mu X.?RS(item,amount).!SR(\mathsf{date}).X$$

where ! represents output and ? represents input.

Assume that the control data register the opening of a new store (Lo) in London, which must receive its items from the Italian seller.

A new global type is set up by putting in parallel the current global type $\mathbb{G}_0$ with the global type G, produced by the application of the adaptation function to the control data:

$$\mathsf{G} = \mu \mathbf{t}.\mathsf{Lo} \to \mathsf{IS} : LS(\text{Item,Amount}).\mathsf{IS} \to \mathsf{Lo} : SL(\text{DeliveryDate}).\mathbf{t}$$

The global type G adds the participant Lo to the conversation and modifies the participant IS by adding the monitor

$$\mu \mathbf{t}.\mathsf{Lo}?LS(\text{Item, Amount}).\mathsf{Lo}!SL(\text{Date}).\mathbf{t}$$

and the process

$$\mu X.?LS(item,amount).!SL(\mathsf{date}).X$$

[1] Sorts are written with upper case initials, expression *variables* and values with lower case initials, but different fonts.

in parallel with his previous ones.

Communications are assumed to be synchronous. This design choice deserves some comments. The main purpose of this paper is the study of an adaptation mechanism which turns out to be *safe*, as well as flexible and minimalist, in order to guarantee that communications will continue to evolve in a correct way in case of critical changes in the dynamic environment. The use of asynchronous communications, as in [4], would only add technical complications to deal with reconfiguration of channel queues during adaptation, without providing any significant insight to our issue.

This paper is an improved and extended version of [5]; namely, the formal part is completely revised and simplified, more examples are presented and proofs of properties are added. Section 6 is completely new, but for the statements of the last two theorems.

Outline. The paper has a standard structure. After the running example (Section 2), we present syntax (Section 3), types (Section 4) and semantics (Section 5) of the calculus. Properties are proved in Section 6. In Section 7 we draw some conclusions and discuss related works.

2. Running Example

Let us now enrich the example of the Introduction by assuming that factories interact with the general manager (supported by the technical service) about production policies and suppliers interact with factories and stores by exchanging marketing data. Moreover, the stores can communicate with each other for requiring some products. The session initially consists of:

- a general manager (GM) and a technical service (TS);
- an Italian (IF) and an American factory (AF);
- an Italian (IS) and an American supplier (AS);
- three stores, located in Rome (Ro), New York (NY) and Chicago (Ch).

The global type prescribing the communications is $\mathsf{G} = \mathsf{G}_1 \mid \mathsf{G}_2 \mid \mathsf{G}_3 \mid \mathsf{G}_4 \mid \mathsf{G}_5 \mid \mathsf{G}_6 \mid \mathsf{G}_7$, where $\mathsf{G}_1, \mathsf{G}_2$ are as in the Introduction and:

$$\mathsf{G}_3 = \begin{cases} \mu \mathbf{t}.\mathsf{GM} \to \mathsf{IF} : \mathit{GIF}(\text{ProductionLines}).\mathsf{GM} \to \mathsf{AF} : \mathit{GAF}(\text{ProductionLines}). \\ \mathsf{IF} \to \mathsf{GM} : \mathit{IFG}(\text{ProgressReport}).\mathsf{AF} \to \mathsf{GM} : \mathit{AFG}(\text{ProgressReport}).\mathbf{t} \end{cases}$$

$$\mathsf{G}_4 = \mu \mathbf{t}.\mathsf{AS} \to \mathsf{AF} : \mathit{SF}(\text{Item,Amount}).\mathsf{AF} \to \mathsf{AS} : \mathit{FS}(\text{DeliveryDate}).\mathbf{t}$$

$$\mathsf{G}_5 = \mu \mathbf{t}.\mathsf{Ch} \to \mathsf{AS} : \mathit{CS}(\text{Item,Amount}).\mathsf{AS} \to \mathsf{Ch} : \mathit{SC}(\text{DeliveryDate}).\mathbf{t}$$

$$\mathsf{G}_6 = \begin{cases} \mu \mathbf{t}.\mathsf{NY} \to \mathsf{AS} : \mathit{NS}(\text{Item,Amount}). \\ \quad \mathsf{AS} \to \mathsf{NY} : \{\mathit{YES}(\text{DeliveryDate}).\mathsf{NY} \to \mathsf{Ch} : \mathit{NCY}(\text{NoItem}).\mathbf{t}, \\ \qquad\qquad \mathit{NO}(\text{NoItem}) : \mathsf{NY} \to \mathsf{Ch} : \mathit{NCN}(\text{Item,Amount}). \\ \qquad\qquad\qquad\qquad \mathsf{Ch} \to \mathsf{NY} : \mathit{CN}(\text{DeliveryDate}).\mathbf{t}\} \end{cases}$$

$$\mathsf{G}_7 = \mu \mathbf{t}.\mathsf{GM} \to \mathsf{TS} : GT(\text{PlantCheck}).\mathsf{TS} \to \mathsf{GM} : TG(\text{PlantReport}).\mathbf{t}$$

Note that according to G_6 the New York store can ask for items both to the America supplier and, if he cannot provide the requested items, to the Chicago store. Moreover, G_1 and G_4 differ for the participants, but not for the labels, and this allows the process

$$\mu X.?SF(item, amount).!FS(\mathsf{deliveryDate}).X$$

to incarnate both IF for G_1 and AF for G_4. This process in parallel with

$$\mu X.?GIF(productionLines).!IFG(\mathsf{progressReport}).X,$$

is the code of IF for the whole $\mathbb{G}$.

Assume now that the catastrophic event of a fire, incapacitating the American factory, requires the session to update itself. The American factory modifies the control data by putting the information about its unavailability. The adaptation function F, applied to the modified control data, kills AF and produces the global type

$$\mathbb{G}' = \begin{cases} \mathsf{FF} \to \mathsf{GM} : FG(\text{DamagesReport}). \\ \mu \mathbf{t}.\mathsf{GM} \to \mathsf{TS} : GR(\text{RecPlan}).\mathsf{TS} \to \mathsf{GM} : RG(\text{TechRevisions}).\mathbf{t} \mid \\ \mu \mathbf{t}.\mathsf{AS} \to \mathsf{IS} : ASI(\text{Item,Amount}).\mathsf{IS} \to \mathsf{AS} : ISA(\text{DeliveryDate}).\mathbf{t} \end{cases}$$

which adds the firefighter FF and prescribes how the participants GM, TS, IS, AS must be reconfigured. According to $\mathbb{G}'$, the FF sends to the general manager a report on the damages and then the general manager opens with the technical service a conversation about the reconstruction plan. The American seller now asks for products to the Italian seller. Notice that the labels of the communications between them must be different from those in G_1 and in G_2 which are not modified, since the Italian seller continues to ask products to the Italian factory and to serve the store in Rome as before. Then the new global type for the reconfigured session is the parallel of $\mathbb{G}'$ with the global type that is obtained from $\mathbb{G}$ by pruning all communications involving AF from G_3, by erasing G_4, since all communications involve AF, and by deleting G_7, since all communications are between GM and TS. The rational is to erase the communications in which

- either at least one of the two participants is killed,
- or both participants are reconfigured.

The new process incarnating IS is obtained by putting

$$\mu X.?ASI(item, amount).!ISA(\mathsf{deliveryDate}).X$$

in parallel with the previous process incarnating IS. Then the code of the IS participant becomes:

$$\mu X.?ASI(item, amount).!ISA(\mathsf{deliveryDate}).X \mid \\ \mu X.!SF(\mathsf{item}, \mathsf{amount}).?FS(date).X \mid \mu X.?RS(item, amount).!SR(\mathsf{date}).X$$

For readability in this example (and in that of the Introduction) we described adaptation as modification of global types. In the semantics (Section 5) global types are left implicit and the adaptation rule modifies monitors and processes.

The above example will be used as the running example in the rest of the paper. The following examples will refer to global types, monitors, processes etc. here defined, by developing further details in order to illustrate features and technical definitions of our calculus.

3. Syntax

We use the following base sets: *values*, ranged over by $v, v', \ldots$; *expressions*, ranged over by $\mathsf{e}, \mathsf{e}', \ldots$; *expression variables*, ranged over by $x, y, z \ldots$; *labels*, ranged over by $\ell, \ell', \ldots$; *session participants*, ranged over by $\mathsf{p}, \mathsf{q}, \ldots$; *process variables*, ranged over by $X, Y, \ldots$.

3.1. Global Types

Following a widely common approach [2], the set-up of protocols starts from global types. Global types establish overall communication schemes. In our setting they also control the reconfiguration phase, in which a session adapts itself to new environmental conditions. Exchanged values are marked by labels, as in [6].

We assume some basic *sorts*, ranged over by S, i.e. $S ::= \mathsf{Bool} \mid \mathsf{Int} \mid \ldots$.

Single threaded global types give sequential communications with alternatives and a default choice. In $\mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i \in I}$ participant p sends to participant q a label l_i together with a value of sort S_i for some $i \in I$. The set I contains a distinguished element $\mathsf{d}(I)$ used as default when the communication between p and q is erased, see Definition 5.2. We implicitly assume that $\ell_i \neq \ell_j$ for all $i \neq j$.

Global types are parallel compositions of single threaded global types with distinct labels for common participants.

Definition 3.1. *1.* Single threaded global types *are defined by:*

$$\mathsf{G} \;::=\; \mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i \in I} \;\mid\; \mu \mathbf{t}.\mathsf{G} \;\mid\; \mathbf{t} \;\mid\; \mathsf{end}$$

2. Global types *are defined by:*

$$\mathbb{G} \;::=\; \mathsf{G} \mid \mathbb{G} \mid \mathbb{G}$$

where global types have distinct labels for common participants.

In writing examples we omit brackets when there is only one branch.

Example 3.2. $\mathsf{FF} \to \mathsf{GM} : \mu \mathbf{t}.FG(\mathrm{DamagesReport}).\mathbf{t}$ *and* $\mu \mathbf{t}.\mathsf{GM} \to \mathsf{TS} : FG(\mathrm{RecPlan}).\mathbf{t}$ *are single threaded global types, but their parallel composition*

$$\mu \mathbf{t}.\mathsf{FF} \to \mathsf{GM} : FG(\mathrm{DamagesReport}).\mathbf{t} \mid \mu \mathbf{t}.\mathsf{GM} \to \mathsf{TS} : FG(\mathrm{RecPlan}).\mathbf{t}$$

is not a global type, since participant GM *has parallel communications with the same label FG.*

We allow parallels of global types with shared participants, a possibility usually forbidden [1, 2, 7].

$$(\mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i\in I}) \upharpoonright \mathsf{r} =$$

$$\begin{cases} \mathsf{p}?\{\ell_i(S_i).\mathsf{G}_i \upharpoonright \mathsf{q}\}_{i\in I} & \text{if } \mathsf{r} = \mathsf{q} \\ \mathsf{q}!\{\ell_i(S_i).\mathsf{G}_i \upharpoonright \mathsf{p}\}_{i\in I} & \text{if } \mathsf{r} = \mathsf{p} \\ \mathsf{G}_{i_0} \upharpoonright \mathsf{r} & \text{if } \mathsf{r} \neq \mathsf{p} \text{ and } \mathsf{r} \neq \mathsf{q} \text{ where } i_0 \in I \\ & \text{and } \mathsf{G}_i \upharpoonright \mathsf{r} = \mathsf{G}_j \upharpoonright \mathsf{r} \text{ for all } i, j \in I \\ \mathsf{u}?\bigcup_{k\in K}\{\ell_k(S_k).\mathsf{G}_k \upharpoonright \mathsf{r}\} & \text{if } \mathsf{r} \neq \mathsf{p}, \mathsf{r} \neq \mathsf{q} \text{ where } \mathsf{G}_i \upharpoonright \mathsf{r} = \mathsf{u}?\{\ell_j(S_j).\mathsf{G}_j\}_{j\in J_i} \text{ for all } i \in I \text{ and} \\ & K = \bigcup_{i\in I} J_i \text{ and } J_i \cap J_h = \emptyset \text{ for all } i, h \in I \text{ such that } i \neq h \\ & \text{and } \ell_m \neq \ell_n \text{for all } m, n \in K \text{ such that } m \neq n \end{cases}$$

$$(\mu\mathbf{t}.\mathsf{G}) \upharpoonright \mathsf{p} = \begin{cases} \mu\mathbf{t}.\mathsf{G} \upharpoonright \mathsf{p} & \text{if } \mathsf{p} \in \mathsf{G}, \\ \mathsf{end} & \text{otherwise.} \end{cases} \qquad \mathbf{t} \upharpoonright \mathsf{p} = \mathbf{t} \qquad \mathsf{end} \upharpoonright \mathsf{p} = \mathsf{end}$$

$$(\mathbb{G} \mid \mathbb{G}') \upharpoonright \mathsf{p} = (\mathbb{G} \upharpoonright \mathsf{p}) \mid (\mathbb{G}' \upharpoonright \mathsf{p})$$

Table 1: Projection of a global type onto a participant.

3.2. Monitors

Single threaded monitors can be viewed as projections of single threaded global types onto individual participants, as in the standard approach of [1] and [8]. In our calculus, however, monitors are more than types, as in [4]: they have an active role in session dynamics, since they guide communications and adaptations. As expected, monitors are parallel compositions of single threaded monitors with distinct labels.

Definition 3.3. *1.* Single threaded monitors *are defined by:*

$$\mathsf{M} \;::=\; \mathsf{p}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \;\mid\; \mathsf{q}!\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \;\mid\; \mu\mathbf{t}.\mathsf{M} \;\mid\; \mathbf{t} \;\mid\; \mathsf{end}$$

2. Monitors *are defined by:*

$$\mathbb{M} \;::=\; \mathsf{M} \;\mid\; \mathbb{M} \mid \mathbb{M}$$

where monitors in parallel have distinct labels.

