A Coordination Protocol Language for Power Grid Operation Control[☆]

Yehia Abd Alrahman[a], Hugo Torres Vieira[b]

[a] *University of Gothenburg, Chalmers University of Technology, Gothenburg, Sweden*
[b] *IMT School for Advanced Studies Lucca, Lucca, Italy*

Abstract

Future power distribution grids will comprise a large number of components, each potentially able to carry out operations autonomously. Clearly, in order to ensure safe operation of the grid, individual operations must be coordinated among the different components. Since operation safety is a global property, modelling component coordination typically involves reasoning about systems at a global level. In this paper, we propose a language for specifying grid operation control protocols from a global point of view. In our model, operation control is yielded in communications driven by both the grid topology and by state-based information, features captured by novel language principles previously unexplored. We show how the global specifications can be used to automatically generate local controllers of individual components, and that the distributed implementation yielded by such controllers operationally corresponds to the global specification. We showcase our development by modelling a fault management scenario in power grids.

Keywords: Power Grids, Interaction Protocols, Process Calculus

1. Introduction

Modern power grids comprise a large number of geographically dispersed components. For example, the transmission lines of the North American power grid link all electricity generation and distribution in the continent [1]. Moreover, technologically challenging features such as distributed generation are uprising, for instance due to the integration of renewable energy sources (cf. [2]). Clearly, relying on centralised control and protection systems cannot cope with the level of dynamicity and demands of such large and complex distributed systems. In particular to what concerns mechanisms that support error recovery and power flow management, it is crucial to equip power grids with decentralised operation (cf. [3, 4, 5, 6]).

The key idea is to consider the power grid as a collection of independent and autonomous substations, collaborating to achieve a desired global goal. It is then necessary to equip each substation with an independent controller that is able to communicate and cooperate with others, forming a large distributed computing system. Each controller must be connected to sensors associated with its own substation so that it can assess its own status and report them to its neighbouring controllers via communication paths. The communication paths of the controllers should follow the electrical connection paths. In this way, each substation can have (at least) a local view of the grid, represented by the connections to its neighbours.

Such distinguishing features of power grids call for new methods and tools to address the posed technological challenges, considering system reliability is of utmost concern given the critical nature of such systems. In the literature we may find a number of research efforts that have addressed power grids in particular, apart from approaches that address cyber-physical systems in a more general way. Recent proposals build on a variety of techniques, for instance (without exhausting the broadness of the literature): multi-agent systems

[☆] Yehia Abd Alrahman is funded by the ERC consolidator grant D-SynMA under the European Union's Horizon 2020 research and innovation programme (grant agreement No 772459).

Email addresses: yehia.abd.alrahman@gu.se (Yehia Abd Alrahman), hugotvieira@imtlucca.it (Hugo Torres Vieira)

(e.g., [7, 8]); simulation and testing (e.g., [9, 10, 11]); game theory (e.g., [12, 13]); optimisation decomposition methods [14, 15]; and control systems (e.g., [16, 17, 18]). However, to the best of our knowledge, there are no proposals that provide programming language support for the development of power grid operation control (with the noticeable exception of [19]), in particular to what concerns techniques that address the coordination of the operation of the distributed controllers in a certifiably reliable way.

In this paper, we propose a specification language for designing coordination protocols that support decentralised power grid operation control. Our development targets the specific features mentioned above, namely considering that substations can communicate with neighbours (in particular the ones related by a power supply relation), and where communication is driven by the substation local status (capturing sensor information). For the purpose of operation coordination, we consider that control is delegated in communications, hence substations are activated by receiving messages and yield control by sending messages. We build on proposals for communication-centred systems that consider coordination protocols are specified in a global perspective (e.g., [20, 21]), i.e., considering the set of interactions as a whole, and tailor our language so as to embed the identified distinguishing features. As a result, our language presents novel principles that are specially fit for power grids and other cyber-physical systems with common characteristics.

In order to support the rigorous validation of protocol specifications, we show how the protocol specifications can be combined with a specification of the power grid configuration so as to yield an operational model that captures system evolutions in a rigorous way. Moreover, since the protocol language serves the purpose of specifying the overall operation control it is crucial to provide mechanisms that relate the global protocol with the distributed substations controller code. For this purpose, we show how the protocol specifications can be used to synthesise the controller code in an automatic way. Noticeably, the controller code is based on reactive definitions that are continuously able to react to message reception, provided that the state conditions enable both the reception and the reaction, and where reactions consist in message emission.

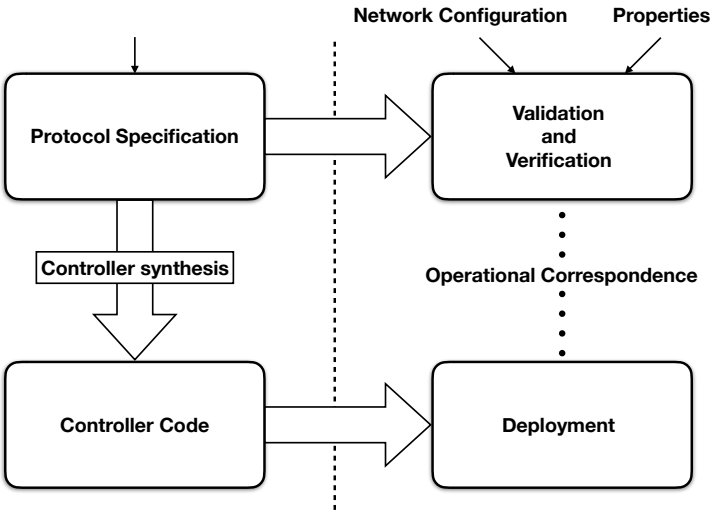
We also show how the controller code can be combined with a specification of the power grid configuration (exactly like the one used for global protocols specifications) so as to yield a distributed operational model. We then validate the developed automatic controller code synthesis, by means of an operational correspondence result that establishes a one-to-one correspondence between evolutions in the global model and in the distributed model. This result provides support for verification techniques that reason at the level of the global specification, which is more amenable to verify (global) system-level properties, since we can ensure that any property that holds for the global specification also holds for the distributed implementation.

We illustrate how this work may be exploited by means of the illustration in Fig. 1 that shows the development stages we are considering. On the left hand side, we find the code development pipeline, starting by the protocols specifications that must be provided. In contrast, controller code is automatically generated by the synthesis process. On the right hand side, we see how the two artefacts can be used, namely the protocol specifications can be used for the purpose of validation and verification, considering that the network configuration and properties to be verified are provided. The automatically generated controller code can be used in the deployment phase, left underspecified. The operational correspondence result allows to relate the validation and verification carried out using the global model and the actual deployed implementation.

We remark that our model does not address all the relevant dimensions of the communication infrastructure of power grids (for instance, we do not handle communication failures nor security concerns) that may possibly be integrated as a separate layer (which is often the case for the two mentioned dimensions). We also do not aim at a full-fledged model of power grids, and we abstract away from the physical infrastructure by considering a minimal logical interface given by means of substation local state information. Our goal here is to provide a language that allows for the high-level specification of the power grid operation control logic that addresses specific features in a central way. We recognise that even directly at the level of the operation control logic other dimensions may be necessary, for instance when the coordination requires accounting for failures (as explored in [19] relying on the protocol language introduced in [22]), but we nevertheless believe that our language principles can be exploited in combination with other approaches.

In the remainder of this section, we informally introduce and motivate our design choices by pointing to a fault management scenario for the sake of a more intuitive reading. In Sect. 2 we present the syntax and semantics of the global language. In Sect. 3, we return to the fault management scenario so as to show the programming flavour considering a case study from the realm of power grids. In Sect. 4 we introduce the

Figure 1: Development Stages



distributed model, used as the target language in Sect. 5 where we show how to automatically synthesise the behaviour of individual controllers of substations given the global specification of the protocol. The operational correspondence between global and distributed models is attested in our main result (Theorem 5.10). Finally, in Sect. 6, we comment on research directions and present related work.

1.1. Operation Control Protocols, Informally

We consider a scenario involving a power distribution grid where a fault occurs in one of the transmission power lines, and a recovery protocol that locates and isolates the fault by means of interactions among the nodes (substations) in the grid.

A central notion in our proposal is that nodes involved in such a protocol yield control by means of synchronisation actions. For example, consider a node that is willing to Locate the fault and another one that can react and continue the task of locating the fault. The two nodes may synchronise on action Locate and the enabling node yields the control to the reacting one. We therefore consider that *active* nodes enable synchronisations and, as a consequence of a synchronisation, transfer the *active* role to the reacting node. So, for example, after some node 1 synchronises with some node 2 on Locate, node 2 may enable the following interaction, e.g., a synchronisation with some node 3 also on Locate.

We may therefore specify protocols as a (structured) set of interactions without identifying the actual nodes involved a priori, since these are determined operationally due to the transference of the active role in synchronisations. We write $[\mathsf{Locate}^{\text{-}}]\_P$ to specify a protocol stating that a synchronisation on Locate is to take place first, after which the protocol proceeds as specified by P (for now we abstract from the remaining elements using _). Also, we write $(1)P'$ to specify that node 1 is active on protocol P'. So, by $(1)[\mathsf{Locate}^{\text{-}}]\_P$ we represent that node 1 may enable the synchronisation on Locate. Furthermore, we may write $(1)[\mathsf{Locate}^{\text{-}}]\_P \rightarrow [\mathsf{Locate}^{\text{-}}]\_(2)P$ to represent such a synchronisation step ($\rightarrow$) between nodes 1 and 2, where node 1 yields the control and node 2 is activated to carry out the continuation protocol P. This allows to capture the transference of the active role in the different stages of the protocol.

Interaction in our model is driven by the network communication topology that accounts for radial power supply configurations (i.e, tree-like structures where root nodes provide power to the respective subtrees), and for a notion of proximity (so as to capture, e.g., backup links and address network reconfigurations). We therefore consider that nodes can interact if they are in a provide/receive power relation or in a neighbouring relation. To this end, synchronisation specifications include a direction to determine the target of the

synchronisation. For example, by $[\mathsf{Locate}^{\star}]_{\_}$ we represent that Locate targets all (child) nodes ($\star$) that receive power from the enabling node, used in the scenario by the root(s) to search for the fault in the power supply (sub)tree. Also, by $[\mathsf{Recover}^{\blacktriangle}]_{\_}$ we specify that Recover targets the power provider ($\blacktriangle$) of the enabling node (i.e., the parent node), used in the scenario to signal when the fault has been found. Finally, by $[\mathsf{Power}^{\blacktriangleright}]_{\_}$ we represent that Power targets a neighbouring node ($\blacktriangleright$), used in the scenario for capturing power restoration and associated network reconfiguration.

Combining two of the above example synchronisations by means of recursion and summation ($+$), we may then write a simplified fault management protocol

$$\textsc{Simple} \triangleq \mathbf{rec}\ X.([\mathsf{Locate}^{\star}]^{c_1}_{c_1 \vee c_2} X + [\mathsf{Recover}^{\blacktriangle}]^{c_2}_{\mathsf{tt}} \mathbf{0}) \tag{1}$$

which specifies an alternative between Locate and Recover synchronisations, where in the former case the protocol starts over and in the latter case terminates. Also, to support a more fine grained description of protocols, the synchronisation actions specify conditions (omitted previously with _) that must hold for both enabling and reacting nodes in order for a synchronisation to take place. For example, only nodes that are at the root of a (sub)tree that has a fault may enable a synchronisation on Locate, which is captured in condition c_1 (left unspecified here). Also, only nodes that satisfy such condition (c_1) or that are without power supply (which is captured in condition c_2) can react to Locate. For Recover, the specification says that only nodes that are without power supply (c_2) can enable the synchronisation, while any node can react to it (captured by condition true tt). The general idea is that the Locate synchronisations propagate throughout the power supply tree, one level per synchronisation, up to the point the fault is located and synchronisation Recover leads to termination.

The combination of the summation and of synchronisation conditions thus allows for a fine-grained specification of the operation control protocol. Furthermore, the conditions specified for synchronisations are checked against the state of nodes, so state information is accounted for in our model. This implies that a node that is active to carry out a protocol might actually not be enabled by the conditions. For instance, if 1 is not a root of a (sub)tree with a fault (hence does not satisfy c_1), and is not without power supply (hence does not satisfy c_2) then the configuration (1)Simple where 1 is active to carry our Simple has no possible synchronisations (since 1 does not satisfy the conditions associated to Locate and Recover). Notice however that a state change (e.g., due to the occurrence of a fault) may trigger the synchronisations, so we may consider 1 is actively waiting to carry out Simple once the prescribed conditions hold.

To represent the dynamics of systems, the state of nodes may evolve throughout the stages of the operation protocol. We therefore consider that synchronisation actions may have side-effects on the state of the nodes that synchronise. This allows to avoid introducing specialised primitives and simplifies reasoning on protocols, given the atomicity of the synchronisation and side-effect in one step.

The general principles described above, identified in the context of a fault management scenario, guided the design of the language presented in the next section. Although this is not the case for a single fault, we remark that our language addresses scenarios where several nodes may be active simultaneously in possibly different stages of the protocol. A detailed account of the fault management scenario is presented in Sect. 3, including a protocol designed to account for configurations with several faults.

2. A Model for Operation Control Protocols

The syntax of the language is given in Table 1, where we assume a set of node identifiers ($id, \ldots$), synchronisation action labels ($f, \ldots$), and logical conditions ($c, i, o, \ldots$). Protocols ($P, Q, \ldots$) combine static specifications and the *active* node construct. The latter is denoted by $(id)P$, representing that node id is active to carry out the protocol P. Static specifications include termination $\mathbf{0}$, fork $P \,|\, Q$ which says that both P and Q are to be carried out, and infinite behaviour defined in terms of the recursion $\mathbf{rec}\ X.P$ and recursion variable X, with the usual meaning. Finally, static specifications include synchronisation summations ($S, \ldots$), where $S_1 + S_2$ says that either S_1 is to be carried out or S_2 (exclusively), and where $[f^d]^o_i P$ represents a synchronisation action: a node active on $[f^d]^o_i P$ that satisfies condition o may synchronise on f with the node(s) identified by the direction d for which condition i holds, leading to the activation of

Table 1: Global Language Syntax

(Protocol)	$P ::= \mathbf{0} \quad \mid \quad P \,\vert\, P \quad \mid \quad \mathbf{rec}\ X.P \quad \mid \quad X \quad \mid \quad S \quad \mid \quad (id)P$
(Summation)	$S ::= [f^d]^o_i P \quad \mid \quad S + S$
(Direction)	$d ::= \star \quad \mid \quad \blacktriangle \quad \mid \quad \blacktriangleright \quad \mid \quad \bullet$

the latter node(s) on protocol P. Intuitively, the node active on $[f^d]^o_i P$ enables the synchronisation, which results in the reaction of the targeted nodes that are activated to carry out the continuation protocol P.