An input monitor $\mathsf{p}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I}$ is able to drive a process that can receive, for each $i \in I$, a value of sort S_i, labeled by ℓ_i, having as continuation a process which is adequate for M_i. This corresponds to an external choice. Dually an output monitor $\mathsf{q}!\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I}$ is able to drive a process which can send (by an internal choice) a value of sort S_i, labeled by ℓ_i, and then continues as prescribed by M_i for each $i \in I$. As for global types, the set I contains a distinguished element $\mathsf{d}(I)$ used as default.

We point out that a main concern of our approach is that adaptation points must be unpredictable in the overall choreography. Thus monitors and global types only prescribe communication patterns to participants, being totally unaware of the operations

that update data. These operations are distinctive characteristics of the implementing processes.

In writing global types and monitors, we assume that the parallel composition is associative and commutative and has end as the neutral element.

The projection of global types onto participants is given in Table 1. A projection onto a participant r not involved, as sender or receiver, in a choice is defined if either the projection onto r of all continuations are the same (condition $\mathsf{G}_i \upharpoonright \mathsf{q} = \mathsf{G}_j \upharpoonright \mathsf{q}$ for all $i, j \in I$) or all these projections can be merged in an input monitor (last case), as in [6].

Example 3.4. *The projection of the global type* G_6 *onto participant* Ch *is*

$$\mu \mathbf{t}.\mathsf{NY}?\{\mathit{NCY}(\text{NoItem}).\mathbf{t}, \mathit{NCN}(\text{Item,Amount}).\mathsf{NY}!\mathit{CN}(\text{DeliveryDate}).\mathbf{t}\}$$

A global type $\mathbb{G}$ is *well formed* if its projections are defined for all participants. We always consider well-formed global types.

3.3. Processes

Processes represent code that is associated to monitors in order to implement participants. As in [4] and differently from standard session calculi (see [9] and the references there), processes do not specify the participants involved in sending and receiving actions. The associated monitors determine senders and receivers. Processes do not have explicit channels, since we consider only one session.

Definition 3.5. *1.* Single threaded processes *are defined by:*

$$\begin{array}{rcl} P & ::= & ?\ell(x).P \mid !\ell(\mathsf{e}).P \mid P+P \mid \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P\ \mathsf{else}\ P \mid \\ & & \mu X.P \mid X \mid \mathsf{op}.P \mid \mathbf{0} \end{array}$$

2. Processes *are defined by:*

$$\mathbb{P} \quad ::= \quad P \quad \| \, \mathbb{P} \mid \mathbb{P}.$$

The syntax of processes is rather standard, in particular the operator $+$ represents external choice. The op operator represents an action on control data, for instance a "read" or "write" operation. We leave unspecified the kind of actions, since we are only interested in the dynamic changes of this data, which determine the self-reconfiguration of the whole session.

The parallel composition and the external choice are associative and commutative. Moreover, $\mathbf{0}$ is the neutral element of the parallel composition.

Example 3.6. *The control data of the session also contain information about the status of the factories. Indeed, processes implementing factories have in parallel a conditional process*

$$\mu X.\mathsf{if}\ \mathsf{check}\ \mathsf{then}\ \mathsf{SET\_ALARM}\ \mathsf{else}\ X$$

where check *is a predicate whose value depends on the local data of the factories (omitted in our model) and the operation* SET_ALARM *updates control data to signal that a factory is dramatically damaged. This process does not perform any communication action, it only has the effect of modifying data. Then it is transparent for the global types.*

3.4. Networks

A process is always controlled by a monitor, which ensures that all performed actions fit the protocol prescribed by the global type. Each monitor controls a single process. We write $\mathbb{M}[\mathbb{P}]$ to represent a process $\mathbb{P}$ controlled by a monitor $\mathbb{M}$, dubbed *monitored process*. Participants correspond to pairs of participant identifiers and monitored processes, denoted by $\mathsf{p} \ltimes \mathbb{M}[\mathbb{P}]$. A well-typed monitored process cannot communicate with itself, since rule [MP] in Table 6 forbids $\mathsf{p} \ltimes \mathbb{M}[\mathbb{P}]$ if p occurs in $\mathbb{M}$.

The parallel composition (denoted by $\parallel$) of participant identifiers with monitored processes forms a network.

Definition 3.7. Networks *are defined by:*

$$N \ ::= \mathsf{p} \ltimes \mathbb{M}[\mathbb{P}] \ \mid \ N \parallel N$$

The parallel composition $\parallel$ is associative and commutative and it has $\mathsf{p} \ltimes \mathsf{end}[\mathbf{0}]$, for any p, as neutral elements.

3.5. Sessions

A session is represented as the composition (via "$\between$") of a network, a set of control data σ, a *collection of processes* $\mathscr{P}$ and an *adaptation function* F. The collection of processes is used to find processes adequate to monitors, see Definition 4.3. Thus processes provide implementation codes, that are related to the specific application context. The adaptation function says how to run adaptations according to critical data variations.

Definition 3.8. Sessions *are defined by:*

$$\mathscr{S} \ ::= \ N \between \sigma \between \mathscr{P} \between F$$

Example 3.9. *In our example, the collection of processes must contain*

- *processes implementing the participants* FF, TS, AS *and* IS *which are adequate for the monitors obtained by projecting* $\mathbb{G}'$ *and*
- *a process implementing the participant* GM *which is adequate for the parallel composition of the monitor obtained by projecting* $\mathbb{G}'$ *with the monitor obtained by projecting* G_3 *after erasing the communications with* AF *(dubbed* G'_3 *in Example 6.15).*

The adaptation function, applied to control data where the status of AF *is* ALARM*, returns the global type* $\mathbb{G}'$*, puts the status of* AF *to* KO *and prescribes to remove the participant* AF*.*

Notice that AF *is killed and* IF*,* Ro*,* NY*,* Ch *are not changed by the adaptation. For this reason processes implementing these participants are not needed in the collection of processes.*

4. Typing Processes

Process types describe process communication behaviours [10]. They have prefixes corresponding to sending and receiving of labels and values.

Definition 4.1. *1.* Single threaded types *are inductively defined by:*

$$\mathsf{T} ::= \bigwedge_{i\in I} ?\ell_i(S_i).\mathsf{T}_i \mid \bigvee_{i\in I} !\ell_i(S_i).\mathsf{T}_i \mid \mu \mathbf{t}.\mathsf{T} \mid \mathbf{t} \mid \mathsf{end}$$

where all labels in intersections and unions are distinct.

2. Types *are inductively defined by:*

$$\mathbb{T} ::= \mathsf{T} \mid \mathbb{T} \mid \mathbb{T}$$

where types in parallel have distinct labels.

Types are built from input and output prefixes ($?l(S)$ and $!l(S)$) by means of constructs for intersection types (used to type external choices) and union types (used to type conditionals).

We assume that intersection, union and parallel composition are associative and commutative. Moreover, end is the neutral element of parallel composition. In writing examples we omit intersections/unions when there is only one branch.

Recursive global types, monitors, processes and process types with the same regular trees are considered equal [11, 20.2]. We assume that all recursions are guarded.

An *environment* Γ is a finite mapping from expression variables to sorts and from process variables to single threaded types:

$$\Gamma ::= \emptyset \mid \Gamma, x : S \mid \Gamma, X : \mathsf{T}$$

where the notation $\Gamma, x : S$ ($\Gamma, X : \mathsf{T}$) means that x (X) does not occur in Γ.

We assume that expressions are typed by sorts, as usual. The typing judgments for expressions are of the shape

$$\Gamma \vdash \mathsf{e} : S$$

and the typing rules for expressions are standard.

The typing judgments for processes have the form

$$\Gamma \vdash \mathbb{P} : \mathbb{T}$$

Typing rules for processes are given in Table 2. The external choice is typed by an intersection type, since an external choice offers both behaviours of the composing processes. Dually, a conditional is an internal choice and so it is typed by a union type. As in [4], a single threaded process is equipped with a single threaded type, ranged over by T, describing its communication actions. Parallel processes are typed by parallel types.

A process which does not perform communications, like the one defined in Example 3.6, has type end.

$$\frac{\Gamma, x : S \vdash P : \mathsf{T}}{\Gamma \vdash ?\ell(x).P : ?\ell(S).\mathsf{T}}\ \textsc{Rcv} \qquad \frac{\Gamma \vdash \mathsf{e} : S \qquad \Gamma \vdash P : \mathsf{T}}{\Gamma \vdash !\ell(\mathsf{e}).P : !\ell(S).\mathsf{T}}\ \textsc{Send}$$

$$\frac{\Gamma \vdash P_1 : \mathsf{T}_1 \quad \Gamma \vdash P_2 : \mathsf{T}_2}{\Gamma \vdash P_1 + P_2 : \mathsf{T}_1 \wedge \mathsf{T}_2}\ \textsc{Choice}$$

$$\frac{\Gamma \vdash \mathsf{e} : \mathrm{Bool} \quad \Gamma \vdash P_1 : \mathsf{T}_1 \quad \Gamma \vdash P_2 : \mathsf{T}_2}{\Gamma \vdash \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 : \mathsf{T}_1 \vee \mathsf{T}_2}\ \textsc{If}$$

$$\frac{\Gamma, X : \mathsf{T} \vdash P : \mathsf{T}}{\Gamma \vdash \mu X.P : \mathsf{T}}\ \textsc{rec} \qquad \Gamma, X : \mathsf{T} \vdash X : \mathsf{T}\ \ \textsc{Ax}$$

$$\frac{\Gamma \vdash P : \mathsf{T}}{\Gamma \vdash \mathsf{op}.P : \mathsf{T}}\ \textsc{OpT} \qquad \Gamma \vdash \mathbf{0} : \mathsf{end}\ \ \textsc{End}$$

$$\frac{\Gamma \vdash \mathbb{P}_1 : \mathbb{T}_1 \quad \Gamma \vdash \mathbb{P}_2 : \mathbb{T}_2}{\Gamma \vdash \mathbb{P}_1 \mid \mathbb{P}_2 : \mathbb{T}_1 \mid \mathbb{T}_2}\ \textsc{Par}$$

Table 2: Typing rules for processes.

[SUB-END]
$\mathsf{end} \leq \mathsf{end}$

[SUB-IN]
$$\frac{\forall i \in I : \quad \mathsf{T}_i \leq \mathsf{T}'_i}{\bigwedge_{i \in I \cup J} ?\ell_i(S_i).\mathsf{T}_i \leq \bigwedge_{i \in I} ?\ell_i(S_i).\mathsf{T}'_i}$$

[SUB-OUT]
$$\frac{\forall i \in I : \quad \mathsf{T}_i \leq \mathsf{T}'_i}{\bigvee_{i \in I} !\ell_i(S_i).\mathsf{T}_i \leq \bigvee_{i \in I \cup J} !\ell_i(S_i).\mathsf{T}'_i}$$

Table 3: Subtyping.

Example 4.2. *For the process incarnating* IF *we can derive the type:*

$$\mu\mathbf{t}.?SF(\text{Item,Amount}).!FS(\text{DeliverDate}).\mathbf{t} \mid \mu\mathbf{t}.?GIF(\text{ProductionLines}).!IFG(\text{ProgressReport}).\mathbf{t}$$

We end this section by defining the notion of *adequacy* between a process $\mathbb{P}$ and a monitor $\mathbb{M}$, denoted $\mathbb{P} \propto \mathbb{M}$, which is based on the matching between types and monitors. This matching is made rather flexible by using the *subtype* relation on types defined in Table 3. The double line in rules indicates that the rules are interpreted *coinductively* [11, 21.1]. Subtyping is monotone, for input/output prefixes, with respect to continuations and it follows the usual set theoretic inclusion of intersection and union.

Definition 4.3. *1. A single threaded process P is* adequate *to a single threaded monitor* M *(notation $P \propto \mathsf{M}$) if $\vdash P : \mathsf{T}$ and $\mathsf{T} \leq |\mathsf{M}|$, where the mapping $|\ |$ is defined by:*

$$|\mathsf{p}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I}| = \bigwedge_{i\in I}?\ell_i(S_i).|\mathsf{M}_i| \qquad |\mathsf{q}!\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I}| = \bigvee_{i\in I}!\ell_i(S_i).|\mathsf{M}_i|$$
$$|\mu\mathbf{t}.\mathsf{M}| = \mu\mathbf{t}.|\mathsf{M}| \qquad |\mathbf{t}| = \mathbf{t} \qquad |\mathsf{end}| = \mathsf{end}$$

2. *The* adequacy *of a process* $\mathbb{P}$ *to a monitor* $\mathbb{M}$ *(notation* $\mathbb{P} \propto \mathbb{M}$*) is the smallest relation such that:*
   - $P \propto \mathsf{M}$ *since they satisfy the condition of (1);*
   - $\mathbb{P}_1 \propto \mathbb{M}_1$ *and* $\mathbb{P}_2 \propto \mathbb{M}_2$ *imply* $\mathbb{P}_1 \mid \mathbb{P}_2 \propto \mathbb{M}_1 \mid \mathbb{M}_2$.