A direction d specifies the target(s) of a synchronisation action, that may be of four kinds: $\star$ targets all (children) nodes to which the enabling node provides power to; $\blacktriangle$ targets the power provider (i.e., the parent) of the enabler; $\blacktriangleright$ targets a neighbour of the enabler; and $\bullet$ targets the enabler itself, used to capture local computation steps. We remark on the $\star$ direction given its particular nature: since one node can supply power to several others, synchronisations with $\star$ direction may actually involve several reacting nodes, up to the respective condition. We interpret $\star$ synchronisations as broadcasts, in the sense that we take $\star$ to target all (direct) child nodes that satisfy the reacting condition, which hence comprises the empty set (in case the node has no children or none of them satisfy the condition). The interpretation of binary interaction differs, as synchronisation is only possible if the identified target node satisfies the condition.

Example 2.1. *Consider the following protocol, assuming definitions for* Recovery, Isolation *and* Restoration

$$(id)([\mathsf{Locate}^\star]^{o_1}_{i_1}\textsc{Recovery} + [\mathsf{End}^\bullet]^{o_2}_{i_2}(\textsc{Isolation} \,|\, \textsc{Restoration}))$$

which specifies node id is active to synchronise on Locate *or* End*, exclusively.*

Example 2.1 already hints on the two ways of introducing concurrency in our model. On the one hand, broadcast can lead to the activation of several nodes: in the example, each one of the nodes reacting to Locate will carry out the Recovery protocol. On the other hand, the fork construct allows for a single node to carry out two subprotocols, possibly activating different nodes in the continuation: in the example, a node active to carry out (Isolation | Restoration) may synchronise with different nodes in each one of the branches.

In order to define the semantics of the language, we introduce structural congruence that, in particular, captures the relation between the active node construct $(id)P$ and protocol specifications (including the active node construct itself and the fork construct). Structural congruence is the least congruence relation on protocols that satisfies the rules given in Table 2. The first set of rules captures expected principles, namely that fork and summation are associative and commutative, and that fork has identity element $\mathbf{0}$ (we remark that the syntax excludes $\mathbf{0}$ as a branch in summations). Rule $(id)(P \,|\, Q) \equiv (id)P \,|\, (id)Q$ captures the interpretation of the fork construct: it is equivalent to specify that a node id is active on a fork, and to specify that a fork has id active on both branches. Rule $(id_1)(id_2)P \equiv (id_2)(id_1)P$ says that active nodes can be permuted and rule $(id)\mathbf{0} \equiv \mathbf{0}$ says that a node active to carry out termination is equivalent to the termination itself. Structural congruence together with reduction define the operational semantics of the model. Intuitively, structural congruence rewriting allows active nodes to "float" in the term towards the synchronisation actions.

Example 2.2. *Considering the active node distribution in a fork, we have that*

$$[\mathsf{Locate}^\star]^{o_1}_{i_1}\textsc{Recovery} + [\mathsf{End}^\bullet]^{o_2}_{i_2}(id)(\textsc{Isolation} \,|\, \textsc{Restoration})$$

is structural congruent to

$$[\mathsf{Locate}^\star]^{o_1}_{i_1}\textsc{Recovery} + [\mathsf{End}^\bullet]^{o_2}_{i_2}((id)\textsc{Isolation} \,|\, (id)\textsc{Restoration})$$

The definition of reduction depends on the network topology and on the fact that nodes satisfy certain logical conditions. We consider state information for each node so as to capture both "local" information

Table 2: Structural Congruence

$P \mid \mathbf{0} \equiv P$	$P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$	$P_1 \mid P_2 \equiv P_2 \mid P_1$
$\mathbf{rec}\ X.P \equiv P[\mathbf{rec}\ X.P/X]$	$S_1 + (S_2 + S_3) \equiv (S_1 + S_2) + S_3$	$S_1 + S_2 \equiv S_2 + S_1$
$(id)(P \mid Q) \equiv (id)P \mid (id)Q$	$(id_1)(id_2)P \equiv (id_2)(id_1)P$	$(id)\mathbf{0} \equiv \mathbf{0}$

about the topology (such as the identities of the power provider and of the set of neighbours) and other information relevant for condition assessment (such as the status of the power supply). The network state, denoted by Δ, is a mapping from node identifiers to states, where a state, denoted by s, is a register $id[id', t, n, k, a, e]$ containing the following information: id is the node identifier; id' identifies the power provider; t captures the status of the input power connection; n is the set of identifiers of neighbouring nodes; k is the power supply capacity of the node; a is the number of active power supply links (i.e., the number of nodes that receive power from this one); and e is the number of power supply links that are in a faulty state.

As mentioned in Section 1.1, we check conditions against states for the purpose of allowing synchronisations. Given a state s we denote by $s \models c$ that state s satisfies condition c, where we leave the underlying logic unspecified. For example, we may say that $s \models (k > 0)$ to check that s has capacity greater than 0.

Also mentioned in Section 1.1 is the notion of side-effects, in the sense that synchronisation actions may result in state changes so as to model system evolution. We use the following operations to apply side-effects:

$f^d!(s, id)$**:** is an operation that modifies state s of the active node according to the side-effects of enabling f^d and considering id is the identity of the reactive node.

$f^d?(s, id)$**:** is an operation that modifies state s of the reactive node according to the side-effects of enabling f^d and considering id is the identity of the active node.

$\texttt{upd}(id, id', f^d, \Delta)$ **:** is an operation that yields the network state obtained by updating Δ considering that node id synchronises on f with node id', hence the update regards the side-effects of f in the involved nodes. Namely, given $\Delta = (\Delta', id \mapsto s, id' \mapsto s')$ we have that $\texttt{upd}(id, id', f^d, \Delta)$ is defined as $(\Delta', id \mapsto f^d!(s, id'), id' \mapsto f^d?(s', id))$.

We consider side-effects only for binary synchronisations (for $d \in \{\blacktriangleright, \blacktriangle\}$), but state changes could also be considered for other directions in similar lines.

The definition of reduction relies on an auxiliary operation, denoted $d(\Delta, id)$, that yields the recipient(s) of a synchronisation action, given the direction d, the network state Δ, and the enabler of the action id. The operation, defined as follows, thus yields the power provider of the node in case the direction is $\blacktriangle$, (any) one of the neighbours in case the direction is $\blacktriangleright$, all the nodes that have as parent the enabler in case the direction is $\star$, and is undefined for direction $\bullet$.

$$
\begin{array}{lcl}
\blacktriangle(\Delta, id) & \triangleq & id' \quad (\text{if } \Delta(id) = id[id', t, n, k, a, e]) \\
\blacktriangleright(\Delta, id) & \triangleq & id' \quad (\text{if } \Delta(id) = id[id'', t, n, k, a, e] \text{ and } id' \in n) \\
\star(\Delta, id) & \triangleq & \{id' \mid \blacktriangle(\Delta, id') = id\} \\
\bullet(\Delta, id) & \triangleq & \textit{undefined}
\end{array}
$$

The reduction relation is given in terms of configurations consisting of a protocol P and a mapping Δ, for which we use $\Delta; P$ to denote the combination. By $\Delta; P \to \Delta'; P'$ we represent that configuration $\Delta; P$ evolves in one step to configuration $\Delta'; P'$, potentially involving state changes (Δ and Δ' may differ) and (necessarily) involving a step in the protocol from P to P'.

Reduction is defined as the least relation that satisfies the rules shown in Table 3, briefly described next. Rule Bin captures the case of binary interaction, hence when the direction (d) of the synchronisation action targets either the parent ($\blacktriangle$) or a neighbour ($\blacktriangleright$). Protocol $(id)([f^d]^o_i P)$ says that node id is active for synchronisation on $[f^d]^o_i P$, so id can enable a synchronisation on f provided that the state of id satisfies condition o, as specified in premise $\Delta(id) \models o$. Furthermore, the reacting node id', specified in the premise $d(\Delta, id) = id'$, is required to satisfy condition i. In such circumstances, the configuration can evolve and the

Table 3: Reduction Rules

$$\frac{d \in \{\blacktriangle, \blacktriangleright\} \quad \Delta(id) \models o \quad d(\Delta, id) = id' \quad \Delta(id') \models i \qquad \Delta' = \mathtt{upd}(id, id', f^d, \Delta)}{\Delta; (id)([f^d]_i^o P) \to \Delta'; [f^d]_i^o((id')P)} \text{ (Bin)}$$

$$\frac{\Delta(id) \models o \quad \star(\Delta, id) = I' \quad I = \{id' \mid id' \in I' \wedge \Delta(id') \models i\}}{\Delta; (id)([f^\star]_i^o P) \to \Delta; [f^\star]_i^o((\tilde{I})P)} \text{ (Brd)}$$

$$\frac{\Delta(id) \models o \quad \Delta(id) \models i}{\Delta; (id)([f^\bullet]_i^o P) \to \Delta; [f^\bullet]_i^o((id)P)} \text{ (Loc)} \qquad \frac{\Delta; (id)S_1 \to \Delta'; S_1'}{\Delta; (id)(S_1 + S_2) \to \Delta'; S_1' + S_2} \text{ (IdSum)}$$

$$\frac{\Delta; P \to \Delta'; P'}{\Delta; [f^d]_i^o P \to \Delta'; [f^d]_i^o P'} \text{ (Synch)} \qquad \frac{\Delta; P_1 \to \Delta'; P_1'}{\Delta; P_1 + P_2 \to \Delta'; P_1' + P_2} \text{ (Sum)} \qquad \frac{\Delta; P_1 \to \Delta'; P_1'}{\Delta; P_1 \mid P_2 \to \Delta'; P_1' \mid P_2} \text{ (Par)}$$

$$\frac{\Delta; P \to \Delta'; P'}{\Delta; (id)P \to \Delta'; (id)P'} \text{ (Id)} \qquad \frac{P \equiv P' \qquad \Delta; P' \to \Delta'; Q' \qquad Q' \equiv Q}{\Delta; P \to \Delta'; Q} \text{ (Struct)}$$

resulting state is obtained by updating the states of the involved nodes considering the side-effects of the synchronisation, and where the resulting protocol $[f^d]_i^o((id')P)$ specifies that id' is active on the continuation protocol P. Notice that the synchronisation action itself is preserved (we return to this point when addressing the language closure rules).

Rule Brd captures the case of broadcast interaction ($\star$), following lines similar to Bin. Apart from the absence of state update, the main difference is that now a set of potential reacting nodes is identified (I' denotes a set of node identifiers), out of which all those satisfying condition i are singled out (I). The latter are activated in the continuation protocol, to represent which we use $(\tilde{I})$ to abbreviate $(id_1) \ldots (id_m)$ considering $I = id_1, \ldots, id_m$. We remark that the set of actual reacting nodes may be empty (e.g., if none of the potential ones satisfies condition i), in which case $(\tilde{\emptyset})P$ is defined as P. The fact that the reduction step nevertheless takes place, even without actual reacting nodes, motivates the choice of the broadcast terminology, and differs from the binary interaction which is blocked while the targeted reacting node does not satisfy the condition.

Example 2.3. *Let us assume that node id_1 satisfies conditions o_1 and nodes id_2 and id_3 satisfy condition i_1 in Δ and both have node id_1 as power provider. In such case we may derive, using rule* Brd*, the reduction:*

$$\Delta; (id_1)[\mathsf{Locate}^\star]_{i_1}^{o_1} P_1 \to \Delta; [\mathsf{Locate}^\star]_{i_1}^{o_1}(id_2)(id_3)P_1$$

where nodes id_2 and id_3 react to the Locate *action enabled by id_1. Note that if id_2 and id_3 do not satisfy i_1 (or their power provider is not id_1) then the reduction step can nevertheless take place:*

$$\Delta; (id_1)[\mathsf{Locate}^\star]_{i_1}^{o_1} P_1 \to \Delta; [\mathsf{Locate}^\star]_{i_1}^{o_1} P_1$$

so it suffices for the enabling node to satisfy the output condition (o_1) for the broadcast to take place.

Rule Loc captures the case of local computation steps ($\bullet$). For the sake of uniformity we keep (both) output and input conditions that must be met by the state of the active node. Notice that the node that carries out the f step retains control, i.e., the same id is active before and after the synchronisation on f. Like for broadcast, we consider local steps do not involve any state update.

Rule IdSum states that an (id) operator active over a summation protocol can be considered for a reduction of (exclusively) one of the branches of the summation, provided that the id actually triggers the reduction, while preserving the summation. Notice that the final configuration in the premise is a summation (S_1') and hence cannot specify an active node (cf. Table 1), so necessarily the (id) enables a synchronisation

(via rules Bin, Brd or Loc). Notice also that this rule is needed because the (id) operator does not distribute over summation like it does for fork (cf. Table 2).

The previous semantic rules suggest that the structure of a protocol is preserved under reduction, while accounting to state changes and declaring the active node(s) in the next reduction step. However, as a protocol progresses different nodes might be active at different stages of that protocol and thus reduction is required to take place at any stage. To capture this intuition, rules for protocol language closure are introduced. Namely, a reduction may take place after a synchronisation action (rule Synch), within a summation (rule Sum), within a fork (rule Par), and after an id operator (rule Id). Thus, preserving the structure of the protocol allows for nodes to be active on (exactly) the same stage of the protocol simultaneously and/or at different moments in time.

Rule Struct closes reduction under structural congruence, so as to allow the reduction rules to focus on the case of an active node that immediately scopes over a synchronisation action summation.

Example 2.4. *Assuming that node id_2 satisfies conditions i_2 and o_2 in Δ, we may derive, relying on rules* Loc*, rule* IdSum*, rule* Struct*, and rule* Id*, the reduction:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Delta(id_2) \models o_2 \qquad \Delta(id_2) \models i_2}{\Delta;(id_2)[\mathsf{End}^\bullet]^{o_2}_{i_2} P_2 \to \Delta;[\mathsf{End}^\bullet]^{o_2}_{i_2}(id_2)P_2}\ \textsc{Loc}}{\Delta;(id_2)([\mathsf{End}^\bullet]^{o_2}_{i_2} P_2 + ([\mathsf{Locate}^\star]^{o_1}_{i_1} P_1) \to \Delta;[\mathsf{End}^\bullet]^{o_2}_{i_2}(id_2)P_2 + [\mathsf{Locate}^\star]^{o_1}_{i_1} P_1}\ (\textsc{IdSum})}{\Delta;(id_2)([\mathsf{Locate}^\star]^{o_1}_{i_1} P_1 + [\mathsf{End}^\bullet]^{o_2}_{i_2} P_2) \to \Delta;[\mathsf{Locate}^\star]^{o_1}_{i_1} P_1 + [\mathsf{End}^\bullet]^{o_2}_{i_2}(id_2)P_2}\ (\textsc{Struct})}{\Delta;(id_1)(id_2)([\mathsf{Locate}^\star]^{o_1}_{i_1} P_1 + [\mathsf{End}^\bullet]^{o_2}_{i_2} P_2) \to \Delta;(id_1)([\mathsf{Locate}^\star]^{o_1}_{i_1} P_1 + [\mathsf{End}^\bullet]^{o_2}_{i_2}(id_2)P_2)}\ (\textsc{Id})$$

where node id_2 carries out the End *local action. Note that node id_1 is still active on the summation protocol, and synchronisations on both branches of the summation are possible. Thus, both nodes id_1 and id_2 are active simultaneously on different stages of the protocol and a reduction may be triggered by id_1 or id_2.*

In order to make precise the notion that reduction may take place at any level of the protocol, we introduce active contexts that include all language constructs except for recursion. We may show that reduction is closed under such contexts.

Definition 2.5 (Active Contexts)**.** *We denote by $\mathscr{C}[\cdot]$ a global language term with one hole defined as follows*

$$\mathscr{C}[\cdot] \ ::= \ P \,|\, \mathscr{C}[\cdot] \quad | \quad (id)\mathscr{C}[\cdot] \quad | \quad [f^d]^o_i \mathscr{C}[\cdot] \quad | \quad S + \mathscr{C}[\cdot] \quad | \quad \cdot$$

Lemma 2.6 (Reduction Closed Under Active Contexts)**.** *If $\Delta; P \to \Delta'; Q$ then $\Delta; \mathscr{C}[P] \to \Delta'; \mathscr{C}[Q]$.*

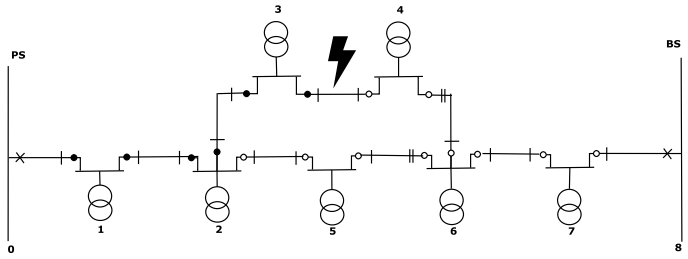
Proof. By induction on the structure of $\mathscr{C}[\cdot]$ following expected lines as for each possible active context there is a corresponding reduction rule. □

Since we are interested in developing protocols that may be used in different networks, we consider such development to be carried out using what we call static protocols, i.e., protocols that do not include the active node construct. Then, to represent a concrete operating system, active nodes may be added at "top-level" to the static specification (e.g., (id)Recovery where Recovery does not include any active nodes), together with the network state. Also, for the purpose of simplifying protocol design, we consider that action labels are unique (up to recursion unfolding). As usual, we exclude protocols where recursion appears unguarded by at least one synchronisation action (e.g., **rec** $X.X$).

Such notions allow us to streamline the distributed implementation presented in the next section. From now on, we only consider well-formed protocols that follow the above guidelines, i.e., originate from specifications where the active node construct only appears top-level, all action labels are distinct, and recursion is guarded.

Definition 2.7 (Static Protocols)**.** *We call static protocols the set of terms in the (id)-free fragment of the language given in Table 1.*

Figure 2: Power Distribution Grid



Definition 2.8 (Well-formed Protocol)**.** *We say protocol P is well-formed if there are P', I, Δ, and Δ' such that $\Delta;(\tilde{I})P' \rightarrow^{\star} \Delta';P$, where $\rightarrow^{\star}$ denotes the reflexive transitive closure of $\rightarrow$ and where P' is a static protocol that does not contain action label repetition and unguarded recursion.*

We finish this section by reporting on a result that informs on the global model, showing that protocols can be described as a parallel composition of (active node scoped) synchronisation action summations.

Proposition 2.9 (Protocol Normal Form)**.** *Let P be a protocol where recursion is guarded. We have that $P \equiv (\tilde{I}_1)S_1 \mid \ldots \mid (\tilde{I}_k)S_k$, where $k \geq 0$.*

Proof. By induction on the structure of P.

Case P is $[f^d]^o_i Q$: Immediate.

Case P is $(id)Q$: By the induction hypothesis $Q \equiv (\tilde{I}_1)S_1 \mid \ldots \mid (\tilde{I}_k)S_k$, and hence $(id)Q \equiv (id)(\tilde{I}_1)S_1 \mid \ldots \mid (id)(\tilde{I}_k)S_k$.

Case P is $\mathbf{0}$: Immediate.

Case P is X: Does not apply since recursion is guarded in P.

Case P is $\mathbf{rec}\ X.Q$: By the induction hypothesis we have that $Q \equiv (\tilde{I}_1)S_1 \mid \ldots \mid (\tilde{I}_k)S_k$, hence we have that $\mathbf{rec}\ X.Q \equiv (\tilde{I}_1)(S_1[\mathbf{rec}\ X.Q/X]) \mid \ldots \mid (\tilde{I}_k)(S_k[\mathbf{rec}\ X.Q/X])$.

Case P is $Q_1 \mid Q_2$: By the induction hypothesis $Q_1 \equiv (\tilde{I}_1)S_1 \mid \ldots \mid (\tilde{I}_k)S_k$ and $Q_2 \equiv (\tilde{I}'_1)S'_1 \mid \ldots \mid (\tilde{I}'_k)S'_k$, hence $Q_1 \mid Q_2 \equiv (\tilde{I}_1)S_1 \mid \ldots \mid (\tilde{I}_k)S_k \mid (\tilde{I}'_1)S'_1 \mid \ldots \mid (\tilde{I}'_k)S'_k$.

Case P is $S_1 + S_2$: Immediate.

□

3. Fault Management in Power Distribution Grids

In this section, we model a non trivial fault management protocol that handles error detection, localisation, and isolation in power grids. Furthermore, we model autonomous network reconfiguration and power restoration. We specify the behaviour of our protocol from a global point of view (i.e., using the global language) which is more natural and less error-prone with respect to considering the local viewpoints. The scenario we present here is inspired in related literature in a broad sense, but we nevertheless point to the coordination algorithms presented in [23] for a concrete reference of the sort of specifications we are targeting.

We consider a cross section of a power grid's network as reported in Fig. 2. The cross section shows a radial power grid's network with a primary power substation **PS**, a backup power substation **BS**, and seven secondary power substations, numbered from **1** to **7**. The type of this network is called radial because

every substation has only one incoming power input and possibly multiple power outputs. In some sense, the substation can be viewed as a power router.

Primary and backup substations have circuit breakers $\times$ that open up when a fault occurs in their domain, e.g., a transmission power line breaks. Each secondary substation has fault indicators (fault $\bullet$ and no fault $\circ$), line switches (closed | and open $\|$), and an embedded controller that implements the substation's behaviour and manages interactions with others. Fig. 2 illustrates a configuration where the secondary substations **1**-**5** are energised by the primary substation **PS**, while secondary substations **6** and **7** are energised by the backup substation **BS**. Secondary substations cannot operate the switches or exchange information without authorisation from the primary substation which supplies the power.

Let us consider that a fault occurs in the domain of the primary substation **PS**, e.g., the transmission power line between substations **3** and **4** breaks. The primary substation can sense the existence of a fault because its circuit breaker $\times$ opens up, but it cannot determine the location of the fault. Hence, the substation **PS** initiates the fault recovery protocol by synchronising with its directly connected secondary substations and delegates them to activate the local fault recovery protocol. The delegation between secondary substations propagates in the direction given by their fault indicators.

A secondary substation first validates the error signal by measuring its voltage level. If the voltage is zero, the secondary substation activates the fault recovery protocol, otherwise the signal is discarded. Once a secondary substation activates the fault recovery protocol, it inspects its own fault indicators and determines whether the fault is on its input or output power lines. If the fault is on its output, it delegates the substations connected to its output power lines to collaborate to locate the fault. If the fault is on its input power line, the substation (in our case, substation **4**) takes control and initiates both isolation and power restoration. The former consists of isolating the faulty line and restoring the power to the network's segment located before the fault, while the latter consists of restoring the power to the network's segment located after the fault.

For isolation, substation **4** opens its switches to the faulty line segment and collaborates with **3** to open its affected line switches as well. Notice that now the domain of the primary substation **PS** is segmented into two islands: $\{\mathbf{PS}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{5}\}$ and $\{\mathbf{4}\}$. The control is transferred back to the primary substation in a step by step fashion and the power is restored to the first island. For power restoration, substation **4** asks for power supply from one of its neighbours that is capable of supplying an additional substation (in our case, substation **6**). Once substation **6** supplies power to **4**, the latter changes its power source to **6** and now substation **4** belongs to the domain of the backup substation **BS**.

We now show how to provide a simple and intuitive global specification for our scenario using the linguistic primitives of the global language. In what follows, we use the following terminology: the state of a source link t can be 0 (to indicate a faulty link) or 1 otherwise. We will use z in place of the source *id* when a substation is not connected to a power supply. The initial state of each substation follows from Fig. 2. For instance, substations **3**, **4**, and **6** have the following initial states $\mathbf{3}[2, 1, \{2, 4\}, 1, 1, 1]$, $\mathbf{4}[3, 0, \{3, 6\}, 1, 0, 0]$, and $\mathbf{6}[7, 1, \{4, 5, 7\}, 2, 0, 0]$, respectively. The recovery protocol is reported below.

$$\begin{aligned} \textsc{Recovery} &\triangleq \mathbf{rec}\ X.([\mathsf{Locate}^{\star}]_{i_1}^{o_1} X + [\mathsf{End}^{\bullet}]_{i_2}^{o_2} \textsc{Islanding}) \\ \textsc{Islanding} &\triangleq \textsc{IsolationStart} \mid \textsc{Restoration} \end{aligned}$$

The protocol states that either Locate is broadcasted to the children of the enabling substation, after which the protocol starts over, or End is carried out and the node retains control. In case End is carried out, the enabling substation proceeds by activating the Islanding protocol. The latter specifies a fork between IsolationStart and the Restoration protocols.

A substation enabled on Recovery can broadcast Locate only when it has at least one faulty output, i.e., $o_1 = (e > 0)$. Furthermore, receiving substations can synchronise on Locate only if they have fault on their output or their input, i.e., $i_1 = (e > 0) \vee (t = 0)$. On the other hand, End can be carried out when the enabling substation has a fault on its input, i.e., $o_2 = (t = 0)$ and $i_2 = \mathsf{tt}$. Both actions have no side-effects on states.

The Isolation and the Restoration protocols are reported below:

$$\textsc{IsolationStart} \triangleq [\mathsf{Recover}^{\blacktriangle}]_{i_3}^{o_3}\mathbf{0} + [\mathsf{RecoverDone}^{\blacktriangle}]_{i_4}^{o_3}\textsc{Isolation}$$

$$\textsc{Isolation} \triangleq \mathbf{rec}\ X.([\mathsf{Isolate}^{\blacktriangle}]_{i_3}^{o_5}\mathbf{0} + [\mathsf{IsolateDone}^{\blacktriangle}]_{i_4}^{o_5}X + [\mathsf{Stop}^{\bullet}]_{i_2}^{o_6}\mathbf{0}))$$

$$\textsc{Restoration} \triangleq [\mathsf{Power}^{\blacktriangleright}]_{i_7}^{o_7}\mathbf{0}$$

The IsolationStart protocol states that a synchronisation between the enabling substation and its parent on the Recover or RecoverDone actions may happen only if the enabling substation has a fault on its input, i.e., $o_3 = (t = 0)$. The parent reacts on Recover if the number of faults on its output is greater than one, i.e., $i_3 = (e > 1)$. In this case the parent does not proceed, since there are still more faulty links to be handled. Instead, the RecoverDone synchronisation captures the case for the handling of the last faulty link, i.e., $i_4 = (e = 1)$, in which case the parent takes control and proceeds to Isolation. In both cases, the enabling station disconnects itself from the faulty line (setting parent to z) and the parent decrements its output faults (e), its active outputs (a), and also its capacity (k). Furthermore, both enabling and parent node remove each other from the list of neighbours, thus isolating the faulty link.

When the parent is enabled on Isolation then three synchronisation branches on the Isolate, the IsolateDone and the Stop actions can be carried out. An enabling substation can synchronise with its parent on an Isolate or IsolateDone actions if it is not the primary station, i.e., $o_5 = (i \neq \infty)$. As a side-effect on the state of the parent, the number of faults is decremented (e). Notice that the interpretation of the two branches on Isolate and IsolateDone is similar to the one on Recover and RecoverDone. Only the primary station ($o_6 = (i = \infty)$) can execute the Stop action, ending the Isolation protocol. As mentioned before, local actions have no side-effects.

Finally, the Restoration protocol states that a disconnected substation, i.e., $o_7 = (i = z)$, can synchronise with one of its neighbours ($\blacktriangleright$) on the Power action as a request for a power supply. Only a neighbour that has enough power and is fully functional ($i_7 = (k > a \wedge e = 0)$) can engage in the synchronisation. By doing so, the neighbour increments its active outputs (a) and the enabling substation marks its neighbour as its power source. The protocol then terminates after the network reconfiguration. Note that although the IsolationStart and the Restoration protocols are specified in parallel, a specific order is actually induced by the synchronisation conditions. Thanks to the output condition $o_7 = (i = z)$ of the Power action, we are sure that the power supply can be reestablished only after the faulty link is disconnected, which is a side-effect of either Recover or RecoverDone actions. Thus, the synchronisation on Power can only happen after the synchronisation on either Recover or RecoverDone.

The static protocol Recovery abstracts from the concrete network configuration. To represent a concrete grid network, active substations must be added at "top-level" to the Recovery protocol, together with the network state, i.e., Δ; (**PS**)Recovery where Δ is a mapping from a substation identifiers to states. Notice that the primary station, **PS**, is initially active because, according to our scenario, it is the only station that has rights to initiate protocols.