Example 4.4. *The type of Example 4.2 can be obtained by applying the mapping* $|\ |$ *to the monitor:*

$$\mu\mathbf{t}.\mathsf{IS}?SF(\text{Item,Amount}).\mathsf{IS}!FS(\text{DeliverDate}).\mathbf{t} \mid \mu\mathbf{t}.\mathsf{GM}?GIF(\text{ProductionLines}).\mathsf{GM}!IFG(\text{ProgressReport}).\mathbf{t}$$

Observe that adequacy is preserved by the parallel composition with processes without communications. Formally, if $\mathbb{P} \propto \mathbb{M}$ and $\vdash P : \mathsf{end}$, then $P \mid \mathbb{P} \propto \mathbb{M}$. Rule [MP] in Table 6 assures the adequacy of processes to monitors in well-typed monitored processes.

Adequacy is decidable, since processes have unique types and subtyping is decidable [12].

5. Semantics

Processes can communicate labels and values, or can read/modify the control data trough op operations. The semantics of processes is described via the LTS defined in Table 4, where the treatment of inputs, outputs, conditionals and recursions is standard. By $\mathsf{e}\downarrow v$ we denote that the evaluation of the expression e produces the value v.

$$?\ell(x).P \xrightarrow{?\ell(v)} P\{v/x\} \qquad !\ell(\mathsf{e}).P \xrightarrow{!\ell(v)} P \ \ \mathsf{e}\downarrow v \qquad \mathsf{op}.P \xrightarrow{\mathsf{op}} P$$

$$\mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow{\tau} P \ \ \mathsf{e}\downarrow \mathsf{true} \qquad \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow{\tau} Q \ \ \mathsf{e}\downarrow \mathsf{false}$$

$$\frac{P \xrightarrow{\beta} P'}{P+Q \xrightarrow{\beta} P'} \qquad \frac{P \xrightarrow{\gamma} P'}{P+Q \xrightarrow{\gamma} P'+Q} \qquad \frac{\mathbb{P} \xrightarrow{\delta} \mathbb{P}'}{\mathbb{P} \mid \mathbb{P}'' \xrightarrow{\delta} \mathbb{P}' \mid \mathbb{P}''}$$

Table 4: LTS of processes.

In the rules for external choice, β ranges over $?\ell(v), !\ell(v)$ and γ ranges over τ, op. In the rule for parallel composition δ ranges over β and γ. The external choices are done by the communication actions, while the operations on the global state are transparent. This is needed since the operations on the control data are recorded neither in the process types nor in the monitors. An operation on data in an external choice can be performed also if a branch, different from that containing the operation, is then executed.

The evolution of a session depends on the evolution of its network. Monitors guide the communications of processes by choosing the senders/receivers and by allowing only some actions among those offered by the processes. This is formalised by the following LTS for monitors:

$$\mathsf{p}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \xrightarrow{\mathsf{p}?\ell_j} \mathsf{M}_j \quad j\in I \qquad\qquad \mathsf{q}!\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \xrightarrow{\mathsf{q}!\ell_j} \mathsf{M}_j \quad j\in I$$

$$\frac{\mathsf{M} \xrightarrow{\alpha} \mathsf{M}'}{\mathsf{M} \mid \mathbb{M}'' \xrightarrow{\alpha} \mathsf{M}' \mid \mathbb{M}''}$$

where α ranges over $\mathsf{p}?l$ and $\mathsf{q}!l$.

Rule [Com] allows monitored processes to exchange messages:

$$\frac{\mathbb{M}_1 \xrightarrow{\mathsf{q}?\ell} \mathbb{M}_1' \quad \mathbb{P}_1 \xrightarrow{?\ell(v)} \mathbb{P}_1' \quad \mathbb{M}_2 \xrightarrow{\mathsf{p}!\ell} \mathbb{M}_2' \quad \mathbb{P}_2 \xrightarrow{!\ell(v)} \mathbb{P}_2'}{\mathsf{p}\ltimes\mathbb{M}_1[\mathbb{P}_1] \parallel \mathsf{q}\ltimes\mathbb{M}_2[\mathbb{P}_2] \longrightarrow \mathsf{p}\ltimes\mathbb{M}_1'[\mathbb{P}_1'] \parallel \mathsf{q}\ltimes\mathbb{M}_2'[\mathbb{P}_2']}\ \textsc{Com}$$

Example 5.1. *Let*

$$\begin{array}{lcl} \mathsf{M}_1 & = & \mu\mathbf{t}.\mathsf{IS}?SF(\text{Item,Amount}).\mathsf{IS}!FS(\text{DeliveryDate}).\mathbf{t} \\ \mathsf{M}_2 & = & \mu\mathbf{t}.\mathsf{IF}!SF(\text{Item,Amount}).\mathsf{IF}?FS(\text{DeliveryDate}).\mathbf{t} \\ P_1 & = & \mu X.?SF(item,amount).!FS(\mathsf{deliveryDate}).X \\ P_2 & = & \mu X.!SF(\mathsf{item},\mathsf{amount}).?FS(deliveryDate).X \end{array}$$

Then

$$\mathsf{IF}\ltimes\mathsf{M}_1[P_1] \parallel \mathsf{IS}\ltimes\mathsf{M}_2[P_2] \longrightarrow \mathsf{IF}\ltimes\mathsf{M}_1'[P_1'] \parallel \mathsf{IS}\ltimes\mathsf{M}_2'[P_2']$$

where

$$\begin{array}{lcl} \mathsf{M}_1' & = & \mathsf{IS}!FS(\text{DeliveryDate}).\mathsf{M}_1 \\ \mathsf{M}_2' & = & \mathsf{IF}?FS(\text{DeliveryDate}).\mathsf{M}_2 \\ P_1' & = & !FS(\mathsf{deliveryDate}).P_1 \\ P_2' & = & ?FS(deliveryDate).P_2 \end{array}$$

Monitored processes can modify control data thanks to rule [Op]:

$$\frac{\mathbb{P} \xrightarrow{\mathsf{op}} \mathbb{P}'}{\mathsf{p}\ltimes\mathbb{M}[\mathbb{P}] \parallel N \between \sigma \between \mathscr{P} \between F \longrightarrow \mathsf{p}\ltimes\mathbb{M}[\mathbb{P}'] \parallel N \between \mathsf{op}(\sigma) \between \mathscr{P} \between F}\ \textsc{Op}$$

We define the set $\mathsf{pa}(\mathbb{G})$ of participants of a global type $\mathbb{G}$ as follows:

$$\mathsf{pa}(\mathsf{p}\to\mathsf{q}:\{\ell_i(S_i).\mathsf{G}_i\}_{i\in I}) = \{\mathsf{p},\mathsf{q}\}\cup\mathsf{pa}(\mathsf{G}_i)\ (i\in I)^2$$

$$\mathsf{pa}(\mu\mathbf{t}.\mathsf{G}) = \mathsf{pa}(\mathsf{G}) \qquad \mathsf{pa}(\mathbf{t}) = \mathsf{pa}(\mathsf{end}) = \emptyset \qquad \mathsf{pa}(\mathbb{G}_1 \mid \mathbb{G}_2) = \mathsf{pa}(\mathbb{G}_1)\cup\mathsf{pa}(\mathbb{G}_2)$$

Adaptation is triggered by a function: this function applied to control data σ returns a global type $\mathbb{G}$, which is the staging of new interactions for the adapted session, a set $\mathscr{K}$ of participants that are removed and a new control data σ'. The outcome of the adaptation function determines a reconfiguration of the system as summarised below. Let $\mathscr{A}$ denote the set of the session participants before adaptation. The session running before the adaptation step is modified in the following way:

[2] The projectability of G assures $\{\mathsf{p},\mathsf{q}\}\cup\mathsf{pa}(\mathsf{G}_i) = \{\mathsf{p},\mathsf{q}\}\cup\mathsf{pa}(\mathsf{G}_j)$ for all $i,j\in I$.

1. the participants in $\mathscr{K}$ are removed;
2. all communications between participants in $\mathscr{A} \setminus \mathscr{K}$ and participants in $\mathscr{K}$ are erased from all monitors of participants in $\mathscr{A} \setminus \mathscr{K}$;
3. all communications between two participants in $\mathscr{A} \cap \mathsf{pa}(\mathbb{G})$ are erased from all monitors of participants in $\mathscr{A} \cap \mathsf{pa}(\mathbb{G})$;
4. the monitors resulting as projections of $\mathbb{G}$ are added in parallel with the monitors of participants in $\mathscr{A} \cap \mathsf{pa}(\mathbb{G})$ obtained as described in (3);
5. the processes with adapted monitors are modified in order to be adequate to these new monitors;
6. the monitored processes of new participants in $\mathsf{pa}(\mathbb{G}) \setminus \mathscr{A}$ are added to the network.

In order to write the formal rule [ADAPT], which performs the adaptation step, we need some preliminary definitions and notations. By $\mathbb{M} \setminus\!\setminus \mathscr{A}$ we denote the monitor obtained from $\mathbb{M}$ by erasing all communications with participants belonging to $\mathscr{A}$ and continuing with the monitors having the default indexes (we recall that they are denoted by $\mathsf{d}(I)$, where I is the set of indexes).

Definition 5.2. *We define* $\mathbb{M} \setminus\!\setminus \mathscr{A}$ *by induction on* $\mathbb{M}$.

$$\mathsf{p}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \setminus\!\setminus \mathscr{A} = \begin{cases} \mathsf{p}?\{\ell_i(S_i).\mathsf{M}_i \setminus\!\setminus \mathscr{A}\}_{i\in I} & \text{if } \mathsf{p} \notin \mathscr{A}, \\ \mathsf{M}_{\mathsf{d}(I)} & \text{otherwise.} \end{cases}$$

$$\mathsf{p}!\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \setminus\!\setminus \mathscr{A} = \begin{cases} \mathsf{p}!\{\ell_i(S_i).\mathsf{M}_i \setminus\!\setminus \mathscr{A}\}_{i\in I} & \text{if } \mathsf{p} \notin \mathscr{A}, \\ \mathsf{M}_{\mathsf{d}(I)} & \text{otherwise.} \end{cases}$$

$$(\mu \mathbf{t}.\mathsf{M}) \setminus\!\setminus \mathscr{A} = \mu \mathbf{t}.\mathsf{M} \setminus\!\setminus \mathscr{A} \qquad \mathbf{t} \setminus\!\setminus \mathscr{A} = \mathbf{t}$$

$$\mathsf{end} \setminus\!\setminus \mathscr{A} = \mathsf{end} \qquad (\mathbb{M} \mid \mathbb{M}') \setminus\!\setminus \mathscr{A} = \mathbb{M} \setminus\!\setminus \mathscr{A} \mid \mathbb{M}' \setminus\!\setminus \mathscr{A}$$

We discuss now the mappings mon and proc.

$$\mathsf{mon}(\mathsf{p}, \mathbb{M}, \mathbb{G}, \mathscr{K}) = \begin{cases} \mathbb{M} \setminus\!\setminus \mathscr{K} & \text{if } \mathsf{p} \notin \mathsf{pa}(\mathbb{G}), \\ (\mathbb{G} \upharpoonright \mathsf{p}) \mid (\mathbb{M} \setminus\!\setminus (\mathsf{pa}(\mathbb{G}) \cup \mathscr{K})) & \text{otherwise.} \end{cases}$$

The mapping mon applied to a participant p, his current monitor $\mathbb{M}$, a global type $\mathbb{G}$ and a set of killed participants $\mathscr{K}$ gives the new monitor for p. If p is not a participant of $\mathbb{G}$, then the new monitor is simply $\mathbb{M}$ where all communications with killed participants are erased. Note that adaptation is transparent to p whenever p does not communicate with killed participants. Otherwise, the new monitor is the parallel composition of the projection of $\mathbb{G}$ onto p with the monitor $\mathbb{M}$, where all communications with participants of $\mathbb{G}$ and killed participants are erased. Clearly mon is a partial mapping, since it is

undefined when some labels occur both in $\mathbb{G}\upharpoonright \mathsf{p}$ and in $\mathbb{M} \setminus\!\setminus (\mathsf{pa}(\mathbb{G}) \cup \mathscr{K})$. We could make mon total simply by always performing a fresh renaming of the labels in $\mathbb{G}$. This dramatic solution however would never allow us to use the current process for a participant in $\mathsf{pa}(\mathbb{G})$, even if the process is adequate to the new monitor. For this reason, in rule [ADAPT] (see page 16) by $\hat{\mathbb{G}}$ we denote $\mathbb{G}$ if $\mathsf{mon}(\mathsf{p}, \mathbb{M}, \mathbb{G}, \mathscr{K})$ is defined for all $\mathsf{pa}(\mathbb{G})$ and a fresh relabelling of $\mathbb{G}$ otherwise.

Example 5.3. *The monitor obtained by projecting the global type* G_3 *onto* GM *is:*

$$\mathsf{M}_3 = \begin{cases} \mu \mathbf{t}.\mathsf{IF}!\mathit{GIF}(\text{ProductionLines}).\mathsf{AF}!\mathit{GAF}(\text{ProductionLines}). \\ \mathsf{IF}?\mathit{IFG}(\text{ProgressReport}).\mathsf{AF}?\mathit{AFG}(\text{ProgressReport}).\mathbf{t} \end{cases}$$

We get

$$\mathsf{mon}(\mathsf{GM}, \mathsf{M}_3, \mathbb{G}', \{\mathsf{AF}\}) = \begin{cases} \mathsf{FF}?\mathit{FG}(\text{DamagesReport}). \\ \mu \mathbf{t}.\mathsf{TS}!\mathit{GR}(\text{RecPlan}).\mathsf{TS}?\mathit{RG}(\text{TechRevisions}).\mathbf{t} \mid \\ \mu \mathbf{t}.\mathsf{IF}!\mathit{GIF}(\text{ProductionLines}).\mathsf{IF}?\mathit{IFG}(\text{ProgressReport}).\mathbf{t} \end{cases}$$

Notice that $\mathsf{mon}(\mathsf{GM}, \mathsf{M}_3, \mathbb{G}', \{\mathsf{AF}\}) = (\mathbb{G}' \upharpoonright \mathsf{GM}) \mid (\mathsf{M}_3 \setminus\!\setminus \{\mathsf{AF}, \mathsf{GM}, \mathsf{TS}, \mathsf{AS}, \mathsf{IS}\})$.

The mapping proc applied to a participant p, his current process $\mathbb{P}$, his current monitor $\mathbb{M}$, a global type $\mathbb{G}$, a set of killed participants $\mathscr{K}$ and a collection of processes $\mathscr{P}$ gives the new process for p. This process must be adequate for $\mathbb{M}' = \mathsf{mon}(\mathsf{p}, \mathbb{M}, \mathbb{G}, \mathscr{K})$ and we want to preserve all the single threaded processes of $\mathbb{P}$ that are not affected by the adaptation. If $\mathbb{P}$ is adequate for $\mathbb{M}'$, then the new process is $\mathbb{P}$. Otherwise we consider $\mathbb{P}$ as the parallel of $\mathbb{P}_1$ and $\mathbb{P}_2$ and $\mathbb{M}'$ as the parallel of $\mathbb{M}_1$ and $\mathbb{M}_2$, where $\mathbb{P}_1$ and $\mathbb{M}_1$ are the parallels of *all* the processes in $\mathbb{P}$ and *all* the monitors in $\mathbb{M}'$ such that $\mathbb{P}_1 \propto \mathbb{M}_1$. The resulting process is then the parallel of $\mathbb{P}_1$ with a process $\mathbb{P}' \in \mathscr{P}$ such that $\mathbb{P}' \propto \mathbb{M}_2$. We can define:

$$\mathsf{proc}(\mathsf{p}, \mathbb{P}, \mathbb{M}, \mathbb{G}, \mathscr{K}, \mathscr{P}) = \begin{cases} \mathbb{P} & \text{if } \mathbb{P} \propto \mathbb{M}' \\ \mathsf{gap}(\mathbb{P}, \mathbb{M}') \mid \mathbb{P}' & \text{if } \mathbb{P}' \in \mathscr{P} \text{ and } \mathbb{P}' \propto \mathsf{sam}(\mathbb{P}, \mathbb{M}') \end{cases}$$

where $\mathbb{M}' = \mathsf{mon}(\mathsf{p}, \mathbb{M}, \mathbb{G}, \mathscr{K})$ and

$$\mathsf{gap}(\Pi_{i \in I} P_i, \Pi_{j \in J} \mathsf{M}_j) = \Pi_{i \in I'} P_i$$
$$\mathsf{sam}(\Pi_{i \in I} P_i, \Pi_{j \in J} \mathsf{M}_j) = \Pi_{j \in J \setminus J'} \mathsf{M}_j$$

taking I' as the maximum subset of I such that there is $J' \subseteq J$ and $\Pi_{i \in I'} P_i \propto \Pi_{j \in J'} \mathsf{M}_j$. If $\mathbb{P}$ has parallel components performing no communication, then these processes are preserved in $\mathsf{gap}(\mathbb{P}, \mathbb{M}')$, by definition of adequacy.

Notice that by construction $\mathsf{proc}(\mathsf{p}, \mathbb{P}, \mathbb{M}, \mathbb{G}, \mathscr{K}, \mathscr{P}) \propto \mathsf{mon}(\mathsf{p}, \mathbb{M}, \mathbb{G}, \mathscr{K})$.

Example 5.4. *1. If* $\mathbb{M} = \mathsf{q}?\ell_1(\text{Int}) \mid \mathsf{r}?\ell_2(\text{Int}) \mid \mathsf{q}'!\ell_3(\text{Bool})$ *and* $\mathbb{G} = \mathsf{p} \to \mathsf{q}' : \ell_4(\text{Int})$ *and* $\mathbb{P} = ?\ell_1(x) \mid ?\ell_2(y) \mid !\ell_3(\mathsf{true})$*, and* $\mathscr{P}$ *contains* $!\ell_4(7)$*, then*

$$\mathsf{proc}(\mathsf{p}, \mathbb{P}, \mathbb{M}, \mathbb{G}, \{\mathsf{r}\}, \mathscr{P}) = ?\ell_1(x) \mid !\ell_4(7)$$

2. *If* M_3 *is as in Example 5.3,*

$$P_3 = \left\{ \begin{array}{l} \mu X.!GIF(\mathsf{productionLines}).!GAF(\mathsf{productionLines}). \\ ?IFG(progressReport).?AFG(progressReport).X \end{array} \right.$$

and $\mathscr{P}$ *contains the process:*

$$\mathbb{P}'_3 = \left\{ \begin{array}{l} ?FG(damagesReport).\mu X.!GR(\mathsf{recPlan}).?RG(techRevisions).X \mid \\ \mu X.!GIF(\mathsf{productionLines}).?IFG(progressReport).X \end{array} \right.$$

then

$$\mathsf{proc}(\mathsf{GM}, P_3, \mathsf{M}_3, \mathbb{G}', \{\mathsf{AF}\}, \mathscr{P}) = \mathbb{P}'_3$$

We can now define the adaptation rule:

$$\frac{\begin{array}{c} F(\sigma) = (\mathbb{G}, \mathscr{K}, \sigma') \quad \mathscr{A}' = \mathscr{A} \setminus \mathscr{K} \quad \mathscr{B} = \mathsf{pa}(\mathbb{G}) \setminus \mathscr{A} \\ \mathbb{M}'_{\mathsf{p}} = \mathsf{mon}(\mathsf{p}, \mathbb{M}_{\mathsf{p}}, \hat{\mathbb{G}}, \mathscr{K}) \quad \mathbb{P}'_{\mathsf{p}} = \mathsf{proc}(\mathsf{p}, \mathbb{P}_{\mathsf{p}}, \mathbb{M}_{\mathsf{p}}, \hat{\mathbb{G}}, \mathscr{K}, \mathscr{P}) \\ \forall \mathsf{p} \in \mathscr{B}. \mathbb{Q}_{\mathsf{p}} \in \mathscr{P} \;\&\; \mathbb{Q}_{\mathsf{p}} \propto \hat{\mathbb{G}} \upharpoonright \mathsf{p} \end{array}}{\begin{array}{c} \prod_{\mathsf{p} \in \mathscr{A}} \mathsf{p} \ltimes \mathbb{M}_{\mathsf{p}}[\mathbb{P}_{\mathsf{p}}] \between \sigma \between \mathscr{P} \between F \longrightarrow \\ \prod_{\mathsf{p} \in \mathscr{A}'} \mathsf{p} \ltimes \mathbb{M}'_{\mathsf{p}}[\mathbb{P}'_{\mathsf{p}}] \parallel \prod_{\mathsf{p} \in \mathscr{B}} \mathsf{p} \ltimes \hat{\mathbb{G}} \upharpoonright \mathsf{p}[\mathbb{Q}_{\mathsf{p}}] \between \sigma' \between \mathscr{P} \between F \end{array}} \textsc{Adapt}$$

Rule [ADAPT] must be used when the adaptation function F applied to the control data σ is defined. In this case F returns a global type $\mathbb{G}$, a set of participants $\mathscr{K}$ which must be killed and a new control data σ'. The global type $\hat{\mathbb{G}}$ is $\mathbb{G}$ with possibly some relabelling, as defined in the explanation of function mon. The set of participants before the adaptation is $\mathscr{A}$. Then, after the adaptation, the new set of participants is $(\mathscr{A} \setminus \mathscr{K}) \cup (\mathsf{pa}(\mathbb{G}) \setminus \mathscr{A})$, i.e. the previous set of participants without the killed ones, plus the participants of $\mathbb{G}$ who do not occur in $\mathscr{A}$. For these new participants the processes are taken from the collection $\mathscr{P}$. Instead, for the participants in $\mathscr{A} \setminus \mathscr{K}$ we compute the new monitors and processes using the mappings mon and proc, respectively.

Table 5 lists the reduction rules of networks and sessions, which are not discussed above. Evaluation contexts are defined by

$$\mathscr{E} ::= [\,] \mid \mathscr{E} \parallel N$$

We observe that no strategy evaluation is formalised in the operational rules. Indeed, the proof of safety, given in next section, does not depend on the order in which rules are applied. However, in a realistic application of our framework, we assume that rule [ADAPT] has the precedence over all other rules, to ensure a prompt adaptation to critical events in the environment. Accordingly, rule [OP], that registers in the control data those critical events, has precedence over the remaining rules.

$$\frac{\mathbb{P} \xrightarrow{\tau} \mathbb{P}'}{\mathsf{p} \ltimes \mathbb{M}[\mathbb{P}] \longrightarrow \mathsf{p} \ltimes \mathbb{M}[\mathbb{P}']}\ \textsc{Tau}$$

$$\frac{N \longrightarrow N'}{\mathscr{E}[N] \between \sigma \between \mathscr{P} \between F \longrightarrow \mathscr{E}[N'] \between \sigma \between \mathscr{P} \between F}\ \textsc{SN}$$

$$\frac{N \between \sigma \between \mathscr{P} \between F \longrightarrow N' \between \sigma' \between \mathscr{P} \between F}{\mathscr{E}[N] \between \sigma \between \mathscr{P} \between F \longrightarrow \mathscr{E}[N'] \between \sigma' \between \mathscr{P} \between F}\ \textsc{Ctx}$$

Table 5: Further network and session reductions.

6. Safety

In this section we show the main properties of our calculus, i.e. adequacy of processes to the enclosing monitors, subject reduction and progress. The proofs rely on the assurance that monitors are projections of a unique global type. The main novelties of our calculus are the parallel monitors and the adaptation rule.

The parallel monitors require some care in the proof that the reduction rule [Com] preserves the types. After rule [ADAPT] some participants communicate in parallel following both the (modified) global type of the session before the application of the rule and the global type resulting from the application of the adaptation function to control data. The resulting overall global type is implicitly represented by its projections.

This section is organised as follows. First, we tell how the LTSs give the shapes of monitors and processes. Second, we present the typing rules for networks and sessions together with an inversion lemma and a lemma on the existence of global types representing communications of typeable networks (Subsection 6.1). The following two subsections deal with properties useful for showing subject reduction when the rules [Com] and [ADAPT] are applied, respectively. Then Subsection 6.4 gives the proof of subject reduction and Subsection 6.5 concludes with key features of the calculus.

6.1. Type System

Monitor LTS transitions reveal the monitor shapes, as detailed in the next lemma, which can be proved by straightforward case analysis.

Lemma 6.1. *1. If* $\mathbb{M} \xrightarrow{\mathsf{p}?\ell} \mathbb{M}'$*, then* $\mathbb{M} = \mathsf{p}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \mid \mathbb{M}''$ *and* $\ell = \ell_j$ *and* $\mathbb{M}' = \mathsf{M}_j \mid \mathbb{M}''$ *for some* $j \in I$*.*

2. If $\mathbb{M} \xrightarrow{\mathsf{q}!\ell} \mathbb{M}'$*, then* $\mathbb{M} = \mathsf{q}!\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \mid \mathbb{M}''$ *and* $\ell = \ell_j$ *and* $\mathbb{M}' = \mathsf{M}_j \mid \mathbb{M}''$ *for some* $j \in I$*.*

Similarly process LTS transitions reveal the process shapes.

Lemma 6.2. *1. If* $\mathbb{P} \xrightarrow{?\ell(v)} \mathbb{P}'$*, then* $\mathbb{P} = P \mid \mathbb{P}''$ *and* $\mathbb{P}' = P' \mid \mathbb{P}''$*, where:*

   (a) *either* $P = ?\ell(x).Q$ *and* $P' = Q\{v/x\}$;

   (b) *or* $P = Q + R$ *and* $Q \xrightarrow{?\ell(v)} Q'$ *and* $P' = Q'$.

2. *If* $\mathbb{P} \xrightarrow{!\ell(v)} \mathbb{P}'$, *then* $\mathbb{P} = P \mid \mathbb{P}''$ *and* $\mathbb{P}' = P' \mid \mathbb{P}''$, *where:*

   (a) *either* $P = !\ell(\mathsf{e}).Q$ *and* $\mathsf{e} \downarrow v$ *and* $P' = Q$;

   (b) *or* $P = Q + R$ *and* $Q \xrightarrow{!\ell(v)} Q'$ *and* $P' = Q'$.

3. *If* $\mathbb{P} \xrightarrow{\tau} \mathbb{P}'$, *then* $\mathbb{P} = P \mid \mathbb{P}''$ *and* $\mathbb{P}' = P' \mid \mathbb{P}''$, *where one of the following conditions holds:*

   (a) $P = \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ Q\ \mathsf{else}\ R$ *and* $\mathsf{e} \downarrow \mathsf{true}$ *and* $P' = Q$;

   (b) $P = \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ Q\ \mathsf{else}\ R$ *and* $\mathsf{e} \downarrow \mathsf{false}$ *and* $P' = R$;

   (c) $P = Q + R$ *and* $Q \xrightarrow{\tau} Q'$ *and* $P' = Q' + R$.

4. *If* $\mathbb{P} \xrightarrow{\mathsf{op}} \mathbb{P}'$, *then* $\mathbb{P} = P \mid \mathbb{P}''$ *and* $\mathbb{P}' = P' \mid \mathbb{P}''$, *where:*

   (a) *either* $P = \mathsf{op}.Q$ *and* $P' = Q$;

   (b) *or* $P = Q + R$ *and* $Q \xrightarrow{\mathsf{op}} Q'$ *and* $P' = Q' + R$.