Example 3.1. *We show some of the reduction steps of the recovery protocol. For the sake of clarity, we label the reduction relation with the performed action and its direction and we index it with last applied rule. Also, since recursion in the recovery protocol is at the level of summation, every time a locate action is enabled the recovery protocol is unfolded inside a summation. We consider active contexts to simplify the presentation of*

Table 4: The Syntax of the Distributed Language

(Definition)	D	$::=$	$\langle c\rangle f^d?.R \mid R \mid D \,|\, D$
(Reaction)	R	$::=$	$C \mid \mathbf{0} \mid R \,|\, R$
(Choice)	C	$::=$	$\langle c\rangle f^d! \mid C + C$
(Network)	$\mathscr{N}$	$::=$	$s : D \mid \mathscr{N} \,\|\, \mathscr{N}$
(Action)	α	$::=$	$\langle c\rangle f^d? \mid \langle c\rangle f^d! \mid \langle c\rangle f$
(Label)	λ	$::=$	$\mathcal{L}^f \mid \tau$
(Link)	$\mathcal{L}$	$::=$	$id \to id \mid id \leftarrow id \mid id!\star \mid id?\star$

the unfoldings, namely $\mathscr{C}[\cdot] = [\mathsf{Locate}^\star]^{o_1}_{i_1} \cdot + [\mathsf{End}^\bullet]^{o_2}_{i_2}\textsc{Islanding}$. *We may then write:*

$$
\begin{array}{rl}
 & \Delta; (\mathbf{PS})\textsc{Recovery} \\
\xrightarrow{\mathsf{Locate}^\star}_{\textsc{Id}} & \Delta; \mathscr{C}[(\mathbf{1})\textsc{Recovery}] \\
\xrightarrow{\mathsf{Locate}^\star}_{\textsc{Sum}} & \Delta; \mathscr{C}[\mathscr{C}[(\mathbf{2})\textsc{Recovery}]] \\
\xrightarrow{\mathsf{Locate}^\star}_{\textsc{Sum}} & \Delta; \mathscr{C}[\mathscr{C}[\mathscr{C}[(\mathbf{3})\textsc{Recovery}]]] \\
\xrightarrow{\mathsf{Locate}^\star}_{\textsc{Sum}} & \Delta; \mathscr{C}[\mathscr{C}[\mathscr{C}[\mathscr{C}[(\mathbf{4})\textsc{Recovery}]]]] \\
\xrightarrow{\mathsf{End}^\bullet}_{\textsc{Sum}} & \Delta'; \mathscr{C}[\mathscr{C}[\mathscr{C}[\mathscr{C}[[\mathsf{Locate}^\star]^{o_1}_{i_1}\textsc{Recovery} + [\mathsf{End}^\bullet]^{o_2}_{i_2}(\mathbf{4})\textsc{Islanding}]]]] \\
 & \equiv (\mathscr{C}'[\cdot] = \mathscr{C}[\mathscr{C}[\mathscr{C}[\mathscr{C}[[\mathsf{Locate}^\star]^{o_1}_{i_1}\textsc{Recovery} + [\mathsf{End}^\bullet]^{o_2}_{i_2}\cdot]]]]) \\
 & \Delta'; \mathscr{C}'[(\mathbf{4})\textsc{IsolationStart} \,|\, (\mathbf{4})\textsc{Restoration}] \\
\xrightarrow{\mathsf{RecoverDone}^\blacktriangle}_{\textsc{Sum}} & \Delta''; \mathscr{C}'[[\mathsf{Recover}^\blacktriangle]^{o_3}_{i_3}\mathbf{0} + [\mathsf{RecoverDone}^\blacktriangle]^{o_3}_{i_4}((\mathbf{3})\textsc{Isolation}) \,|\, (\mathbf{4})\textsc{Restoration}] \\
\vdots &
\end{array}
$$

Initially, the primary station **PS** *broadcasts* Locate *to its children, in our scenario it is substation* **1**. *Thus substation* **1** *is active in the next step which is actually the recursive protocol itself. Note that reductions involve expanding the protocol term as carried out actions are preserved (e.g., each time the* Locate *action is carried out a copy of context* $\mathscr{C}[\cdot]$ *is required). Note also that in the third step where substation* **2** *broadcasts* Locate *to its children* **3** *and* **5**, *only* **3** *is activated in the next step because substation* **3** *satisfies the input condition of* Locate *while* **5** *does not. Once the* Locate *signal reaches substation* **4**, *the control is retained by substation* **4** *by executing* End *and now substation* **4** *is active simultaneously on* IsolationStart *and* Restoration *protocols. Substation* **4** *executes* RecoverDone *and in the next step substation* **3** *is active on* Isolation *while substation* **4** *is still active on* Restoration.

We remark that the protocol is able to handle configurations with multiple faults, where several nodes may be active simultaneously, e.g., Locate albeit belonging to different parts (subtrees) of the network.

4. A Distributed Model for Operation Control

In this section, we present the target model that will be used to carry out the global protocol descriptions in a distributed way. We consider a network of nodes where each node has a state s (previously introduced) and a behaviour given in terms of *definitions*, *reactions* and *choices*. Intuitively, definitions allow nodes to synchronise on actions, after which proceeding as specified in reactions. The latter are defined as alternative behaviours specified in choices. The syntax of behaviours and networks is shown in Table 4, reusing syntactic elements given in the global language (namely action labels f, directions d, and conditions c).

Table 5: Semantics of Definitions

$$\langle c\rangle f^d?.R \xrightarrow{\langle c\rangle f^d?} R \mid \langle c\rangle f^d?.R \quad \textsc{Inp} \qquad \langle c\rangle f^d! \xrightarrow{\langle c\rangle f^d!} \mathbf{0} \quad \textsc{Out}$$

$$\frac{C_1 \xrightarrow{\alpha} C_1'}{C_1 + C_2 \xrightarrow{\alpha} C_1'} \textsc{Sum} \qquad \frac{D_1 \xrightarrow{\alpha} D_1'}{D_1 \mid D_2 \xrightarrow{\alpha} D_1' \mid D_2} \textsc{Int} \qquad \frac{D_1 \xrightarrow{\langle c_1\rangle f^\bullet!} D_1' \quad D_2 \xrightarrow{\langle c_2\rangle f^\bullet?} D_2'}{D_1 \mid D_2 \xrightarrow{\langle c_1 \wedge c_2\rangle f} D_1' \mid D_2'} \textsc{Self}$$

A definition D may either be a pair of simultaneously active (sub)definitions $D_1 \mid D_2$, a reaction R, or the (persistent) input $\langle c\rangle f^d?.R$. The latter allows a node to react to a synchronisation on f (according to direction d), provided that the node satisfies condition c, leading to the activation of reaction R. A reaction R can either be a pair of simultaneously active (sub)reactions $R_1 \mid R_2$ (that can be specified as continuation of an input), the inaction $\mathbf{0}$ or a choice C. The latter is either a pair of (sub)choices $C_1 + C_2$ or the output $\langle c\rangle f^d!$ that allows a node to enable a synchronisation on f (targeting the nodes specified by direction d), provided that the node satisfies condition c.

A network $\mathscr{N}$ is either a pair of (sub)networks $\mathscr{N}_1 \parallel \mathscr{N}_2$ or a node $s : D$ which comprises a behaviour given by a definition (D) and a state (s). We recall that a state is a register of the form $id[id', t, n, k, a, e]$.

The operational semantics of networks is defined by a labelled transition system (LTS), which relies on the operational semantics of definitions, also defined by an LTS. We denote by $D_1 \xrightarrow{\alpha} D_2$ that definition D_1 exhibits action α and evolves to D_2. The actions ranged over by α are $\langle c\rangle f^d?$, $\langle c\rangle f^d!$, and $\langle c\rangle f$. Action $\langle c\rangle f^d?$ represents the ability to react to a synchronisation on f, provided that the node satisfies condition c, and according to direction d. Similarly, $\langle c\rangle f^d!$ represents the ability to enable a synchronisation on f, also considering the condition c and the targeting direction d. A local computation step is captured by $\langle c\rangle f$, which also specifies the label of the action f and a condition c. The rules that define the LTS of definitions, briefly explained next, are shown in Table 5.

Rule Inp states that an input can exhibit the corresponding reactive transition, comprising synchronisation action label f, condition c, and direction d. The input results in the activation of the respective reaction R, while the input itself is preserved, which means that all inputs are persistently available. Likewise, rule Out states that an output can exhibit the corresponding synchronisation enabling transition, after which it terminates. Rules Sum and Int are standard, specifying alternative and interleaving behaviour, respectively. Rule Self captures local computation steps, which are actually the result of a synchronisation between an output (reaction) and an input (definition) which specify direction $\bullet$. We remark that both conditions are registered in the action label of the conclusion (by means of a conjunction), so the node must satisfy them in order for the computation step to be carried out. For the sake of brevity, we omit the symmetrical rules of Sum, Int, and Self.

The operational semantics that defines the LTS of networks is shown in Table 6, where we denote by $\mathscr{N}_1 \xrightarrow{\lambda} \mathscr{N}_2$ that network $\mathscr{N}_1$ exhibits label λ and evolves to network $\mathscr{N}_2$. The transition labels ranged over by λ are $\mathcal{L}^f$ and τ, capturing network interactions and local computation steps. The label $\mathcal{L}^f$ comprises the action label f and the communication link $\mathcal{L}$ which represents either binary ($id \rightarrow id$ and $id \leftarrow id$) or broadcast ($id!\star$ and $id?\star$) interaction.

The binary link $id_1 \rightarrow id_2$ specifies that node id_1 is willing to enable a synchronisation with node id_2 while $id_1 \leftarrow id_2$ specifies that node id_1 is willing to react to a synchronisation from node id_2. Furthermore, the broadcast link $id!\star$ specifies that node id is willing to enable a synchronisation with all of its direct children, while $id?\star$ specifies that a node is willing to react to a broadcast from node id. We use $id(s)$ and $i(s)$ to denote the identities of the node itself and of the parent (i.e., if $s = id_1[id_2, t, n, k, a, e]$ then $id(s) = id_1$ and $i(s) = id_2$).

We consider short-range broadcast in the sense that only direct children of the node id can be the target of the synchronisation on $id!\star^f$ (i.e., only nodes that satisfy condition $i(s) = id$). Moreover, a node can accept a synchronisation from its parent only: (1) if its local definitions are able to react to the synchronisation action (i.e., $D \xrightarrow{\langle c\rangle f^\star?} D'$ is defined); also, (2) if the current state of the node satisfies the condition of the

Table 6: Semantics of Networks

$$\frac{D \xrightarrow{\langle c\rangle f} D' \quad s \models c}{s : D \xrightarrow{\tau} s : D'}\ \textsc{Loc} \qquad \frac{D \xrightarrow{\langle c\rangle f^{\blacktriangle}!} D' \quad s \models c \quad s' = f^{\blacktriangle}!(s, i(s))}{s : D \xrightarrow{id(s)\to i(s)^f} s' : D'}\ \textsc{oBinU}$$

$$\frac{D \xrightarrow{\langle c\rangle f^{\blacktriangleright}!} D' \quad s \models c \quad s' = f^{\blacktriangleright}!(s, id) \quad id \in n(s)}{s : D \xrightarrow{id(s)\to id^f} s' : D'}\ \textsc{oBinR}$$

$$\frac{d \in \{\blacktriangle, \blacktriangleright\} \quad D \xrightarrow{\langle c\rangle f^{d}?} D' \quad s \models c \quad s' = f^{d}?(s, id)}{s : D \xrightarrow{id(s)\leftarrow id^f} s' : D'}\ \textsc{iBin}$$

$$\frac{D \xrightarrow{\langle c\rangle f^{\star}!} D' \; s \models c}{s : D \xrightarrow{id(s)!\star^f} s : D'}\ \textsc{oBrd} \qquad \frac{D \xrightarrow{\langle c\rangle f^{\star}?} D' \; s \models c}{s : D \xrightarrow{i(s)?\star^f} s : D'}\ \textsc{iBrd} \qquad \frac{\mathit{discard}(s : D, id?\star^f)}{s : D \xrightarrow{id?\star^f} s : D}\ \textsc{dBrd}$$

$$\frac{\mathscr{N}_1 \xrightarrow{\lambda_1} \mathscr{N}_1' \quad \mathscr{N}_2 \xrightarrow{\lambda_2} \mathscr{N}_2'}{\mathscr{N}_1 \parallel \mathscr{N}_2 \xrightarrow{\gamma(\lambda_1,\lambda_2)} \mathscr{N}_1' \parallel \mathscr{N}_2'}\ \textsc{Com} \qquad \frac{\mathscr{N}_1 \xrightarrow{\lambda} \mathscr{N}_1' \quad \lambda \notin \{id!\star^f,\ id?\star^f\}}{\mathscr{N}_1 \parallel \mathscr{N}_2 \xrightarrow{\lambda} \mathscr{N}_1' \parallel \mathscr{N}_2}\ \textsc{Par}$$

defined reaction (i.e., $D \xrightarrow{\langle c\rangle f^{\star}?} D'$ and $s \models c$). We define a predicate $\mathit{discard}(s : D, id?\star^f)$ that takes as parameters a node $s : D$ and a network label $id?\star^f$. This predicate is used to identify the case when nodes may discard broadcasts, i.e., when any of the above conditions is not satisfied.

We briefly describe the rules given in Table 6, which address individual nodes and networks. The former, roughly, lift the LTS of definitions to the level of nodes taking into account conditions and side-effects of synchronisation actions.

Rule Loc states that a node can evolve silently with a τ transition when its definitions exhibit a local computation step $\langle c\rangle f$, provided that the state of the node satisfies the condition of the computation step $s \models c$. Rule oBinU is used to synchronise with the parent node, rule oBinR is used to synchronise with a neighbour and rule iBin is used to react to a synchronisation from either a child or a neighbour node. More precisely, rules oBinU/oBinR and iBinU express that nodes can respectively exhibit enabling/reactive transitions provided that local definitions in D exhibit the corresponding transitions, with synchronisation action label f, direction $\blacktriangle$ or $\blacktriangleright$ and condition c. Also, the condition c is checked against the state s, and the latter is updated according to the side-effects of the synchronisation (for both enabling and reacting nodes, as described previously for the global language semantics). We remark that the rules for binary interaction register in the communication link the identities of the interacting parties.

Rule oBrd states that a node can enable a broadcast synchronisation on action f if local definitions in D can exhibit the corresponding enabling transition $D \xrightarrow{\langle c\rangle f^{\star}!} D'$, and condition c is satisfied by the local state $s \models c$. Similarly, rule iBrd states that a node can react to a broadcast synchronisation from its parent only when local definitions in D can exhibit the corresponding transition $D \xrightarrow{\langle c\rangle f^{\star}?} D'$ and the condition is satisfied by the local state $s \models c$. Otherwise, rule dBrd may be applied, capturing the case when the node discards the synchronisation and remains unaltered.

Rule Par (and its omitted symmetric) governs the interleaving of networks when binary and local actions are observed, i.e, $\lambda \notin \{id!\star^f,\ id?\star^f\}$. Rule Com governs the synchronisation of networks when either binary or broadcast actions are observed. The function $\gamma(\lambda_1, \lambda_2)$ identifies the resulting label, and is defined as follows.