Notice that processes of the shape (2b) cannot be typed, since an intersection of output types is not a type.

The typing judgements for networks and sessions are of the shape

$$\vdash N \rhd \Delta \qquad\qquad \vdash \mathscr{S} \rhd \Delta$$

where Δ is a *session typing*. *Session typings* associate participants to monitors:

$$\Delta ::= \emptyset \mid \Delta, \mathsf{p} : \mathbb{M}$$

Table 6 gives the typing rules for networks and sessions. We define the set $\mathsf{pa}(\mathbb{M})$ of participants of a monitor $\mathbb{M}$ as follows:

$$\mathsf{pa}(\mathsf{p}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I}) = \mathsf{pa}(\mathsf{p}!\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I}) = \{\mathsf{p}\} \bigcup_{i\in I} \mathsf{pa}(\mathsf{M}_i)$$
$$\mathsf{pa}(\mu \mathbf{t}.\mathsf{M}) = \mathsf{pa}(\mathsf{M}) \qquad \mathsf{pa}(\mathbf{t}) = \mathsf{pa}(\mathsf{end}) = \emptyset \qquad \mathsf{pa}(\mathbb{M}_1 \mid \mathbb{M}_2) = \mathsf{pa}(\mathbb{M}_1) \cup \mathsf{pa}(\mathbb{M}_2)$$

To type a monitored process, rule [MP] requires that the participant p, which is associated to the monitored process $\mathbb{M}[\mathbb{P}]$, does not belong to $\mathsf{pa}(\mathbb{M})$ (condition $\mathsf{p} \notin \mathsf{pa}(\mathbb{M})$) and that $\mathbb{P}$ is adequate to $\mathbb{M}$ (condition $\mathbb{P} \propto \mathbb{M}$). For typing the parallel composition of networks, rule [NPAR] requires the consistency of the session typing in the conclusion (condition $\mathsf{cons}(\Delta_1, \Delta_2)$), where consistency is defined as follows.

Definition 6.3. *A session typing* Δ *is* consistent*, notation* $\mathsf{cons}(\Delta)$*, if there is a global type* $\mathbb{G}$ *such that* $\mathsf{p} : \mathbb{M} \in \Delta$ *implies* $\mathbb{M} = \mathbb{G} \upharpoonright \mathsf{p}$.

$$\frac{}{\vdash \mathsf{p} \ltimes \mathsf{end}[\mathbf{0}] \rhd \emptyset}\ \mathsf{end} \qquad \frac{\mathbb{P} \neq \mathbf{0} \quad \mathbb{P} \propto \mathbb{M} \quad \mathsf{p} \notin \mathsf{pa}(\mathbb{M})}{\vdash \mathsf{p} \ltimes \mathbb{M}[\mathbb{P}] \rhd \{\mathsf{p} : \mathbb{M}\}}\ \mathrm{MP}$$

$$\frac{\vdash N_1 \rhd \Delta_1 \quad \vdash N_2 \rhd \Delta_2 \quad \mathsf{cons}(\Delta_1, \Delta_2)}{\vdash N_1 \parallel N_2 \rhd \Delta_1, \Delta_2}\ \mathrm{NPAR}$$

$$\frac{\vdash N \rhd \Delta \quad \mathsf{comp}(\Delta)}{\vdash N \between \sigma \between \mathscr{P} \between F \rhd \Delta}\ \mathrm{SES}$$

Table 6: Typing rules for networks and sessions.

However, consistency is not enough to get progress, we have to assure that all participants are present. This requirement is guaranteed by the completeness of the session typing, given by Definition 6.4.

Definition 6.4. *A session typing Δ is* complete*, notation* $\mathsf{comp}(\Delta)$*, if there is a global type $\mathbb{G}$ such that* $\mathsf{pa}(\mathbb{G}) = \{\mathsf{p} \mid \mathsf{p} : \mathbb{M} \in \Delta \,\&\, \mathbb{M} \neq \mathsf{end}\}$ *and* $\mathsf{p} : \mathbb{M} \in \Delta$ *implies* $\mathbb{M} = \mathbb{G} \upharpoonright \mathsf{p}$.

Clearly completeness of session typings implies their consistency. Rule [SES] requires the completeness of session typing (condition $\mathsf{comp}(\Delta)$). The control data, the collection of processes and the adaptation function are transparent for the typing.

Example 6.5. *Let* $\mathsf{M}_1, \mathsf{M}_2, P_1, P_2$ *be defined as in Example 5.1. We can derive:*

$$\vdash \mathsf{IF} \ltimes \mathsf{M}_1[P_1] \parallel \mathsf{IS} \ltimes \mathsf{M}_2[P_2] \rhd \{\mathsf{IF} : \mathsf{M}_1, \mathsf{IS} : \mathsf{M}_2\}$$

The inversion lemma takes into account both the typing rules for processes (Table 2) and those for networks and sessions (Table 6).

Lemma 6.6 (Inversion lemma)**.** *1. Let* $\Gamma \vdash P : \mathsf{T}$.

(a) If $P = ?\ell(x).Q$*, then* $\mathsf{T} = ?\ell(S).\mathsf{T}'$ *and* $\Gamma, x : S \vdash Q : \mathsf{T}'$.

(b) If $P = !\ell(\mathsf{e}).Q$*, then* $\mathsf{T} = !\ell(S).\mathsf{T}'$ *and* $\Gamma \vdash \mathsf{e} : S$ *and* $\Gamma \vdash Q : \mathsf{T}'$.

(c) If $P = P_1 + P_2$*, then* $\mathsf{T} = \mathsf{T}_1 \wedge \mathsf{T}_2$ *and* $\Gamma \vdash P_1 : \mathsf{T}_1$ *and* $\Gamma \vdash P_2 : \mathsf{T}_2$.

(d) If $P = \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$*, then* $\mathsf{T} = \mathsf{T}_1 \vee \mathsf{T}_2$ *and* $\Gamma \vdash \mathsf{e} : \mathrm{Bool}$ *and* $\Gamma \vdash P_1 : \mathsf{T}_1$ *and* $\Gamma \vdash P_2 : \mathsf{T}_2$.

(e) If $P = \mathsf{op}.Q$*, then* $\Gamma \vdash Q : \mathsf{T}$.

(f) If $P = \mu X.Q$*, then* $\Gamma, X : \mathsf{T} \vdash Q : \mathsf{T}$.

(g) If $P = X$*, then* $\Gamma = \Gamma', X : \mathsf{T}$.

(h) If $P = \mathbf{0}$*, then* $\mathsf{T} = \mathsf{end}$.

2. If $\vdash \mathbb{P}_1 \mid \mathbb{P}_2 : \mathbb{T}$*, then* $\mathbb{T} = \mathbb{T}_1 \mid \mathbb{T}_2$ *and* $\Gamma \vdash \mathbb{P}_1 : \mathbb{T}_1$ *and* $\Gamma \vdash \mathbb{P}_2 : \mathbb{T}_2$.

3. If $\vdash \mathsf{p} \ltimes \mathbb{M}[\mathbb{P}] \rhd \Delta$*, then:*

$$\mathscr{G}(\mathsf{q}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I},\mathsf{p}) = \mathsf{q}\to\mathsf{p}:\{\ell_i(S_i).\mathscr{G}(\mathsf{M}_i,\mathsf{p})\}_{i\in I}$$
$$\mathscr{G}(\mathsf{q}!\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I},\mathsf{p}) = \mathsf{p}\to\mathsf{q}:\{\ell_i(S_i).\mathscr{G}(\mathsf{M}_i,\mathsf{p})\}_{i\in I}$$
$$\mathscr{G}(\mu\mathbf{t}.\mathsf{M},\mathsf{p}) = \mu\mathbf{t}.\mathscr{G}(\mathsf{M},\mathsf{p}) \qquad \mathscr{G}(\mathbf{t},\mathsf{p}) = \mathbf{t}$$
$$\mathscr{G}(\mathsf{end},\mathsf{p}) = \mathsf{end} \qquad \mathscr{G}(\mathbb{M}\mid\mathbb{M}',\mathsf{p}) = \mathscr{G}(\mathbb{M},\mathsf{p})\mid\mathscr{G}(\mathbb{M}',\mathsf{p})$$

Table 7: The function $\mathscr{G}(\mathbb{M},\mathsf{p})$.

(a) either $\Delta=\emptyset$ *and* $\mathbb{M}=\mathsf{end}$ *and* $\mathbb{P}=\mathbf{0}$*;*

(b) or $\Delta=\{\mathsf{p}:\mathbb{M}\}$ *and* $\mathbb{P}\neq\mathbf{0}$ *and* $\mathbb{P}\propto\mathbb{M}$ *and* $\mathsf{p}\notin\mathsf{pa}(\mathbb{M})$.

4. *If* $\vdash N_1 \parallel N_2 \rhd \Delta$*, then* $\mathsf{cons}(\Delta)$ *and* $\Delta=\Delta_1,\Delta_2$ *and* $\vdash N_1\rhd\Delta_1$ *and* $\vdash N_2\rhd\Delta_2$.

5. *If* $\vdash N \between \sigma \between \mathscr{P} \between F \rhd \Delta$*, then* $\vdash N\rhd\Delta$ *and* $\mathsf{comp}(\Delta)$.

Proof. Easy from the definition of the typing relation. □

It is important to notice that each typeable network/session has a consistent session typing, as shown in Lemma 6.8(2). The crucial case is rule [MP]. For this rule we define the *characteristic global type* $\mathscr{G}(\mathbb{M},\mathsf{p})$ of the monitor $\mathbb{M}$ for a participant $\mathsf{p}\notin\mathsf{pa}(\mathbb{M})$, see Table 7. This global type describes the communications between p and all participants in $\mathsf{pa}(\mathbb{M})$, as prescribed by $\mathbb{M}$. More precisely, $\mathbb{M}$ is the projection onto p of the global type $\mathscr{G}(\mathbb{M},\mathsf{p})$, see (1) of the following lemma.

Example 6.7. *If* G_1 *is defined as in the Introduction and* M_1 *is defined as in Example 5.1, then*

$$\mathscr{G}(\mathsf{M}_1,\mathsf{IF}) = \mathsf{G}_1$$

Lemma 6.8. 1. $\mathbb{M} = \mathscr{G}(\mathbb{M},\mathsf{p})\upharpoonright\mathsf{p}$.

2. *If* $\vdash N\rhd\Delta$*, then* $\mathsf{cons}(\Delta)$.

Proof. (1) By induction on the definition of characteristic global type.
(2) By induction on the derivation of $\vdash N\rhd\Delta$, using (1) for rule [MP]. □

*6.2. Properties for Rule [*COM*]*

A substitution lemma is handy as usual for a communication rule. In our calculus we need also to take into account that the substitution can be in one branch of an external choice. This is done in the following lemma.

Lemma 6.9. 1. *If* $\Gamma\vdash\mathsf{e}:S$ *and* $\mathsf{e}\downarrow v$ *and* $\Gamma,x:S\vdash P:\mathsf{T}$*, then* $\Gamma\vdash P\{v/x\}:\mathsf{T}$.

2. *If* $\Gamma\vdash\mathsf{e}:S$ *and* $\mathsf{e}\downarrow v$ *and* $\Gamma\vdash P:\mathsf{T}$ *with* $\mathsf{T}\leq ?\ell(S).\mathsf{T}'$ *and* $P\xrightarrow{?\ell(v)}P'$*, then* $\Gamma\vdash P':\mathsf{T}'$.

Proof. (1) By induction on the derivation of $\Gamma, x : S \vdash P : \mathsf{T}$.

(2) By induction on the derivation of $P \xrightarrow{?\ell(v)} P'$. In the first step $P = ?\ell(x).Q$ and $P' = Q\{v/x\}$, so we conclude by (1) and Lemma 6.6(1a). In the induction step $P = Q + R$ and $Q \xrightarrow{?\ell(v)} Q'$ and $P' = Q'$, so induction applies. □

Applying rule [Com] both monitors consume one communication, respectively one input and one output. This can be formalised by means of the following notion of reduction between session typings.

Definition 6.10. *The* reduction of session typings *is the minimal reduction relation such that:*

$$\{\mathsf{p} : \mathsf{q}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \mid \mathbb{M}, \mathsf{q} : \mathsf{p}!\{\ell_i(S_i).\mathsf{M}'_i\}_{i\in I} \mid \mathbb{M}'\} \implies \{\mathsf{M}_j \mid \mathbb{M}, \mathsf{M}'_j \mid \mathbb{M}'\} \quad j \in I$$

$$\Delta \implies \Delta' \textit{ implies } \Delta, \Delta'' \implies \Delta', \Delta''$$

The reduction of session typings is required to preserve their consistency, i.e. to map monitors which are projections of a global type into monitors which are projections of a global type too, see Lemma 6.14. To show this property it is useful to consider how a communication modifies a global type, that is the aim of the following definition.

Definition 6.11. *The* consumption *of the communication* $\mathsf{p} \xrightarrow{\ell} \mathsf{q}$ *for the global type* $\mathbb{G}$ *(notation* $\mathbb{G} \setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q}$*) is the global type inductively defined by:*

$$(\mathsf{r} \to \mathsf{s} : \{\ell_i(S_i).\mathsf{G}_i\}_{i\in I}) \setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q} = \begin{cases} \mathsf{G}_j & \textit{if } \mathsf{r} = \mathsf{p}, \\ & \mathsf{s} = \mathsf{q}, \ell_j = \ell \\ \mathsf{r} \to \mathsf{s} : \{\ell_i(S_i).\mathsf{G}_i \setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q}\}_{i\in I} & \textit{if } \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset \end{cases}$$

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

$$(\mathbb{G} \mid \mathbb{G}') \setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q} = \begin{cases} (\mathbb{G} \setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q}) \mid \mathbb{G}' & \textit{if } \mathbb{G} \setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q} \textit{ is defined}, \\ \mathbb{G} \mid (\mathbb{G}' \setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q}) & \textit{otherwise} \end{cases}$$

Example 6.12. *In the running example, we get* $\mathbb{G} \setminus \mathsf{NY} \xrightarrow{NS} \mathsf{AS} = \mathsf{G}_1 \mid \mathsf{G}_2 \mid \mathsf{G}_3 \mid \mathsf{G}_4 \mid \mathsf{G}_5 \mid \mathsf{G}'_6 \mid \mathsf{G}_7$*, where*

$$\mathsf{G}'_6 = \begin{cases} \mathsf{AS} \to \mathsf{NY} : \{YES(\text{DeliveryDate}).\mathsf{NY} \to \mathsf{Ch} : NCY(\text{NoItem}).\mathsf{G}_6, \\ \qquad NO(\text{NoItem}) : \mathsf{NY} \to \mathsf{Ch} : NCN(\text{Item,Amount}). \\ \qquad\qquad \mathsf{Ch} \to \mathsf{NY} : CN(\text{DeliveryDate}).\mathsf{G}_6\} \end{cases}$$

Instead $\mathbb{G} \setminus \mathsf{NY} \xrightarrow{NCY} \mathsf{Ch}$ *is undefined.*

Notice that $\mathbb{G} \setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q}$ is defined only if $\mathbb{G}$ allows as first communication for both p and q a message with label ℓ from p to q. More precisely when $\mathbb{G} \setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q}$ is defined,

then $\mathbb{G}\upharpoonright \mathsf{p} = \mathsf{q}!\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \mid \mathbb{M}$, $\mathbb{G}\upharpoonright \mathsf{q} = \mathsf{p}?\{\ell_i(S_i).\mathsf{M}'_i\}_{i\in I} \mid \mathbb{M}'$ and $\ell = \ell_j$ for some $j \in I$. In this case $(\mathbb{G}\setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q})\upharpoonright \mathsf{p} = \mathsf{M}_j \mid \mathbb{M}$ and $(\mathbb{G}\setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q})\upharpoonright \mathsf{q} = \mathsf{M}'_j \mid \mathbb{M}'$. Moreover $\mathbb{G}\upharpoonright \mathsf{r} = (\mathbb{G}\setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q})\upharpoonright \mathsf{r}$ for $\mathsf{r} \neq \mathsf{p}$, $\mathsf{r} \neq \mathsf{q}$. The vice versa is the content of the following lemma. The proof follows from the definitions of projection and consumption.

Lemma 6.13. *If* $\mathbb{G}\upharpoonright \mathsf{p} = \mathsf{q}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \mid \mathbb{M}$ *and* $\mathbb{G}\upharpoonright \mathsf{q} = \mathsf{p}!\{\ell_i(S_i).\mathsf{M}'_i\}_{i\in I} \mid \mathbb{M}'$ *and* $\ell = \ell_j$ *for some* $j \in I$, *then* $\mathbb{G}\setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q}$ *is defined and* $(\mathbb{G}\setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q})\upharpoonright \mathsf{p} = \mathsf{M}_j \mid \mathbb{M}$ *and* $(\mathbb{G}\setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q})\upharpoonright \mathsf{q} = \mathsf{M}'_j \mid \mathbb{M}'$ *and* $\mathbb{G}\upharpoonright \mathsf{r} = (\mathbb{G}\setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q})\upharpoonright \mathsf{r}$ *for* $\mathsf{r} \neq \mathsf{p}$, $\mathsf{r} \neq \mathsf{q}$.

We can now show that the reduction of session typings preserves consistency.

Lemma 6.14. *If* $\mathsf{cons}(\Delta)$ *and* $\Delta \implies \Delta'$, *then* $\mathsf{cons}(\Delta')$.

Proof. Let $\Delta = \{\mathsf{p}:\mathsf{q}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \mid \mathbb{M}, \mathsf{q}:\mathsf{p}!\{\ell_i(S_i).\mathsf{M}'_i\}_{i\in I} \mid \mathbb{M}'\}$. Then by definition $\Delta \Longrightarrow \{\mathsf{M}_j \mid \mathbb{M}, \mathsf{M}'_j \mid \mathbb{M}'\}$ with $j \in I$. From $\mathsf{cons}(\Delta)$ we get $\mathsf{q}?\{\ell_i(S_i).\mathsf{M}_i\}_{i\in I} \mid \mathbb{M} = \mathbb{G}\upharpoonright \mathsf{p}$ and $\mathsf{p}!\{\ell_i(S_i).\mathsf{M}'_i\}_{i\in I} \mid \mathbb{M}' = \mathbb{G}\upharpoonright \mathsf{q}$ for some $\mathbb{G}$. By Lemma 6.13 $\mathbb{G}\setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q}$ is defined and $(\mathbb{G}\setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q})\upharpoonright \mathsf{p} = \mathsf{M}_j \mid \mathbb{M}$ and $(\mathbb{G}\setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q})\upharpoonright \mathsf{q} = \mathsf{M}'_j \mid \mathbb{M}'$ and $\mathbb{G}\upharpoonright \mathsf{r} = (\mathbb{G}\setminus \mathsf{p} \xrightarrow{\ell} \mathsf{q})\upharpoonright \mathsf{r}$ for $\mathsf{r} \neq \mathsf{p}$, $\mathsf{r} \neq \mathsf{q}$. □

*6.3. Properties for Rule [*ADAPT*]*

The main goal of this subsection is to show a global type such that, under suitable conditions, the monitor obtained as result of the mapping mon is a projection of this global type. This is the content of Lemma 6.16(4). The mapping mon behaves in a different way for participants which do have or not communications in the new global type. For this reason it is handy to define two kinds of erasure of communications from global types, according to their participants. A communication is erased if either one or both the participants belong to a fixed set. More formally:

The *complete erasure* of participants in $\mathscr{A}$ from the global type $\mathbb{G}$ (notation $\mathbb{G}\mathbin{\backslash\!\backslash}\mathscr{A}$) is inductively defined by:

$$\mathsf{p}\to\mathsf{q}:\{\ell_i(S_i).\mathsf{G}_i\}_{i\in I}\mathbin{\backslash\!\backslash}\mathscr{A} = \begin{cases} \mathsf{p}\to\mathsf{q}:\{\ell_i(S_i).\mathsf{G}_i\mathbin{\backslash\!\backslash}\mathscr{A}\}_{i\in I} & \text{if } \mathsf{p}\notin\mathscr{A} \text{ and } \mathsf{q}\notin\mathscr{A},\\ \mathsf{G}_{\mathsf{d}(I)} & \text{otherwise.}\end{cases}$$

$$(\mu\mathbf{t}.\mathsf{G})\mathbin{\backslash\!\backslash}\mathscr{A} = \mu\mathbf{t}.\mathsf{G}\mathbin{\backslash\!\backslash}\mathscr{A} \qquad \mathbf{t}\mathbin{\backslash\!\backslash}\mathscr{A} = \mathbf{t}$$
$$\mathsf{end}\mathbin{\backslash\!\backslash}\mathscr{A} = \mathsf{end} \qquad (\mathbb{G}\mid\mathbb{G}')\mathbin{\backslash\!\backslash}\mathscr{A} = \mathbb{G}\mathbin{\backslash\!\backslash}\mathscr{A}\mid\mathbb{G}'\mathbin{\backslash\!\backslash}\mathscr{A}$$

The *partial erasure* of participants in $\mathscr{A}$ from the global type $\mathbb{G}$ (notation $\mathbb{G}\setminus\mathscr{A}$) is inductively defined by:

$$\mathsf{p}\to\mathsf{q}:\{\ell_i(S_i).\mathsf{G}_i\}_{i\in I}\setminus\mathscr{A} = \begin{cases} \mathsf{p}\to\mathsf{q}:\{\ell_i(S_i).\mathsf{G}_i\setminus\mathscr{A}\}_{i\in I} & \text{if } \mathsf{p}\notin\mathscr{A} \text{ or } \mathsf{q}\notin\mathscr{A},\\ \mathsf{G}_{\mathsf{d}(I)} & \text{otherwise.}\end{cases}$$

$$(\mu\mathbf{t}.\mathsf{G})\setminus\mathscr{A} = \mu\mathbf{t}.\mathsf{G}\setminus\mathscr{A} \qquad \mathbf{t}\setminus\mathscr{A} = \mathbf{t}$$
$$\mathsf{end}\setminus\mathscr{A} = \mathsf{end} \qquad (\mathbb{G}\mid\mathbb{G}')\setminus\mathscr{A} = \mathbb{G}\setminus\mathscr{A}\mid\mathbb{G}'\setminus\mathscr{A}$$

Example 6.15. *In our example we have*

$$\begin{array}{rcl} \mathbb{G} \setminus\!\setminus \{\mathsf{IF},\mathsf{IS}\} & = & \mathsf{G}_3' \mid \mathsf{G}_4 \mid \mathsf{G}_5 \mid \mathsf{G}_6 \mid \mathsf{G}_7 \\ \mathbb{G} \setminus \{\mathsf{IF},\mathsf{IS}\} & = & \mathsf{G}_2 \mid \mathsf{G}_3 \mid \mathsf{G}_4 \mid \mathsf{G}_5 \mid \mathsf{G}_6 \mid \mathsf{G}_7 \end{array}$$

where $\mathsf{G}_3' = \mu \mathbf{t}.\mathsf{GM} \to \mathsf{AF} : \mathit{GAF}(\text{ProductionLines}).\mathsf{AF} \to \mathsf{GM} : \mathit{AFG}(\text{ProgressReport}).\mathbf{t}$.

The main result of this subsection is that, if the monitor $\mathbb{M}$ is the projection of the global type $\mathbb{G}$ onto participant p, then the function mon applied to $\mathsf{p}, \mathbb{M}, \mathbb{G}', \mathscr{K}$ gives a monitor which is the projection of the global type $\mathbb{G}' \mid (\mathbb{G} \setminus \mathsf{pa}(\mathbb{G}')) \setminus\!\setminus \mathscr{K}$ onto p.

Lemma 6.16. *1. If* $\mathsf{p} \notin \mathscr{A}$*, then* $(\mathbb{G} \upharpoonright \mathsf{p}) \setminus\!\setminus \mathscr{A} = (\mathbb{G} \setminus\!\setminus \mathscr{A}) \upharpoonright \mathsf{p}$.

2. If $\mathsf{p} \in \mathscr{A}$*, then* $(\mathbb{G} \upharpoonright \mathsf{p}) \setminus\!\setminus \mathscr{A} = (\mathbb{G} \setminus \mathscr{A}) \upharpoonright \mathsf{p}$.

3. If $\mathsf{p} \notin \mathscr{A}$*, then* $(\mathbb{G} \setminus \mathscr{A}) \upharpoonright \mathsf{p} = \mathbb{G} \upharpoonright \mathsf{p}$.

4. If $\mathbb{M} = \mathbb{G} \upharpoonright \mathsf{p}$*, then* $\mathsf{mon}(\mathsf{p}, \mathbb{M}, \mathbb{G}', \mathscr{K}) = (\mathbb{G}' \mid (\mathbb{G} \setminus \mathsf{pa}(\mathbb{G}')) \setminus\!\setminus \mathscr{K}) \upharpoonright \mathsf{p}$.

Proof. (1), (2) and (3) follow easily by definition.

(4). If $\mathsf{p} \notin \mathsf{pa}(\mathbb{G}')$, then $\mathsf{mon}(\mathsf{p}, \mathbb{M}, \mathbb{G}', \mathscr{K}) = \mathbb{M} \setminus\!\setminus \mathscr{K}$. We get

$$(\mathbb{G}' \mid (\mathbb{G} \setminus \mathsf{pa}(\mathbb{G}')) \setminus\!\setminus \mathscr{K}) \upharpoonright \mathsf{p} = (\mathbb{G} \setminus\!\setminus \mathscr{K}) \upharpoonright \mathsf{p}$$

by (3) and $\mathsf{p} \notin \mathsf{pa}(\mathbb{G}')$. We conclude by (1).

If $\mathsf{p} \in \mathsf{pa}(\mathbb{G}')$ then

$$\begin{array}{rcll} \mathsf{mon}(\mathsf{p}, \mathbb{M}, \mathbb{G}', \mathscr{K}) & = & (\mathbb{G}' \upharpoonright \mathsf{p}) \mid (\mathbb{M} \setminus\!\setminus (\mathsf{pa}(\mathbb{G}') \cup \mathscr{K})) & \\ & = & (\mathbb{G}' \upharpoonright \mathsf{p}) \mid ((\mathbb{M} \setminus\!\setminus (\mathsf{pa}(\mathbb{G}')) \setminus\!\setminus \mathscr{K})) & \\ & = & (\mathbb{G}' \upharpoonright \mathsf{p}) \mid ((\mathbb{G} \setminus \mathsf{pa}(\mathbb{G}')) \upharpoonright \mathsf{p}) \setminus\!\setminus \mathscr{K}) & \text{by (2)} \\ & = & (\mathbb{G}' \upharpoonright \mathsf{p}) \mid ((\mathbb{G} \setminus \mathsf{pa}(\mathbb{G}')) \setminus\!\setminus \mathscr{K}) \upharpoonright \mathsf{p} & \text{by (1)} \\ & = & (\mathbb{G}' \mid (\mathbb{G} \setminus \mathsf{pa}(\mathbb{G}')) \setminus\!\setminus \mathscr{K}) \upharpoonright \mathsf{p} & \end{array}$$

□

Example 6.17. *Lemma 6.16(4) implies that after the adaptation the monitors of the Company are projections of the global type*

$$\mathbb{G}' \mid (\mathbb{G} \setminus \{\mathsf{FF},\mathsf{GM},\mathsf{TS},\mathsf{AS},\mathsf{IS}\}) \setminus\!\setminus \{\mathsf{AF}\}$$

6.4. Subject Reduction

A last lemma is handy to deal with the reduction rules [TAU] and [OP].