Table 7: Structural Congruence - Definitions and Networks

$$D \mid \mathbf{0} \equiv D \qquad D_1 \mid (D_2 \mid D_3) \equiv (D_1 \mid D_2) \mid D_3 \qquad D_1 \mid D_2 \equiv D_2 \mid D_1$$

$$\langle c \rangle f^d?.R \mid \langle c \rangle f^d?.R \equiv \langle c \rangle f^d?.R \qquad D_1 + (D_2 + D_3) \equiv (D_1 + D_2) + D_3 \qquad D_1 + D_2 \equiv D_2 + D_1$$

$$D_1 \equiv D_2 \Rightarrow s : D_1 \equiv s : D_2 \qquad \mathscr{N}_1 \parallel (\mathscr{N}_2 \parallel \mathscr{N}_3) \equiv (\mathscr{N}_1 \parallel \mathscr{N}_2) \parallel \mathscr{N}_3 \qquad \mathscr{N}_1 \parallel \mathscr{N}_2 \equiv \mathscr{N}_2 \parallel \mathscr{N}_1$$

$$\gamma(\lambda_1, \lambda_2) = \begin{cases} \tau & \text{if } \lambda_1 = id_1 \to {id_2}^f \ \&\ \lambda_2 = id_2 \leftarrow {id_1}^f \\ \tau & \text{if } \lambda_1 = id_1 \leftarrow {id_2}^f \ \&\ \lambda_2 = id_2 \to {id_1}^f \\ \lambda_1 & \text{if } \lambda_1 = id!\star^f \ \&\ \lambda_2 = id?\star^f \\ \lambda_2 & \text{if } \lambda_1 = id?\star^f \ \&\ \lambda_2 = id!\star^f \\ \lambda & \text{if } \lambda_1 = \lambda_2 = \lambda = id?\star^f \\ \bot & \text{otherwise} \end{cases}$$

The function returns τ when synchronisation on a binary action is possible and the enabling label when a synchronisation on a broadcast action is possible, as reported in the first four cases respectively. The fifth case states that both networks can react simultaneously to the same broadcast, otherwise the function is undefined as reported in the last case. We remark that the semantics of broadcasts is presented in a standard way (cf. [24, 25]).

For the purpose of the operational correspondence result (Theorem 5.10), we consider structural congruence of networks and of behaviours defined by the rules shown in Table 7. The rules capture expected principles (namely, that operators $\parallel$, $\mid$, and $+$ are associative and commutative, and that $\mid$ has identity element $\mathbf{0}$) and an absorbing principle for persistent inputs (i.e., $\langle i \rangle f^d?.R \mid \langle i \rangle f^d?.R \equiv \langle i \rangle f^d?.R$), which allows to reason about persistent inputs as if they are unique. We may show that structurally equivalent networks have equivalent behaviours as follows, where we denote by $D_1 \xrightarrow{\lambda}_{\equiv} D_2$ (resp. $\mathscr{N}_1 \xrightarrow{\lambda}_{\equiv} \mathscr{N}_2$) that there exists D' (resp. $\mathscr{N}'$) such that $D_1 \xrightarrow{\lambda} D'$ and $D' \equiv D_2$ (resp. $\mathscr{N}_1 \xrightarrow{\lambda} \mathscr{N}'$ and $\mathscr{N}' \equiv \mathscr{N}_2$).

Lemma 4.1 (Definition and Network LTS Closure Under Structural Congruence)**.** *We have that:*

1. *If* $D_1 \xrightarrow{\alpha} D_1'$ *and* $D_1 \equiv D_2$ *then* $D_2 \xrightarrow{\alpha}_{\equiv} D_1'$.
2. *If* $\mathscr{N}_1 \xrightarrow{\lambda} \mathscr{N}_1'$ *and* $\mathscr{N}_1 \equiv \mathscr{N}_2$ *then* $\mathscr{N}_2 \xrightarrow{\lambda}_{\equiv} \mathscr{N}_1'$.

Proof. The proof follows by induction on the shape of the derivation of $D_1 \equiv D_2$ and $\mathscr{N}_1 \equiv \mathscr{N}_2$ in expected lines, where *2* relies on *1*. □

We finish this section by informing on the general structure of definitions as stated in the following result, namely that any choice can be described as a summation of outputs, that reactions are a parallel composition of choices, and that definitions are a parallel composition of input prefixed reactions in parallel with a reaction. In the following we use $\Pi_{i \in I} D_i$ to abbreviate $D_1 \mid \ldots \mid D_k$, and $\Sigma_{i \in I} D_i$ to abbreviate $D_1 + \ldots + D_k$, when $I = 1, \ldots, k$ (if $I = \emptyset$ then $\Pi_{i \in I} D_i$ denotes $\mathbf{0}$).

Proposition 4.2 (Definition Normal Form)**.** *For any choice* C*, reaction* R*, and definition* D *we have that*

$$C \equiv \Sigma_{l \in L} \langle c_l \rangle f_l^{d_l}! \qquad R \equiv \Pi_{j \in J} C_j \qquad D \equiv (\Pi_{i \in I} \langle c_i \rangle f_i^{d_i}?.R_i) \mid R'$$

Proof. By induction of the structure of C, R, and D, following expected lines. In case C is $\langle c \rangle f^d!$ the result is direct, and in case C is $C_1 + C_2$ the result follows by gathering the summations obtained by induction hypothesis. In case R is C or $\mathbf{0}$ the result is direct, and in case R is $R_1 \mid R_2$ the result follows by gathering the products obtained by induction hypothesis. In case D is $\langle c \rangle f^d?.R$ or R the result is direct (since $D \mid \mathbf{0} \equiv D$), and in case D is $D_1 \mid D_2$ the result follows by gathering the products and reactions obtained by induction hypothesis. □

Table 8: Projection

$[\![[f^d]_i^o P]\!]_\sigma^?$	$\triangleq$	$\langle i \rangle f^d?.[\![P]\!]_\sigma^! \mid [\![P]\!]_\sigma^?$		iSynch
$[\![[f^d]_i^o P]\!]_\sigma^!$	$\triangleq$	$\langle o \rangle f^d!$		oSynch
$[\![[f^d]_i^o P]\!]_\sigma^{id}$	$\triangleq$	$[\![P]\!]_\sigma^{id}$		idSynch
$[\![(id)P]\!]_\sigma^{id}$	$\triangleq$	$[\![P]\!]_\sigma^{id} \mid [\![P]\!]_\sigma^!$		idNode
$[\![(id)P]\!]_\sigma^{\mathtt{r}}$	$\triangleq$	$[\![P]\!]_\sigma^{\mathtt{r}}$	$(\mathtt{r} \neq id)$	pNode
$[\![\mathbf{0}]\!]_\sigma^{\mathtt{r}}$	$\triangleq$	$\mathbf{0}$		pNil
$[\![X]\!]_\sigma^{\mathtt{r}}$	$\triangleq$	$\mathbf{0}$	$(\mathtt{r} \neq !)$	pVar
$[\![X]\!]_\sigma^!$	$\triangleq$	$[\![P]\!]_\sigma^!$	$(\sigma(X) = P)$	oVar
$[\![\mathbf{rec}\ X.P]\!]_\sigma^{\mathtt{r}}$	$\triangleq$	$[\![P]\!]_{\sigma[X \mapsto P]}^{\mathtt{r}}$		pRec
$[\![P \mid Q]\!]_\sigma^{\mathtt{r}}$	$\triangleq$	$[\![P]\!]_\sigma^{\mathtt{r}} \mid [\![Q]\!]_\sigma^{\mathtt{r}}$		pPar
$[\![S_1 + S_2]\!]_\sigma^!$	$\triangleq$	$[\![S_1]\!]_\sigma^! + [\![S_2]\!]_\sigma^!$		oSum
$[\![S_1 + S_2]\!]_\sigma^{\mathtt{r}}$	$\triangleq$	$[\![S_1]\!]_\sigma^{\mathtt{r}} \mid [\![S_2]\!]_\sigma^{\mathtt{r}}$	$(\mathtt{r} \neq !)$	pSum
$[\![\Delta; P]\!]$	$\triangleq$	$\Pi_{\forall id \in dom(\Delta)}(\Delta(id) : [\![P]\!]_\emptyset^? \mid [\![P]\!]_\emptyset^{id})$		Proj

5. Local Controller Synthesis

In this section, we present the automatic translation of the global specifications into the target distributed model. Namely, we synthesise the controllers that operate locally in each node from the global specification, and ensure the correctness of the translation by means of an operational correspondence result. The development presented here can therefore be seen as a proof of concept that global descriptions may be automatically compiled to provably correct distributed implementations. In Sect. 5.1 we present the translation which is given in terms of a projection function, characterised by properties reported in Sect. 5.2. Then, in Sect. 5.3, we state the operational correspondence result (Theorem 5.10).

5.1. Automatic Translation

We now present the projection function. Intuitively, for each synchronisation action specified in the source protocol, there will be a corresponding reactive definition in every node. This way, we ensure that all nodes are equipped to react to any synchronisation. Also, in every active node in the global specification there will be a corresponding enabling behaviour. Since the global model prescribes the transference of the active role in synchronisations, the distributed implementation will also specify that reactions of definitions lead to the activation of enabling behaviour.

The projection function, defined by the cases reported in Table 8, realises the above principles, in particular in the three types of projection (ranged over by $\mathtt{r}$). Reactive projection, denoted by ?, is used to generate the persistent inputs that capture reactive behaviour; active projection, denoted by id (for some id), is used to identify and generate the enabling behaviours of node id; and enabling projection, denoted by !, is used to generate the outputs that capture enabling behaviour. We then denote by $[\![P]\!]_\sigma^{\mathtt{r}}$ the projection of the protocol P according to parameters $\mathtt{r}$ and σ. The latter is a mapping from recursion variables to protocols, used to memorise recursive protocols and abstract from their unfolding.

We briefly explain the cases in Table 8. Case iSynch shows the reactive projection of the synchronisation action, yielding a persistent input with the respective condition i, label f, and direction d. The continuation of the input is obtained by the enabling projection (!) of the continuation of the synchronisation action (P). Hence, the reaction leads to the enabling of the continuation which captures the transference of the active role in synchronisations. The result of the projection also specifies the (simultaneously active $\mid$) reactive projection of the continuation protocol, so as to generate the corresponding persistent inputs. Case oSynch shows the enabling projection of the synchronisation action, yielding the output considering the respective

condition o, label f, and direction d. Notice that the fact that outputs do not specify continuations is aligned with the idea that a node yields the active role after enabling a synchronisation.

Case IDSYNCH shows the active projection of the synchronisation action, yielding the active projection of the continuation. The idea is that active projection inspects the structure of the protocol, and introduces enabling behaviour whenever an active node construct specifying the respective id is found (i.e., $(id)P$). This is made precise in case IDNODE, where the projection yields both the enabling projection of the protocol P together with its the active projection, so as to address configurations in which the same node is active in different stages of the protocol. Case PNODE instead shows that the other types of projection of the active node construct result in the respective projection of the continuation.

Case PNIL says that the terminated protocol is projected (in all types of projection) to inaction ($\mathbf{0}$). Case PVAR says that the active/reactive projections of the recursion variable also yield inaction, hence do not require reasoning on the unfolding. Instead, case OVAR shows the enabling projection of the recursion variable, yielding the enabling projection of the variable mapping, which allows to account for the unfolding. This is made precise in case PREC where the mapping is updated with the association of the variable maps to the recursion body.

We remark that well-formed protocols do not specify active node constructs in the body of a recursion, since they originate from protocols where all active node constructs are top-level, hence the active projection of any recursive protocol necessarily yields inaction (nevertheless captured in the general case).

Case PPAR says that (all types of) projection of the fork protocol yields the (simultaneously active) respective projections of the branches of the fork. Cases OSUM and PSUM address the summation protocol: on the one hand, the enabling projection yields the choice between the projections of the branches; on the other hand, reactive/active projection yield the simultaneously active projections of the branches. Notice that choices may only specify outputs, hence only enabling projection may yield alternative (output) behaviour. Reactive projection yields a collection of persistent inputs (one per synchronisation action) which are simultaneously active. Active projection generates the enabling behaviour of active nodes, so if such active nodes are found in (continuations of) the branches of the summation, their enabling behaviour is taken as simultaneously active.

The projection of a configuration, denoted $[\![\Delta; P]\!]$ and defined in the PROJ case, specifies a parallel composition of all nodes of the network (i.e., all those comprised in the network state). Each node is obtained by considering the state yielded by the respective network state mapping, and considering the behaviour is yielded by a combination of the reactive and the active projections of the protocol P. Notice that the active projection is carried out considering the node identifier, hence the result potentially differs between distinct nodes, while the reactive projection is exactly the same for all nodes. Intuitively, consider the reactive projection as the static collection of reactive definitions, and the active projection as the runtime (immediately available) enabling behaviour.

Example 5.1. *The* SIMPLE *protocol previously described in Sect. 1.1, reported below, is a simplified version of the* RECOVERY *protocol in Sect. 3. The primary substation is initially active on* SIMPLE.

$$(\mathbf{PS})\text{SIMPLE} \triangleq (\mathbf{PS})(\mathbf{rec}\ \mathbf{X}.([\mathsf{Locate}^\star]^{\mathbf{o_1}}_{\mathbf{i_1}}\mathbf{X} + [\mathsf{End}^\bullet]^{\mathbf{o_2}}_{\mathbf{i_2}}\mathbf{0}))$$

We can use Table 8 to generate the distributed implementation of the network in Fig. 2. The network state Δ is defined in Section 3 and follows directly form Fig. 2. Every node has the following projection $s_{id} : [\![(\mathbf{PS})\text{SIMPLE}]\!]^?_\emptyset \mid [\![(\mathbf{PS})\text{SIMPLE}]\!]^{id}_\emptyset$. By Table 8, we have that when $id \neq \mathbf{PS}$ the projection is $s_{id} : [\![\text{SIMPLE}]\!]^?_\emptyset$; otherwise the projection is $s_{\mathbf{PS}} : [\![\text{SIMPLE}]\!]^?_\emptyset \mid [\![(\mathbf{PS})\text{SIMPLE}]\!]^{\mathbf{PS}}_\emptyset$. Thus the id-projection is equivalent to $\mathbf{0}$ for all nodes that are not currently active on the protocol and the ?-projection is the same for all nodes. The generated code is as follows:

$$\begin{aligned}
[\![\text{SIMPLE}]\!]^?_\emptyset &\triangleq \langle i_1\rangle\mathsf{Locate}^\star?.\,(\langle o_1\rangle\mathsf{Locate}^\star! + \langle o_2\rangle\mathsf{End}^\bullet!) \mid \langle i_2\rangle\mathsf{End}^\bullet?.0\\
[\![(\mathbf{PS})\text{SIMPLE}]\!]^{\mathbf{PS}}_\emptyset &\triangleq \langle o_1\rangle\mathsf{Locate}^\star! + \langle o_2\rangle\mathsf{End}^\bullet!
\end{aligned}$$

Clearly, only the primary station is initially active and can enable synchronisation on either Locate *or* End*, provided that its state $s_{\mathbf{PS}}$ satisfies o_1 or o_2 respectively. Recalling that o_1 regards faulty output links*

and o_2 regards faulty input link, we have that $\mathbf{PS}$ *satisfies o_1 but not o_2. Hence the conditions can resolve the nondeterministic choice and thus* Locate *is broadcasted. Since substation* $\mathbf{1}$ *is the only child of* $\mathbf{PS}$*, it can react by exhibiting the persistent input* $\mathbf{PS}?\star^{\mathsf{Locate}}$*, considering that $s_1 \models i_1$, leading to the activation of the respective output summation. This evolution can be captured as follows*

$$
\begin{array}{l}
s_{\mathbf{PS}} : \langle i_1\rangle \mathsf{Locate}^\star?.\,(\langle o_1\rangle \mathsf{Locate}^\star! + \langle o_2\rangle \mathsf{End}^\bullet!) \mid \langle i_2\rangle \mathsf{End} \mid \langle o_1\rangle \mathsf{Locate}^\star! + \langle o_2\rangle \mathsf{End}^\bullet! \\
\| \\
s_{\mathbf{1}} : \langle i_1\rangle \mathsf{Locate}^\star?.\,(\langle o_1\rangle \mathsf{Locate}^\star! + \langle o_2\rangle \mathsf{End}^\bullet!) \mid \langle i_2\rangle \mathsf{End} \\
\| \\
\ldots \\
\qquad\qquad \xrightarrow{\mathbf{PS}!\star^{\mathsf{Locate}}} \\
s_{\mathbf{PS}} : \langle i_1\rangle \mathsf{Locate}^\star?.\,(\langle o_1\rangle \mathsf{Locate}^\star! + \langle o_2\rangle \mathsf{End}^\bullet!) \mid \langle i_2\rangle \mathsf{End} \\
\| \\
s_{\mathbf{1}} : \langle i_1\rangle \mathsf{Locate}^\star?.\,(\langle o_1\rangle \mathsf{Locate}^\star! + \langle o_2\rangle \mathsf{End}^\bullet!) \mid \langle i_2\rangle \mathsf{End} \mid \langle o_1\rangle \mathsf{Locate}^\star! + \langle o_2\rangle \mathsf{End}^\bullet! \\
\| \\
\ldots
\end{array}
$$

where we abstract away from the rest of the network not involved in the interaction. Notice that the transition label is $\mathbf{PS}!\star^{\mathsf{Locate}}$ *so as to ensure that all children of* $\mathbf{PS}$ *can receive the message (while other nodes simply discard it). Notice also that there is a 1-to-1 correspondence between the evolution in the global and the distributed models in the sense that considering the appropriate Δ we have that*

$$\Delta; (\mathbf{PS})\textsc{Simple} \to \Delta; [\mathsf{Locate}^\star]^{o_1}_{i_1}((\mathbf{1})\textsc{Simple}) + [\mathsf{End}^\bullet]^{\mathbf{o_2}}_{\mathbf{i_2}}\mathbf{0}$$

and the projection of the final configuration matches the final configuration of the network above.