Lemma 6.18. *If* $\vdash P : \mathsf{T}$ *and* $P \xrightarrow{\gamma} P'$ *with* $\gamma = \tau$ *or* $\gamma = \mathsf{op}$*, then* $\vdash P : \mathsf{T}'$ *with* $\mathsf{T}' \leq \mathsf{T}$.

Proof. By induction on the derivation of $P \xrightarrow{\gamma} P'$. Lemma 6.2(3) and (4) give the shapes of P and P'.

For the first step with $\gamma = \tau$, let $P = \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ Q\ \mathsf{else}\ R$ and $P' = Q$. By Lemma 6.6(1d) $\mathsf{T} = \mathsf{T}_1 \vee \mathsf{T}_2$ and $\vdash Q : \mathsf{T}_1$. For the first step with $\gamma = \mathsf{op}$, we get $P = \mathsf{op}.Q$ and $P' = Q$. By Lemma 6.6(1e) $\vdash Q : \mathsf{T}$. The proof in the other case with $\gamma = \tau$ is similar.

For the induction step, let $P = Q + R$ and $Q \xrightarrow{\gamma} Q'$ and $P' = Q' + R$. By Lemma 6.6(1c) $\mathsf{T} = \mathsf{T}_1 \wedge \mathsf{T}_2$ and $\vdash Q : \mathsf{T}_1$ and $\vdash R : \mathsf{T}_2$. By induction $\vdash Q' : \mathsf{T}_1'$ with $\mathsf{T}_1' \leq \mathsf{T}_1$. By rule [CHOICE] we derive $\vdash Q' + R : \mathsf{T}_1' \wedge \mathsf{T}_2$.

□

Theorem 6.19. *(Subject Reduction)*

1. *If* $\vdash N \rhd \Delta$ *and* $N \longrightarrow^* N'$, *then* $\vdash N' \rhd \Delta'$ *and* $\Delta \Longrightarrow^* \Delta'$.
2. *If* $\vdash \mathscr{S} \rhd \Delta$ *and* $\mathscr{S} \longrightarrow^* \mathscr{S}'$, *then* $\vdash \mathscr{S}' \rhd \Delta'$ *and* $\Delta \Longrightarrow^* \Delta'$.

Proof. By induction on $\longrightarrow^*$.

(1) Let the last applied rule be

$$\frac{\mathbb{M}_1 \xrightarrow{\mathsf{q}?\ell} \mathbb{M}_1' \quad \mathbb{P}_1 \xrightarrow{?\ell(v)} \mathbb{P}_1' \quad \mathbb{M}_2 \xrightarrow{\mathsf{p}!\ell} \mathbb{M}_2' \quad \mathbb{P}_2 \xrightarrow{!\ell(v)} \mathbb{P}_2'}{\mathsf{p} \ltimes \mathbb{M}_1[\mathbb{P}_1] \parallel \mathsf{q} \ltimes \mathbb{M}_2[\mathbb{P}_2] \longrightarrow \mathsf{p} \ltimes \mathbb{M}_1'[\mathbb{P}_1'] \parallel \mathsf{q} \ltimes \mathbb{M}_2'[\mathbb{P}_2']}\ \textsc{Com}$$

Then $N = \mathsf{p} \ltimes \mathbb{M}_1[\mathbb{P}_1] \parallel \mathsf{q} \ltimes \mathbb{M}_2[\mathbb{P}_2]$ and $N' = \mathsf{p} \ltimes \mathbb{M}_1'[\mathbb{P}_1'] \parallel \mathsf{q} \ltimes \mathbb{M}_2'[\mathbb{P}_2']$. By Lemma 6.6(4) Δ is consistent and $\Delta = \Delta_1, \Delta_2$ and $\vdash \mathsf{p} \ltimes \mathbb{M}_1[\mathbb{P}_1] \rhd \Delta_1$ and $\vdash \mathsf{q} \ltimes \mathbb{M}_2[\mathbb{P}_2] \rhd \Delta_2$. By Lemma 6.6(3b) $\Delta = \{\mathsf{p} : \mathbb{M}_1, \mathsf{q} : \mathbb{M}_2\}$ and

$$\vdash \mathbb{P}_1 : \mathbb{T}_1 \text{ and } \mathbb{P}_1 \propto \mathbb{M}_1 \text{ and } \vdash \mathbb{P}_2 : \mathbb{T}_2 \text{ and } \mathbb{P}_2 \propto \mathbb{M}_2 \tag{1}$$

The consistency of Δ implies $\mathbb{M}_1 = \mathbb{G} \upharpoonright \mathsf{p}$ and $\mathbb{M}_2 = \mathbb{G} \upharpoonright \mathsf{q}$ for some global type $\mathbb{G}$. This together with Lemma 6.1 gives

$$\mathbb{M}_1 = \mathsf{p}?\{\ell_i(S_i).\mathsf{M}_i\}_{i \in I} \mid \mathbb{M}_1'' \text{ and } \mathbb{M}_1' = \mathsf{M}_j' \mid \mathbb{M}_1'' \text{ and} \tag{2}$$

$$\mathbb{M}_2 = \mathsf{q}!\{\ell_i(S_i).\mathsf{M}_i\}_{i \in I} \mid \mathbb{M}_2'' \text{ and } \mathbb{M}_2' = \mathsf{M}_j' \mid \mathbb{M}_2'' \text{ and } \ell = \ell_j \text{ for some } j \in I \tag{3}$$

By Definition 6.10 $\Delta \Longrightarrow \Delta'$, where $\Delta' = \{\mathsf{p} : \mathbb{M}_1', \mathsf{q} : \mathbb{M}_2'\}$. This implies that Δ' is consistent by Lemma 6.14.

Lemma 6.2(1) implies $\mathbb{P}_1 = P_1 \mid \mathbb{P}_1''$ and $\mathbb{P}_1' = P_1' \mid \mathbb{P}_1''$ where:

1. either $P_1 = ?\ell(x).Q_1$ and $P_1' = Q_1\{v/x\}$;
2. or $P_1 = Q + R$ and $Q \xrightarrow{?\ell(v)} Q'$ and $P_1' = Q'$.

By (1) and Lemma 6.6(1a) and (1c) $\mathbb{T}_1 = \mathsf{T} \mid \mathbb{T}_1''$ with $\mathsf{T} \leq ?\ell(S).\mathsf{T}_1$ and $\vdash P_1 : \mathsf{T}$.

Lemma 6.2(2) and typability of $\mathbb{P}$ imply $\mathbb{P}_2 = P_2 \mid \mathbb{P}_2''$ and $\mathbb{P}_2' = P_2' \mid \mathbb{P}_2''$ where: $P_2 = !\ell(\mathsf{e}).Q_2$ and $\mathsf{e} \downarrow v$ and $P_2' = Q_2$. Then by (1) and Lemma 6.6(1b) $\mathbb{T}_2 = !\ell(S').\mathsf{T}_2 \mid \mathbb{T}_2''$ and $\vdash \mathsf{e} : S'$ and

$$\vdash P_2' : \mathsf{T}_2 \tag{4}$$

By (1), (2), (3) and Definition 4.3 either $?\ell(S).\mathsf{T}_1 \leq |\mathsf{p}?\{\ell_i(S_i).\mathsf{M}_i\}_{i \in I}|$ or $?\ell(S).\mathsf{T}_1 \wedge \mathsf{T}' \leq |\mathsf{p}?\{\ell_i(S_i).\mathsf{M}_i\}_{i \in I}|$ and $!\ell(S').\mathsf{T}_2 \leq |\mathsf{q}!\{\ell_i(S_i).\mathsf{M}_i\}_{i \in I}|$. In both cases $S = S' = S_j$ and

$$\mathsf{T}_1 \leq |\mathsf{M}_j| \text{ and } \mathsf{T}_2 \leq |\mathsf{M}_j'| \tag{5}$$

Then Lemma 6.9 applied to $\vdash \mathsf{e} : S$ and $\mathsf{e} \downarrow v$ and $\vdash P_1 : \mathsf{T}$ gives

$$\vdash P_1' : \mathsf{T}_1 \tag{6}$$

From (4), (5) and (6) we get $\mathbb{P}_1' \propto \mathbb{M}_1'$ and $\mathbb{P}_2' \propto \mathbb{M}_2'$, then using rules [MP] and [NPAR] we derive $\vdash \mathsf{p} \ltimes \mathbb{M}_1'[\mathbb{P}_1'] \parallel \mathsf{q} \ltimes \mathbb{M}_2'[\mathbb{P}_2'] \rhd \Delta'$.

If $\mathsf{p} \ltimes \mathbb{M}[\mathbb{P}] \longrightarrow \mathsf{p} \ltimes \mathbb{M}[\mathbb{P}']$ by rule [TAU], then $\mathbb{P} = P \mid \mathbb{P}''$ and $P \xrightarrow{\tau} P'$ and $\mathbb{P}' = P' \mid \mathbb{P}''$. Lemma 6.6(3b) implies $\Delta = \{\mathsf{p} : \mathbb{M}\}$ and $\mathbb{P} \vdash \mathsf{p} : \mathbb{T}$ and $\mathbb{P} \propto \mathbb{M}$ and $\mathsf{p} \notin \mathsf{pa}(\mathbb{M})$. Lemma 6.6(2) gives $\mathbb{T} = \mathsf{T} \mid \mathbb{T}'$ and $P \vdash \mathsf{p} : \mathsf{T}$ and $\mathbb{P}'' \vdash \mathsf{p} : \mathbb{T}'$. By Lemma 6.18 $\vdash P' : \mathsf{T}'$ and $\mathsf{T}' \leq \mathsf{T}$. Using rule [PAR] we derive $\mathbb{P}' \vdash \mathsf{p} : \mathsf{T}' \mid \mathbb{T}'$. By Definition 4.3(2) $\mathbb{M} = \mathsf{M} \mid \mathbb{M}'$ and $P \propto \mathsf{M}$ and $\mathbb{P}'' \propto \mathbb{M}'$. Definition 4.3(1) gives $\mathsf{T} \leq |\mathsf{M}|$. Then by Definition 4.3(1) $P' \propto \mathsf{M}$ and by Definition 4.3(2) $\mathbb{P}' \propto \mathbb{M}$. We can then derive $\vdash \mathsf{p} \ltimes \mathbb{M}[\mathbb{P}'] \rhd \Delta$ by rule [MP].

(2) If the last applied rule is [OP] the proof is similar to that of rule [TAU].

Let the last applied rule be

$$\frac{\begin{array}{c} F(\sigma) = (\mathbb{G}, \mathscr{K}, \sigma') \quad \mathscr{A}' = \mathscr{A} \setminus \mathscr{K} \quad \mathscr{B} = \mathsf{pa}(\mathbb{G}) \setminus \mathscr{A} \\ \mathbb{M}_\mathsf{p}' = \mathsf{mon}(\mathsf{p}, \mathbb{M}_\mathsf{p}, \hat{\mathbb{G}}, \mathscr{K}) \quad \mathbb{P}_\mathsf{p}' = \mathsf{proc}(\mathsf{p}, \mathbb{P}_\mathsf{p}, \mathbb{M}_\mathsf{p}, \hat{\mathbb{G}}, \mathscr{K}, \mathscr{P}) \\ \forall \mathsf{p} \in \mathscr{B}. \mathbb{Q}_\mathsf{p} \in \mathscr{P} \;\&\; \mathbb{Q}_\mathsf{p} \propto \hat{\mathbb{G}} \upharpoonright \mathsf{p} \end{array}}{\begin{array}{c} \prod_{\mathsf{p} \in \mathscr{A}} \mathsf{p} \ltimes \mathbb{M}_\mathsf{p}[\mathbb{P}_\mathsf{p}] \between \sigma \between \mathscr{P} \between F \longrightarrow \\ \prod_{\mathsf{p} \in \mathscr{A}'} \mathsf{p} \ltimes \mathbb{M}_\mathsf{p}'[\mathbb{P}_\mathsf{p}'] \parallel \prod_{\mathsf{p} \in \mathscr{B}} \mathsf{p} \ltimes \hat{\mathbb{G}} \upharpoonright \mathsf{p}[\mathbb{Q}_\mathsf{p}] \between \sigma' \between \mathscr{P} \between F \end{array}} \text{ ADAPT}$$

By Lemma 6.6(5) $\vdash \prod_{\mathsf{p} \in \mathscr{A}} \mathsf{p} \ltimes \mathbb{M}_\mathsf{p}[\mathbb{P}_\mathsf{p}] \rhd \Delta$ and $\mathsf{comp}(\Delta)$. Let $\mathbb{G}'$ be the global type that shows $\mathsf{comp}(\Delta)$, i.e. such that $\mathbb{G}' \upharpoonright \mathsf{p} = \mathbb{M}_\mathsf{p}$ for all $\mathsf{p} \in \mathscr{A}$. By definition of proc we get $\mathbb{P}_\mathsf{p}' \propto \mathbb{M}_\mathsf{p}'$ for all $\mathsf{p} \in \mathscr{A}'$. Using rule [MP] we can then derive

$$\vdash \mathsf{p} \ltimes \mathbb{M}_\mathsf{p}'[\mathbb{P}_\mathsf{p}'] \rhd \{\mathsf{p} : \mathbb{M}_\mathsf{p}'\} \text{ for all } \mathsf{p} \in \mathscr{A}' \tag{7}$$