5.2. Characterisation of the Projection Function

We now present fundamental properties that provide a general characterisation of the projection function. We start by characterising the processes that result from the different types of projection. Namely, we show that !-projection yields a reaction, corresponding to the behaviours that are to be activated upon a synchronisation. To prove this property, we require an auxiliary one ensuring that the !-projection of recursion guarded processes yields a reaction regardless of the mapping considered in the projection. Then, we show that *id*-projection yields a reaction, corresponding to the enabling behaviours associated to active nodes. Finally, we show that ?-projection yields a parallel composition of input prefixed reactions, corresponding to the collection of synchronisation actions present in the global protocol.

Lemma 5.2 (Projection Normal Form)**.** *Let P be any protocol and σ be any mapping such that the following holds: for all $X \in fv(P)$ and for any σ'' it is the case that $[\![\sigma(X)]\!]^!_{\sigma''} \equiv R'''$. We have that*

1. *If recursion is guarded in P then $[\![P]\!]^!_{\sigma'} \equiv R$ for any σ'.*
2. $[\![P]\!]^!_\sigma \equiv R'$.
3. $[\![P]\!]^?_\sigma \equiv \Pi_{i\in I}\langle c_i\rangle f_i^{d_i}?.R_i$.
4. $[\![P]\!]^{id}_\sigma \equiv R''$.

Proof. By induction on the structure of P, where each item relies on previous ones (except for *3.* and *4.* that are independent between them).

Case P is $[f^d]^o_i Q$:

1. Direct since $[\![[f^d]^o_i Q]\!]^!_{\sigma'}$ is defined as $\langle o\rangle f^d!$.
2. Likewise.

3. We have that $[\![[f^d]_i^o Q]\!]_\sigma^?$ by definition is $\langle i \rangle f^d?.[\![Q]\!]_\sigma^! \mid [\![Q]\!]_\sigma^?$. By induction hypothesis we have that $[\![Q]\!]_\sigma^? \equiv \Pi_{i \in I} \langle c_i \rangle f_i^{d_i}?.R_i$ and by *2.* we have that $[\![Q]\!]_\sigma^! \equiv R'$, hence we conclude $[\![[f^d]_i^o Q]\!]_\sigma^? \equiv \langle i \rangle f^d?.R' \mid \Pi_{i \in I} \langle c_i \rangle f_i^{d_i}?.R_i$.
4. We have that $[\![[f^d]_i^o Q]\!]_\sigma^{id}$ by definition is $[\![Q]\!]_\sigma^{id}$ hence the result follows directly from the induction hypothesis.

Case P is $(id')Q$:

1. We have that $[\![(id')Q]\!]_{\sigma'}^!$ by definition is $[\![Q]\!]_{\sigma'}^!$ hence the result follows directly from the induction hypothesis.

2-3. Likewise.

4. We have that $[\![(id')Q]\!]_\sigma^{id}$ by definition is $[\![Q]\!]_\sigma^{id}$ if $id \neq id'$ in which case the result follows directly from the induction hypothesis. In case $id = id'$ we have that $[\![(id')Q]\!]_\sigma^{id}$ is defined as $[\![Q]\!]_\sigma^{id} \mid [\![Q]\!]_\sigma^!$. By *2.* we have that $[\![Q]\!]_\sigma^! \equiv R'$ and by induction hypothesis we have that $[\![Q]\!]_\sigma^{id} \equiv R''$ hence $[\![(id')Q]\!]_\sigma^{id} \equiv R'' \mid R'$.

Case P is $\mathbf{0}$:

1. Direct since $[\![\mathbf{0}]\!]_{\sigma'}^!$ is defined as $\mathbf{0}$.

2-4. Likewise.

Case P is X:

1. Does not apply.
2. We have that $[\![X]\!]_\sigma^!$ by definition is $[\![\sigma(X)]\!]_\sigma^!$ and by hypothesis we have that $[\![\sigma(X)]\!]_{\sigma''}^! \equiv R'''$ for any σ''.
3. Direct since $[\![X]\!]_\sigma^?$ by definition is $\mathbf{0}$.
4. Likewise.

Case P is rec $X.Q$:

1. We have that $[\![\mathbf{rec}\ X.Q]\!]_{\sigma'}^!$ is defined as $[\![Q]\!]_{\sigma'[X \mapsto Q]}^!$. By induction hypothesis we have that $[\![Q]\!]_{\sigma''}^! \equiv R$ for any σ'', hence $[\![\mathbf{rec}\ X.Q]\!]_{\sigma'}^! \equiv R$ for any σ'.
2. We have that $[\![\mathbf{rec}\ X.Q]\!]_\sigma^!$ is defined as $[\![Q]\!]_{\sigma[X \mapsto Q]}^!$. By *1.* we conclude that $[\![Q]\!]_{\sigma''}^! \equiv R$ for any σ''. Then, by induction hypothesis we have that $[\![Q]\!]_{\sigma[X \mapsto Q]}^! \equiv R'$ and hence $[\![\mathbf{rec}\ X.Q]\!]_\sigma^! \equiv R'$.

3-4. Likewise.

Case P is $Q_1 \mid Q_2$:

1. We have that $[\![Q_1 \mid Q_2]\!]_{\sigma'}^!$ by definition is $[\![Q_1]\!]_{\sigma'}^! \mid [\![Q_2]\!]_{\sigma'}^!$ hence the result follows from the induction hypothesis.

2-4. Likewise.

Case P is $S_1 + S_2$:

1. We have that $[\![S_1 + S_2]\!]_{\sigma'}^!$ by definition is $[\![S_1]\!]_{\sigma'}^! + [\![S_2]\!]_{\sigma'}^!$ hence the result follows from the induction hypothesis.
2. Likewise.
3. We have that $[\![S_1 + S_2]\!]_\sigma^?$ by definition is $[\![S_1]\!]_\sigma^? \mid [\![S_2]\!]_\sigma^?$ hence the result follows from the induction hypothesis.
4. Likewise.

□

We may also inform on a specific case of the *id*-projection, namely regarding static protocols that do not include any occurrence of the (id)-construct (cf. Definition 2.7). Since *id*-projection captures the enabling behaviours of active nodes specified in the protocol, we have that if there are no active nodes in the protocol then the *id*-projection yields inaction.

Lemma 5.3 (Static Protocol *id*-Projection)**.** *If P is a static protocol then $[\![P]\!]_\sigma^{id} \equiv \mathbf{0}$.*

Proof. By induction on the structure of P following expected lines. □

The *id*-projection targets the enabling behaviours associated with active nodes, and the following result allows for a more precise characterisation of this notion. Intuitively, we have that for each active node construct in the protocol, the *id*-projection yields the reaction obtained by the respective !-projection in parallel with the projection of the rest of the protocol.

Lemma 5.4 (Active Node Projection)**.** *If* $P = \mathscr{C}[(id)Q]$ *then* $[\![P]\!]^{id}_{\emptyset} \equiv [\![\mathscr{C}[Q]]\!]^{id}_{\emptyset} \mid [\![Q]\!]^{!}_{\emptyset}$ *and* $[\![P]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![\mathscr{C}[Q]]\!]^{\mathtt{r}}_{\emptyset}$ *when* $\mathtt{r} \neq id$*.*

Proof. By induction on the structure of $\mathscr{C}[\cdot]$.

Case $\mathscr{C}[\cdot]$ is $\cdot$

We have that $P = (id)Q$ and by definition $[\![(id)Q]\!]^{id}_{\emptyset} = [\![Q]\!]^{id}_{\emptyset} \mid [\![Q]\!]^{!}_{\emptyset}$ and $[\![(id)Q]\!]^{\mathtt{r}}_{\emptyset} = [\![Q]\!]^{\mathtt{r}}_{\emptyset}$ when $\mathtt{r} \neq id$.

Case $\mathscr{C}[\cdot]$ is $P' \mid \mathscr{C}'[\cdot]$

We have that $P = P' \mid \mathscr{C}'[(id)Q]$ and by definition

$$[\![P' \mid \mathscr{C}'[(id)Q]]\!]^{\mathtt{r}}_{\emptyset} = [\![P']\!]^{\mathtt{r}}_{\emptyset} \mid [\![\mathscr{C}'[(id)Q]]\!]^{\mathtt{r}}_{\emptyset} \tag{1}$$

both when $\mathtt{r} = id$ and otherwise. By induction hypothesis we have that

$$[\![\mathscr{C}'[(id)Q]]\!]^{id}_{\emptyset} \equiv [\![\mathscr{C}'[Q]]\!]^{id}_{\emptyset} \mid [\![Q]\!]^{!}_{\emptyset} \tag{2}$$

and that

$$[\![\mathscr{C}'[(id)Q]]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![\mathscr{C}'[Q]]\!]^{\mathtt{r}}_{\emptyset} \text{ for } \mathtt{r} \neq id \tag{3}$$

From (1) and (2) and also by definition of projection we conclude

$$[\![P' \mid \mathscr{C}'[(id)Q]]\!]^{id}_{\emptyset} \equiv [\![P']\!]^{id}_{\emptyset} \mid [\![\mathscr{C}'[Q]]\!]^{id}_{\emptyset} \mid [\![Q]\!]^{!}_{\emptyset} \equiv [\![P' \mid \mathscr{C}'[Q]]\!]^{id}_{\emptyset} \mid [\![Q]\!]^{!}_{\emptyset}$$

and from (1) and (3) and also by definition of projection we conclude

$$[\![P' \mid \mathscr{C}'[(id)Q]]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![P']\!]^{\mathtt{r}}_{\emptyset} \mid [\![\mathscr{C}'[Q]]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![P' \mid \mathscr{C}'[Q]]\!]^{\mathtt{r}}_{\emptyset} \text{ for } \mathtt{r} \neq id$$

Case $\mathscr{C}[\cdot]$ is $S + \mathscr{C}'[\cdot]$

Follows similar lines.

Case $\mathscr{C}[\cdot]$ is $(id')\mathscr{C}'[\cdot]$

We have that $P = (id')\mathscr{C}'[(id)Q]$ and by definition

$$[\![(id')\mathscr{C}'[(id)Q]]\!]^{\mathtt{r}}_{\emptyset} \text{ is } [\![\mathscr{C}'[(id)Q]]\!]^{\mathtt{r}}_{\emptyset} \mid [\![\mathscr{C}'[(id)Q]]\!]^{!}_{\emptyset} \text{ in case } \mathtt{r} = id' \tag{4}$$

and

$$[\![(id')\mathscr{C}'[(id)Q]]\!]^{\mathtt{r}}_{\emptyset} \text{ is } [\![\mathscr{C}'[(id)Q]]\!]^{\mathtt{r}}_{\emptyset} \text{ in case } \mathtt{r} \neq id'$$

regardless if $id = id'$ or $id \neq id'$.

Case $\mathtt{r} \neq id'$**:** The result follows directly from the induction hypothesis, where in particular we have

$$[\![\mathscr{C}'[(id)Q]]\!]^{!}_{\emptyset} \equiv [\![\mathscr{C}'[Q]]\!]^{!}_{\emptyset} \tag{5}$$

Case $\mathtt{r} = id'$**:** By induction hypothesis we have that

$$[\![\mathscr{C}'[(id)Q]]\!]^{id'}_{\emptyset} \equiv [\![\mathscr{C}'[Q]]\!]^{id'}_{\emptyset} \mid [\![Q]\!]^{!}_{\emptyset} \text{ when } id = id' \tag{6}$$

and that

$$[\![\mathscr{C}'[(id)Q]]\!]^{id'}_{\emptyset} \equiv [\![\mathscr{C}'[Q]]\!]^{id'}_{\emptyset} \text{ when } id \neq id'$$

When $id = id'$ from (4) and (6), and also by (5) and by definition of projection, we conclude

$$\begin{array}{l}[\![(id')\mathscr{C}'[(id)Q]]\!]^{id'}_{\emptyset} \equiv [\![\mathscr{C}'[Q]]\!]^{id'}_{\emptyset} \mid [\![Q]\!]^{!}_{\emptyset} \mid [\![\mathscr{C}'[(id)Q]]\!]^{!}_{\emptyset} \\ \equiv [\![\mathscr{C}'[Q]]\!]^{id'}_{\emptyset} \mid [\![Q]\!]^{!}_{\emptyset} \mid [\![\mathscr{C}'[Q]]\!]^{!}_{\emptyset} \equiv [\![(id')\mathscr{C}'[Q]]\!]^{id'}_{\emptyset} \mid [\![Q]\!]^{!}_{\emptyset}\end{array}$$

When $id \neq id'$ from (6) and by definition of projection we conclude

$$[\![(id')\mathscr{C}'[(id)Q]]\!]^{id'}_{\emptyset} \equiv [\![\mathscr{C}'[Q]]\!]^{id'}_{\emptyset} \mid [\![\mathscr{C}'[(id)Q]]\!]^{!}_{\emptyset} \equiv [\![\mathscr{C}'[Q]]\!]^{id'}_{\emptyset} \mid [\![\mathscr{C}'[Q]]\!]^{!}_{\emptyset} \equiv [\![(id')\mathscr{C}'[Q]]\!]^{id'}_{\emptyset}$$

Case $\mathscr{C}[\cdot]$ is $[f^d]^o_i\mathscr{C}'[\cdot]$

The case for id projection follows directly from induction hypothesis since by definition

$$[\![[f^d]^o_i\mathscr{C}'[(id)Q]]\!]^{id}_{\sigma} = [\![\mathscr{C}'[(id)Q]]\!]^{id}_{\sigma}$$

and likewise for id' projection ($id \neq id'$). The case for ! projection is direct since by definition both $[\![[f^d]^o_i\mathscr{C}'[(id)Q]]\!]^{!}_{\sigma}$ and $[\![[f^d]^o_i\mathscr{C}'[Q]]\!]^{!}_{\sigma}$ are $\langle o\rangle f^d!$. The case for ? projection follows from the induction hypothesis in expected lines.

□

The following result captures a fundamental property of (any type of) projection, namely that it is preserved under structural congruence.

Lemma 5.5 (Preservation of Projection Under Structural Congruence)**.** *If* $P \equiv Q$ *then* $[\![P]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![Q]\!]^{\mathtt{r}}_{\emptyset}$ *for any* $\mathtt{r}$*.*

Proof. The proof proceeds by induction on the shape of the derivation of $P \equiv Q$.

Case P **is** $Q \mid \mathbf{0}$**:** We need to prove that $[\![Q \mid \mathbf{0}]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![Q]\!]^{\mathtt{r}}_{\emptyset}$ for any $\mathtt{r}$. We have three cases depending on $\mathtt{r}$.

Case $\mathtt{r} = ?$**:** By definition of the projection function, we have that

$$[\![Q \mid \mathbf{0}]\!]^{?}_{\emptyset} = [\![Q]\!]^{?}_{\emptyset} \mid [\![\mathbf{0}]\!]^{?}_{\emptyset}$$

By the definition again, we know that $[\![\mathbf{0}]\!]^{?}_{\emptyset} = \mathbf{0}$ and now we have that $[\![Q \mid \mathbf{0}]\!]^{?}_{\emptyset} \equiv [\![Q]\!]^{?}_{\emptyset} \mid \mathbf{0} \equiv [\![Q]\!]^{?}_{\emptyset}$ as required.

Case $\mathtt{r} = !$**:** By definition of the projection function, we have that

$$[\![Q \mid \mathbf{0}]\!]^{!}_{\emptyset} = [\![Q]\!]^{!}_{\emptyset} \mid [\![\mathbf{0}]\!]^{!}_{\emptyset}$$

By the definition again, we know that $[\![\mathbf{0}]\!]^{!}_{\emptyset} = \mathbf{0}$ and now we have that $[\![Q \mid \mathbf{0}]\!]^{!}_{\emptyset} \equiv [\![Q]\!]^{!}_{\emptyset} \mid \mathbf{0} \equiv [\![Q]\!]^{!}_{\emptyset}$ as required.

Case $\mathtt{r} = id$**:** By definition of the projection function, we have that

$$[\![Q \mid \mathbf{0}]\!]^{id}_{\emptyset} = [\![Q]\!]^{id}_{\emptyset} \mid [\![\mathbf{0}]\!]^{id}_{\emptyset}$$

By the definition again, we know that $[\![\mathbf{0}]\!]^{id}_{\emptyset} = \mathbf{0}$ and now we have that $[\![Q \mid \mathbf{0}]\!]^{id}_{\emptyset} \equiv [\![Q]\!]^{id}_{\emptyset} \mid \mathbf{0} \equiv [\![Q]\!]^{id}_{\emptyset}$ as required.

Case P is $Q_1 \mid (Q_2 \mid Q_3)$: We need to prove that $[\![Q_1 \mid (Q_2 \mid Q_3)]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![(Q_1 \mid Q_2) \mid Q_3]\!]^{\mathtt{r}}_{\emptyset}$ for any $\mathtt{r}$. We have three cases depending on $\mathtt{r}$.

Case $\mathtt{r} =?$: By definition of the projection function, we have that

$$[\![Q_1 \mid (Q_2 \mid Q_3)]\!]^{?}_{\emptyset} = [\![Q_1]\!]^{?}_{\emptyset} \mid [\![Q_2 \mid Q_3]\!]^{?}_{\emptyset} = [\![Q_1]\!]^{?}_{\emptyset} \mid [\![Q_2]\!]^{?}_{\emptyset} \mid [\![Q_3]\!]^{?}_{\emptyset}$$

and

$$[\![(Q_1 \mid Q_2) \mid Q_3]\!]^{?}_{\emptyset} = [\![Q_1 \mid Q_2]\!]^{?}_{\emptyset} \mid [\![Q_3]\!]^{?}_{\emptyset} = [\![Q_1]\!]^{?}_{\emptyset} \mid [\![Q_2]\!]^{?}_{\emptyset} \mid [\![Q_3]\!]^{?}_{\emptyset} \quad \text{as required.}$$

Case $\mathtt{r} \in \{!, id\}$: Similar to the case of "?".

Case P is $Q_1 + (Q_2 + Q_3)$ or $Q_1 + Q_2$ or $Q_1 \mid Q_2$ or $(id)\mathbf{0}$: follow directly by the definition.

Case P is $(id)(Q_1 \mid Q_2)$: We need to prove that $[\![(id)(Q_1 \mid Q_2)]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![(id)Q_1]\!]^{\mathtt{r}}_{\emptyset} \mid [\![(id)Q_2]\!]^{\mathtt{r}}_{\emptyset}$ for any $\mathtt{r}$. We have three cases depending on $\mathtt{r}$.

Case $\mathtt{r} =?$: By definition of the projection function, we have that

$$[\![(id)(Q_1 \mid Q_2)]\!]^{?}_{\emptyset} = [\![Q_1 \mid Q_2]\!]^{?}_{\emptyset} = [\![Q_1]\!]^{?}_{\emptyset} \mid [\![Q_2]\!]^{?}_{\emptyset}$$

$$[\![(id)Q_1]\!]^{?}_{\emptyset} \mid [\![(id)Q_2]\!]^{?}_{\emptyset} = [\![Q_1]\!]^{?}_{\emptyset} \mid [\![Q_2]\!]^{?}_{\emptyset} \quad \text{as required.}$$

Case $\mathtt{r} =!$: Similar to the case of "?".

Case $\mathtt{r} = id$: We have two cases: $\mathtt{r} \neq id$ or $\mathtt{r} = id$. The former case is similar to $\{?, !\}$ while the latter case can be proved as follows;

$$[\![(id)(Q_1 \mid Q_2)]\!]^{id}_{\emptyset} = [\![Q_1 \mid Q_2]\!]^{id}_{\emptyset} \mid [\![Q_1 \mid Q_2]\!]^{!}_{\emptyset} = [\![Q_1]\!]^{id}_{\emptyset} \mid [\![Q_2]\!]^{id}_{\emptyset} \mid [\![Q_1]\!]^{!}_{\emptyset} \mid [\![Q_2]\!]^{!}_{\emptyset}$$

$$[\![(id)Q_1]\!]^{id}_{\emptyset} \mid [\![(id)Q_2]\!]^{id}_{\emptyset} = [\![Q_1]\!]^{id}_{\emptyset} \mid [\![Q_1]\!]^{!}_{\emptyset} \mid [\![Q_2]\!]^{id}_{\emptyset} \mid [\![Q_2]\!]^{!}_{\emptyset}$$

And we have that $[\![Q_1]\!]^{id}_{\emptyset} \mid [\![Q_2]\!]^{id}_{\emptyset} \mid [\![Q_1]\!]^{!}_{\emptyset} \mid [\![Q_2]\!]^{!}_{\emptyset} \equiv [\![Q_1]\!]^{id}_{\emptyset} \mid [\![Q_1]\!]^{!}_{\emptyset} \mid [\![Q_2]\!]^{id}_{\emptyset} \mid [\![Q_2]\!]^{!}_{\emptyset}$ as required.

Case P is $(id_1)(id_2)Q$: We need to prove that $[\![(id_1)(id_2)Q]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![(id_2)(id_1)Q]\!]^{\mathtt{r}}_{\emptyset}$ for any $\mathtt{r}$. We have three cases depending on $\mathtt{r}$.

Cases $\mathtt{r} \in \{?, !\}$: By definition $[\![(id_1)(id_2)Q]\!]^{?}_{\emptyset} \equiv [\![(id_2)(id_1)Q]\!]^{?}_{\emptyset} = [\![(id_1)(id_2)Q]\!]^{!}_{\emptyset} \equiv [\![(id_2)(id_1)Q]\!]^{!}_{\emptyset} = [\![Q]\!]^{?}_{\emptyset} = [\![Q]\!]^{!}_{\emptyset}$

Case $\mathtt{r} = id$: We have three cases:

1) $id \neq id_1 \wedge id \neq id_2$**:** By definition we have that $[\![(id_1)(id_2)Q]\!]^{id}_{\emptyset} \equiv [\![(id_2)(id_1)Q]\!]^{id}_{\emptyset} = [\![Q]\!]^{id}_{\emptyset}$.

2) $id = id_1$**:** By definition we have that:

$$[\![(id_1)(id_2)Q]\!]^{id_1}_{\emptyset} = [\![(id_2)Q]\!]^{id_1}_{\emptyset} \mid [\![(id_2)Q]\!]^{!}_{\emptyset} = [\![Q]\!]^{id_1}_{\emptyset} \mid [\![Q]\!]^{!}_{\emptyset}$$

$$[\![(id_2)(id_1)Q]\!]^{id_1}_{\emptyset} = [\![(id_1)Q]\!]^{id_1}_{\emptyset} = [\![Q]\!]^{id_1}_{\emptyset} \mid [\![Q]\!]^{!}_{\emptyset} \quad \text{as required.}$$

3) $id = id_2$**:** Similar to case (2).

Case P is rec $X.Q$: To prove that $[\![\mathbf{rec}\ X.Q]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![Q[\mathbf{rec}\ X.Q/X]]\!]^{\mathtt{r}}_{\emptyset}$ for any $\mathtt{r}$ we rely on an auxiliary result (see Appendix A, Lemma A.7), which builds on the fact that $[\![P]\!]^{?}_{\sigma} \mid [\![P]\!]^{?}_{\sigma} \equiv [\![P]\!]^{?}_{\sigma}$ (see Appendix A, Lemma A.5) relying on structural congruence axiom $\langle c \rangle f^{d}?.R \mid \langle c \rangle f^{d}?.R \equiv \langle c \rangle f^{d}?.R$.

□

As previously mentioned the *id*-projection disregards static protocols and exclusively targets the runtime behaviour, addressing protocol evolution so that whenever the protocol evolves so does the result of the *id*-projection. In contrast, the other types of projection (! and ?) exclusively target static protocols, ignoring the active node construct, and are hence invariant under protocol evolution as stated in the following result.

Lemma 5.6 (Preservation of !/?-Projections Under Reduction)**.** *If* $\Delta; P \to \Delta'; Q$ *then* $[\![P]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![Q]\!]^{\mathtt{r}}_{\emptyset}$ *where* $\mathtt{r} \in \{!, ?\}$.

Proof. By induction on the shape of the derivation of $\Delta; P \to \Delta'; Q$.

Case rule Bin is applied: We have that $\Delta; (id)([f^d]^o_i P) \to \Delta'; [f^d]^o_i((id')P)$ and we need to prove that $[\![(id)([f^d]^o_i P)]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![[f^d]^o_i((id')P)]\!]^{\mathtt{r}}_{\emptyset}$. We have different cases:

Case $\mathtt{r} = ?$**:** By definition of the projection function, we have that

$$[\![(id)([f^d]^o_i P)]\!]^{?}_{\emptyset} = \langle i \rangle f^d?.[\![P]\!]^{!}_{\emptyset} \mid [\![P]\!]^{?}_{\emptyset}$$

$$[\![[f^d]^o_i((id')P)]\!]^{?}_{\emptyset} = \langle i \rangle f^d?.[\![(id')P]\!]^{!}_{\emptyset} \mid [\![(id')P]\!]^{?}_{\emptyset}$$

By definition again, we know that $[\![(id')P]\!]^{!}_{\emptyset} = [\![P]\!]^{!}_{\emptyset}$ and $[\![(id')P]\!]^{?}_{\emptyset} = [\![P]\!]^{?}_{\emptyset}$ and we have that $[\![(id)([f^d]^o_i P)]\!]^{?}_{\emptyset} \equiv [\![[f^d]^o_i((id')P)]\!]^{?}_{\emptyset}$ as required.

Case $\mathtt{r} = !$**:** By definition of the projection function, we have that

$$[\![(id)([f^d]^o_i P)]\!]^{!}_{\emptyset} = \langle o \rangle f^d!$$

By definition again, we know that $[\![[f^d]^o_i((id')P)]\!]^{!}_{\emptyset} = \langle o \rangle f^d!$ as required.

Case rules Brd and Loc are applied: can be proved in a similar manner.

Case rules IdSum is applied: We need to prove that if $\Delta; (id)(S_1 + S_2) \to \Delta'; S'_1 + S_2$ then $[\![(id)(S_1 + S_2)]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![S'_1 + S_2]\!]^{\mathtt{r}}_{\emptyset}$. We have different cases:

Case $\mathtt{r} = ?$**:** By definition of the projection function, we have that

$$[\![(id)(S_1 + S_2)]\!]^{?}_{\emptyset} = [\![S_1]\!]^{?}_{\emptyset} \mid [\![S_2]\!]^{?}_{\emptyset}$$

$$[\![S'_1 + S_2]\!]^{?}_{\emptyset} = [\![S'_1]\!]^{?}_{\emptyset} \mid [\![S_2]\!]^{?}_{\emptyset}$$

By the induction hypothesis, we have that $[\![S']\!]^{?}_{\emptyset} = [\![S]\!]^{?}_{\emptyset}$ and we have that $[\![(id)(S_1 + S_2)]\!]^{?}_{\emptyset} \equiv [\![S'_1 + S_2]\!]^{?}_{\emptyset}$ as required.

Case $\mathtt{r} = !$**:** By definition of the projection function, we have that

$$[\![(id)(S_1 + S_2)]\!]^{!}_{\emptyset} = [\![(S_1 + S_2)]\!]^{!}_{\emptyset} = [\![S_1]\!]^{!}_{\emptyset} + [\![S_2]\!]^{!}_{\emptyset}$$

$$[\![S'_1 + S_2]\!]^{!}_{\emptyset} = [\![S'_1]\!]^{!}_{\emptyset} + [\![S_2]\!]^{!}_{\emptyset}$$

By the induction hypothesis, we have that $[\![S']\!]^{!}_{\emptyset} = [\![S]\!]^{!}_{\emptyset}$ and we have that $[\![(id)(S_1 + S_2)]\!]^{!}_{\emptyset} \equiv [\![S'_1 + S_2]\!]^{!}_{\emptyset}$ as required.