By construction $\mathbb{Q}_\mathsf{p} \propto \hat{\mathbb{G}} \upharpoonright \mathsf{p}$ for all $\mathsf{p} \in \mathscr{B}$, so we can also derive

$$\vdash \mathsf{p} \ltimes \hat{\mathbb{G}} \upharpoonright \mathsf{p}[\mathbb{Q}_\mathsf{p}] \rhd \{\mathsf{p} : \hat{\mathbb{G}} \upharpoonright \mathsf{p}\} \text{ for all } \mathsf{p} \in \mathscr{B} \tag{8}$$

Using the global type $\hat{\mathbb{G}} \mid (\mathbb{G}' \setminus \mathsf{pa}(\mathbb{G})) \between \mathscr{K}$ to show the consistency of the session typing $\Delta' = \{\mathsf{p} : \mathbb{M}_\mathsf{p}' \mid \mathsf{p} \in \mathscr{A}'\} \cup \{\mathsf{p} : \hat{\mathbb{G}} \upharpoonright \mathsf{p} \mid \mathsf{p} \in \mathscr{B}\}$ we can apply rule [NPAR] to (7) and (8). In fact $\mathbb{M}_\mathsf{p} = \mathbb{G}' \upharpoonright \mathsf{p}$ for all $\mathsf{p} \in \mathscr{A}$ and then by Lemma 6.16(4) $\mathbb{M}_\mathsf{p}' = (\hat{\mathbb{G}} \mid (\mathbb{G}' \setminus \mathsf{pa}(\mathbb{G})) \between \mathscr{K}) \upharpoonright \mathsf{p}$ for all $\mathsf{p} \in \mathscr{A}'$. Moreover $\hat{\mathbb{G}} \upharpoonright \mathsf{p} = (\hat{\mathbb{G}} \mid (\mathbb{G}' \setminus \mathsf{pa}(\mathbb{G})) \between \mathscr{K}) \upharpoonright \mathsf{p}$ for all $\mathsf{p} \in \mathscr{B}$. The same global type shows $\mathsf{comp}(\Delta')$, so we can conclude applying rule [SES].

If the last applied rule is [SN] the proof follows from (1). If the last applied rule is [CTX] the proof is standard.

□

6.5. Main Results

Starting from a well-typed session the reduction rules produce only monitored processes $\mathbb{M}[\mathbb{P}]$ which satisfy $\mathbb{P} \propto \mathbb{M}$. The crucial case is rule [ADAPT], where the mapping proc builds the processes so that the adequacy between processes and monitors is guaranteed.

Theorem 6.20. *If $\mathscr{S}$ is a well typed session and $\mathscr{S} \longrightarrow \mathscr{S}'$, then all monitored processes in $\mathscr{S}'$ satisfy the adequacy condition.*

Proof. If $\mathscr{S}$ is a well typed session, then we can derive $\vdash \mathscr{S} \rhd \Delta$. By Theorem 6.19(2) we get $\vdash \mathscr{S}' \rhd \Delta'$. The Inversion Lemma implies that if $\mathbb{M}[\mathbb{P}]$ occurs in $\mathscr{S}'$, then we can type it using rule [MP]. So we conclude $\mathbb{P} \propto \mathbb{M}$. □

A set of monitors $\mathscr{M}$ is *closed* if:

- $\mathsf{M} \in \mathscr{M}$ implies $\mathsf{M} \setminus\!\setminus \mathscr{K} \in \mathscr{M}$ for an arbitrary $\mathscr{K}$;
- $\mathbb{M} \in \mathscr{M}$ and $\mathbb{M}' \in \mathscr{M}$ imply $\mathbb{M} \mid \mathbb{M}' \in \mathscr{M}$.

We can also guarantee progress of sessions under a natural condition on the collection $\mathscr{P}$. A collection $\mathscr{P}$ is *complete for a closed set of monitors $\mathscr{M}$* if there are adequate processes in $\mathscr{P}$ for every $\mathbb{M} \in \mathscr{M}$.

Theorem 6.21. *If $\mathscr{P}$ is complete for $\mathscr{M}$ and $\mathscr{S}$ is a well typed session with monitors in $\mathscr{M}$ and the adaptation function F only produces global types whose projections are in $\mathscr{M}$, then $\mathscr{S}$ has progress, i.e. every input or output monitored process will eventually communicate.*

Proof. As proved in [13], a single multiparty session in a standard calculus with global and session types, like the calculus in [1], always enjoys progress whenever it is well typed. In fact, by the Subject Reduction Theorem (Theorem 6.19), reduction preserves well-typedness of sessions and the consistency of session typings. Moreover, all required session participants are present. Thus, all communications among participants will take place, in the order prescribed by the global type.

It is easy to see that our calculus could be mapped to the calculus of [1] while maintaining typability, the main difference being in the reduction rules. Hence, the only rule that could jeopardise progress is [ADAPT]. This does not happen, since the required process are in $\mathscr{P}$ by completeness, and the typability of the obtained session assures that all communications obey a single global type. Therefore, our calculus has progress under the required conditions. □

7. Related Work and Conclusion

Our approach builds on [4], where a calculus based on global types, monitors and processes similar to the present one was introduced. There are two main points of departure from that work. First, in the calculus of [4], there is a periodic check of the global status, written in the global type, which decides *when* the adaptation can take place. Second, in [4] any adaptation involves all the participants, by requiring a global

new reconfiguration of the whole system. In contrast, in the present calculus adaptation is only triggered by dynamic changes in control data, so giving rise to unpredictable reconfiguration steps. Furthermore, reconfiguration can involve a partial set of communications only, when only a part of the behaviour must be changed. The main technical novelty allowing these features is the parallel composition of monitors. Moreover, as remarked in the introduction, communication is synchronous here and asynchronous in [4].

Works addressing adaptation for multiparty communications include [14], [15], [16], [17] and [18]. In the paper [14] global and session types are used to guarantee deadlock-freedom in a calculus of multiparty sessions with asynchronous communications. Only part of the running code is updated. Two different conditions are given for ensuring liveness. The first condition requires that all channel queues are empty before updating. The second condition requires a partial order between the session participants with a unique minimal element. The participants are then updated following this order. We do not need such conditions for progress, since communication is synchronous and adaptation is done in one single big step. The work [15] enhances a choreographic language with constructs defining adaptation scopes and dynamic code update; an associated endpoint language for local descriptions, and a projection mechanism for obtaining (low-level) endpoint specifications from (high-level) choreographies are also described. A typing discipline for these languages is left for future work. The paper [16] proposes a choreographic language for distributed applications. Adaptation follows a rule-based approach, in which all interactions, under all possible changes produced by the adaptation rules, proceed as prescribed by an abstract model. In particular, the system is deadlock-free by construction. The adaptive system is composed by interacting participants deployed on different locations, each executing its own code. The calculus of [17] is inspired by [4], the main difference being that security violations trigger adaptation mechanisms that prevent violations to occur and/or to propagate their effect in the systems. An event based approach is proposed in [18]. Eventful constructors handle adaptation routines checking sessions by means of channel types. Adaptation in [18] cannot explicitly kill session participants as we do.

There is substantial difference between our monitors and the run-time enforcement monitors of [19], where monitors terminate the execution if it is about to violate the security policy being enforced.

In the present paper we address the adaptation by considering one single session, in order to simplify notations in the formal proofs. Indeed, our framework is expressive enough to be smoothly extended to the case of many sessions running in parallel, by simply annotating communications of each participant with the session name as in [4]. Future work includes the extension of our calculus to many self-adapting sessions running in parallel, with session interleaving. We plan to consider application-driven instances of our adaptation mechanism, in which concrete adaptation functions are exploited. This can require the explicit addition of local data to each participant. In this way we will be able to model internal interactions, based on local data, among participants. Using both internal interactions and communications among participants would enable the application of our framework to more realistic cases of study. It would be interesting to consider adaptation on the fly [20] in the context of communication centred programming. Finally, we intend to investigate whether our approach

can be useful in providing a formal model to mechanisms of code hot-swapping in programming languages (such as Erlang).

Acknowledgments. We thank Cinzia Di Giusto for interesting discussions related to this paper. We are grateful to the anonymous reviewers for their helpful comments and suggestions.

References

[1] K. Honda, N. Yoshida, M. Carbone, Multiparty Asynchronous Session Types, in: POPL'08, ACM Press, 2008, pp. 273–284. doi:10.1145/1328438.1328472.

[2] M. Carbone, K. Honda, N. Yoshida, Structured Communication-Centered Programming for Web Services, ACM Transactions on Programming Languages and Systems 34 (2) (2012) 8:1–8:78. doi:10.1145/2220365.2220367.

[3] R. Bruni, A. Corradini, F. Gadducci, A. Lluch-Lafuente, A. Vandin, A Conceptual Framework for Adaptation, in: FASE'12, Vol. 7212 of LNCS, Springer, 2012, pp. 240–254. doi:10.1007/978-3-642-28872-2_17.

[4] M. Coppo, M. Dezani-Ciancaglini, B. Venneri, Self-Adaptive Multiparty Sessions, Service Oriented Computing and Applications 9 (3-4) (2015) 249–268. doi:10.1007/s11761-014-0171-9.

[5] M. Coppo, M. Dezani-Ciancaglini, B. Venneri, Parallel Monitors for Self-adaptive Sessions, in: PLACES'16, Vol. 211 of EPTCS, 2016, pp. 25–36. doi:10.4204/EPTCS.211.3.

[6] P.-M. Deniélou, N. Yoshida, Dynamic Multirole Session Types, in: POPL'11, ACM Press, 2011, pp. 435–446. doi:10.1145/1926385.1926435.

[7] L. Bocchi, T.-C. Chen, R. Demangeon, K. Honda, N. Yoshida, Monitoring Networks through Multiparty Session Types, in: FMOODS/FORTE'13, Vol. 7892 of LNCS, Springer, 2013, pp. 50–65. doi:10.1007/978-3-642-38592-6_5.

[8] L. Bettini, M. Coppo, L. D'Antoni, M. De Luca, M. Dezani-Ciancaglini, N. Yoshida, Global Progress in Dynamically Interleaved Multiparty Sessions, in: CONCUR'08, Vol. 5201 of LNCS, Springer, 2008, pp. 418–433. doi:10.1007/978-3-540-85361-9_33.

[9] H. Hüttel, I. Lanese, V. T. Vasconcelos, L. Caires, M. Carbone, P.-M. Deniélou, D. Mostrous, L. Padovani, A. Ravara, E. Tuosto, H. T. Vieira, G. Zavattaro, Foundations of Session Types and Behavioural Contracts, ACM Computing Surveys 49 (1) (2016) 3:1–3:36. doi:10.1145/2873052.

[10] K. Honda, V. T. Vasconcelos, M. Kubo, Language Primitives and Type Disciplines for Structured Communication-based Programming, in: ESOP'98, Vol. 1381 of LNCS, Springer, 1998, pp. 22–138. doi:10.1007/BFb0053567.

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

[12] S. Gay, M. Hole, Subtyping for Session Types in the Pi Calculus, Acta Informatica 42 (2/3) (2005) 191–225. doi:10.1007/s00236-005-0177-z.

[13] M. Coppo, M. Dezani-Ciancaglini, N. Yoshida, L. Padovani, Global Progress for Dynamically Interleaved Multiparty Sessions, Mathematical Structures in Computer Science 26 (2) (2016) 238–302. doi:10.1017/S0960129514000188.

[14] G. Anderson, J. Rathke, Dynamic Software Update for Message Passing Programs, in: APLAS'12, Vol. 7705 of LNCS, Springer, 2012, pp. 207–222. doi:10.1007/978-3-642-35182-2_15.

[15] M. Bravetti, M. Carbone, T. T. Hildebrandt, I. Lanese, J. Mauro, J. A. Pérez, G. Zavattaro, Towards Global and Local Types for Adaptation, in: SEFM'13, Vol. 8368 of LNCS, Springer, 2014, pp. 3–14. doi:10.1007/978-3-319-05032-4_1.

[16] M. Dalla Preda, S. Giallorenzo, I. Lanese, J. Mauro, M. Gabbrielli, AIOCJ: A Choreographic Framework for Safe Adaptive Distributed Applications, in: SLE'14, Vol. 8706 of LNCS, Springer, 2014, pp. 161–170. doi:10.1007/978-3-319-11245-9_9.

[17] I. Castellani, M. Dezani-Ciancaglini, J. A. Pérez, Self-adaptation and Secure Information Flow in Multiparty Communications, Formal Aspects of Computing 28 (4) (2016) 669–696. doi:10.1007/s00165-016-0381-3.

[18] C. Di Giusto, J. A. Pérez, Event-based Run-time Adaptation in Communication-centric Systems, Formal Aspects of Computing 28 (4) (2016) 531–566. doi:10.1007/s00165-016-0377-z.

[19] F. B. Schneider, Enforceable Security Policies, ACM Transactions on Information and System Security 3 (1) (2000) 30–50. doi:10.1145/353323.353382.

[20] A. Bucchiarone, A. Marconi, C. A. Mezzina, M. Pistore, H. Raik, On-the-Fly Adaptation of Dynamic Service-Based Systems: Incrementality, Reduction and Reuse, in: ICSOC'13, Vol. 8274 of LNCS, Springer, 2013, pp. 146–161. doi:10.1007/978-3-642-45005-1_11.