Case rule Synch is applied: We need to prove that if $\Delta; [f^d]^o_i P \to \Delta'; [f^d]^o_i P'$ then $[\![[f^d]^o_i P]\!]^{\mathtt{r}}_{\emptyset} \equiv [\![[f^d]^o_i P']\!]^{\mathtt{r}}_{\emptyset}$. We have different cases:

Case $\mathtt{r} = ?$**:** By definition of the projection function, we have that

$$[\![[f^d]^o_i P]\!]^{?}_{\emptyset} = \langle i \rangle f^d?.[\![P]\!]^{!}_{\emptyset} \mid [\![P]\!]^{?}_{\emptyset}$$

$$[\![[f^d]^o_i P')]\!]^{?}_{\emptyset} = \langle i \rangle f^d?.[\![P']\!]^{!}_{\emptyset} \mid [\![P']\!]^{?}_{\emptyset}$$

By the induction hypothesis, we have that $[\![P']\!]^{!}_{\emptyset} = [\![P]\!]^{!}_{\emptyset}$ and $[\![P']\!]^{?}_{\emptyset} = [\![P]\!]^{?}_{\emptyset}$ and we have that $[\![[f^d]^o_i P]\!]^{?}_{\emptyset} \equiv [\![[f^d]^o_i P']\!]^{?}_{\emptyset}$ as required.

Case $\mathtt{r} = !$**:** By definition of the projection function, we have that

$$[\![[f^d]_i^o P]\!]_\emptyset^! = [\![[f^d]_i^o P']\!]_\emptyset^! = \langle o \rangle f^d!$$

and we have that $[\![[f^d]_i^o P]\!]_\emptyset^! \equiv [\![[f^d]_i^o P']\!]_\emptyset^!$ as required.

Case Rule Id is applied: We need to prove that if $\Delta; (id)P \to \Delta'; (id)P'$ then $[\![(id)P]\!]_\emptyset^{\mathtt{r}} \equiv [\![(id)P']\!]_\emptyset^{\mathtt{r}}$. We have different cases:

Case $\mathtt{r} = ?$**:** By definition of the projection function, we have that

$$[\![(id)P]\!]_\emptyset^? = [\![P]\!]_\emptyset^? \quad \text{and} \quad [\![(id)P')]\!]_\emptyset^? = [\![P']\!]_\emptyset^?$$

By the induction hypothesis, we have that $[\![P']\!]_\emptyset^? = [\![P]\!]_\emptyset^?$ and we have that $[\![(id)P]\!]_\emptyset^? \equiv [\![(id)P']\!]_\emptyset^?$ as required.

Case $\mathtt{r} = !$**:** By definition of the projection function, we have that

$$[\![(id)P]\!]_\emptyset^! = [\![P]\!]_\emptyset^! \quad \text{and} \quad [\![(id)P')]\!]_\emptyset^! = [\![P']\!]_\emptyset^!$$

By the induction hypothesis, we have that $[\![P']\!]_\emptyset^! = [\![P]\!]_\emptyset^!$ and we have that $[\![(id)P]\!]_\emptyset^! \equiv [\![(id)P']\!]_\emptyset^!$ as required.

Case rule Sum is applied: We need to prove that if $\Delta; P_1 + P_2 \to \Delta'; P_1' + P_2$ then $[\![P_1 + P_2]\!]_\emptyset^{\mathtt{r}} \equiv [\![P_1' + P_2]\!]_\emptyset^{\mathtt{r}}$. We have different cases:

Case $\mathtt{r} = ?$**:** By definition of the projection function, we have that

$$[\![P_1 + P_2]\!]_\emptyset^? = [\![P_1]\!]_\emptyset^? \,|\, [\![P_2]\!]_\emptyset^?$$

$$[\![P_1' + P_2]\!]_\emptyset^? = [\![P_1']\!]_\emptyset^? \,|\, [\![P_2]\!]_\emptyset^?$$

By the induction hypothesis, we have that $[\![P_1']\!]_\emptyset^? = [\![P_1]\!]_\emptyset^?$ and we have that $[\![P_1 + P_2]\!]_\emptyset^? \equiv [\![P_1' + P_2]\!]_\emptyset^?$ as required.

Case $\mathtt{r} = !$**:** By definition of the projection function, we have that

$$[\![P_1 + P_2]\!]_\emptyset^! = [\![P_1]\!]_\emptyset^! + [\![P_2]\!]_\emptyset^!$$

$$[\![P_1' + P_2]\!]_\emptyset^! = [\![P_1']\!]_\emptyset^! + [\![P_2]\!]_\emptyset^!$$

By the induction hypothesis, we have that $[\![P_1']\!]_\emptyset^! = [\![P_1]\!]_\emptyset^!$ and we have that $[\![P_1 + P_2]\!]_\emptyset^! \equiv [\![P_1' + P_2]\!]_\emptyset^!$ as required.

Case rule Par is applied: We need to prove that if $\Delta; P_1 \,|\, P_2 \to \Delta'; P_1' \,|\, P_2$ then $[\![P_1 \,|\, P_2]\!]_\emptyset^{\mathtt{r}} \equiv [\![P_1' \,|\, P_2]\!]_\emptyset^{\mathtt{r}}$. This case can be proved in a similar manner to the previous case but by using rule PAR instead of rule SUM.

Case rule Struct is applied: We need to prove that if $\Delta; P \to \Delta'; Q$ then $[\![P]\!]_\emptyset^{\mathtt{r}} \equiv [\![Q]\!]_\emptyset^{\mathtt{r}}$.

From rule STRUCT, we have that $\Delta; P \to \Delta'; Q$ if $\Delta; P' \to \Delta'; Q'$ where $P' \equiv P$ and $Q' \equiv Q$. By the induction hypothesis we have that $[\![P']\!]_\emptyset^{\mathtt{r}} \equiv [\![Q']\!]_\emptyset^{\mathtt{r}}$. But $P' \equiv P$ and $Q' \equiv Q$, so we apply Lemma 5.5 and conclude the proof.

□

The next result captures that that for each enabling behaviour present in the result of an *id*-projection there is a correspondent active node scoping over the synchronisation (summation) in the source protocol.

Lemma 5.7 (Projection Inversion)**.** *Let P be a well-formed protocol such that $[\![P]\!]^{id}_{\emptyset} \equiv R \,|\, C$. We have that $P \equiv \mathscr{C}[(id)S]$ and $[\![S]\!]^{!}_{\emptyset} \equiv C$.*

Proof. By induction on the structure of P.

Case P is $[f^d]^o_i Q$: By definition we have that $[\![[f^d]^o_i Q]\!]^{id}_{\emptyset}$ is $[\![Q]\!]^{id}_{\emptyset}$. By induction hypothesis we have that $Q \equiv \mathscr{C}[(id)S]$ and $[\![S]\!]^{!}_{\emptyset} \equiv C$. Hence we have $P \equiv [f^d]^o_i \mathscr{C}[(id)S]$.

Case P is $(id')Q$: In case $id \neq id'$ the proof follows from the induction hypothesis. In case $id = id'$ we have that $[\![(id)Q]\!]^{id}_{\emptyset} \equiv R \,|\, C$. By definition we have that $[\![(id)Q]\!]^{id}_{\emptyset}$ is $[\![Q]\!]^{id}_{\emptyset} \,|\, [\![Q]\!]^{!}_{\emptyset}$, hence $[\![Q]\!]^{id}_{\emptyset} \,|\, [\![Q]\!]^{!}_{\emptyset} \equiv R \,|\, C$. We have two separate cases: either $[\![Q]\!]^{id}_{\emptyset} \equiv R' \,|\, C$ and $[\![Q]\!]^{!}_{\emptyset} \equiv R''$ or $[\![Q]\!]^{id}_{\emptyset} \equiv R'$ and $[\![Q]\!]^{!}_{\emptyset} \equiv R'' \,|\, C$.

Case $[\![Q]\!]^{id}_{\emptyset} \equiv R' \,|\, C$ and $[\![Q]\!]^{!}_{\emptyset} \equiv R''$: The proof follows from the induction hypothesis.

Case $[\![Q]\!]^{id}_{\emptyset} \equiv R'$ and $[\![Q]\!]^{!}_{\emptyset} \equiv R'' \,|\, C$: Since P is well-formed we have that recursion is guarded in $(id)Q$, hence from Proposition 2.9 we conclude $Q \equiv (\tilde{I_1})S_1 \,|\, \ldots \,|\, (\tilde{I_k})S_k$. We have that k is greater than 0 since C is a (non empty) choice. By definition we have that $[\![Q]\!]^{!}_{\emptyset}$ is $[\![S_1]\!]^{!}_{\emptyset} \,|\, \ldots \,|\, [\![S_k]\!]^{!}_{\emptyset}$, hence $[\![S_1]\!]^{!}_{\emptyset} \,|\, \ldots \,|\, [\![S_k]\!]^{!}_{\emptyset} \equiv R'' \,|\, C$. Thus there is $i \in 1, \ldots, k$ such that $[\![S_i]\!]^{!}_{\emptyset} \equiv C$. Finally we conclude $(id)Q \equiv (id)(\tilde{I_1})S_1 \,|\, \ldots \,|\, (\tilde{I_i})(id)S_i \,|\, \ldots \,|\, (id)(\tilde{I_k})S_k$.

Case P is $\mathbf{0}$: Impossible since C is a (non empty) choice.

Case P is X: Impossible since C is a (non empty) choice ($[\![X]\!]^{id}_{\emptyset}$ is $\mathbf{0}$).

Case P is rec $X.Q$: Impossible since P is a well-formed protocol ($[\![\mathbf{rec}\ X.Q]\!]^{id}_{\emptyset}$ is $\mathbf{0}$).

Case P is $Q_1 \,|\, Q_2$: Follows from the induction hypothesis.

Case P is $S_1 + S_2$: Follows from the induction hypothesis.

□

It is non-surprising that the properties above are crucial for the proof of operational correspondence addressed in the next section.

5.3. Operational Correspondence

In this section we present our operational correspondence result, where we match the evolutions of protocols with those of the obtained distributed implementation. For the purpose of matching reductions in the protocols we focus on the correspondent actions of the distributed implementation, hence on τ and on $id!\star^{\hat{f}}$ transition labels, which are ranged over by $\hat{\lambda}$. We remark that the correspondence matches each step in the global model with a single step in the distributed model and inversely.

Lemma 5.8 (Completeness) shows that the behaviour exhibited by the yielded distributed implementation is complete with respect to the global model, by proving that every possible evolution in the global model is matched by a correspondent one in the distributed implementation. We informally sketch the structure of the proof, which follows by induction on the shape of the derivation of $\Delta; P \rightarrow \Delta; Q$. For the base cases, the proof relies on the definition of projection and on Lemma 4.1, which shows that structurally equivalent language terms in the target implementation have equivalent behaviours. The proof for the language closure rules, namely Synch, Id, Sum and Par, does not follow directly from the induction hypotheses, and relies on Lemma 5.6 which ensures !/? projections are preserved under reduction. This is the case since both !/? projections work on the static structure of the protocol that is preserved by reduction. More precisely, only (id) operators float in the structure due to reduction, and that is why the id-projection is not preserved under reduction. Notice also that by the definition of the !/? projections, the presence of (id) operators has no effects on the result of the !/? projections (as shown by Lemma 5.4). Another result used in the proof of the closure rules shows that the parallel composition operator at the level of definitions in the target implementation is merely interleaving operator, and installing a new definition in a node only adds new

possible behaviour but it does not forbid already existing behaviours (see Appendix A, Lemma A.1). To conclude the proof of the completeness we also need to address the case of rule Struct in the global model, which relies on Lemma 5.5 that shows structurally equivalent protocols have also structurally equivalent projections (lifted to consider configurations in Appendix A, Lemma A.8).

Lemma 5.8 (Completeness)**.** *If* $\Delta; P \to \Delta'; Q$ *then* $[\![\Delta; P]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; Q]\!]$.

Proof. The proof proceeds by induction on the shape of the derivation of $\Delta; P \to \Delta'; Q$.

Case rule Bin is applied: We have that $\Delta; (id)([f^d]^o_i P) \to \Delta'; [f^d]^o_i((id')P)$ and we need to prove that $[\![\Delta; (id)([f^d]^o_i P)]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; [f^d]^o_i((id')P)]\!]$.

Since a binary interaction happened in the global model, we know that there must be a sender id with $\Delta(id) = s_1$ and a receiver id' with $\Delta(id') = s_2$ such that for some $d \in \{\blacktriangle, \blacktriangleright\}$ we have that $d(\Delta, id) = id'$, $s_1 \models o$ and $s_2 \models i$. As a result of synchronisation on f, we have also that $s_1' = f^d!(s_1, id')$ and $s_2' = f^{d}?(s_2, id)$. This can be concluded from the definition of $\mathtt{upd}(id, id', f^d, \Delta)$ and thus $\Delta' = \Delta[id \mapsto f^d!(s_1, id'), id' \mapsto f^{d}?(s_2, id)]$.

From the definition of the main projection rule in Table 8, we have that:

$$[\![\Delta; Q]\!] = \mathscr{N} \parallel s_1 : [\![Q]\!]^?_\emptyset \mid [\![Q]\!]^{id}_\emptyset \parallel s_2 : [\![Q]\!]^?_\emptyset \mid [\![Q]\!]^{id}_\emptyset$$

Where $Q = (id)([f^d]^o_i P)$ and $\mathscr{N}$ is the rest of the nodes in the network. We do not expand $\mathscr{N}$ because it does not contribute to the transition. By Table 8, we have that

$s_1 : [\![Q]\!]^?_\emptyset \mid [\![Q]\!]^{id}_\emptyset \triangleq s_1 : \langle i\rangle f^d?.[\![P]\!]^!_\emptyset \mid [\![P]\!]^?_\emptyset \mid \mid [\![P]\!]^{id}_\emptyset \mid (\langle o\rangle f^d!)$

$s_2 : [\![Q]\!]^?_\emptyset \mid [\![Q]\!]^{id}_\emptyset \triangleq s_2 : \langle i\rangle f^d?.[\![P]\!]^!_\emptyset \mid [\![P]\!]^?_\emptyset \mid [\![P]\!]^{id'}_\emptyset$

The overall network evolves by rule Com where $s_1 : D_1$ applies either rule oBinU or rule oBinR and $s_2 : D_2$ applies rule iBin, we have that:

$$\mathscr{N} \parallel s_1 : [\![Q]\!]^?_\emptyset \mid [\![Q]\!]^{id}_\emptyset \parallel s_2 : [\![Q]\!]^?_\emptyset \mid [\![Q]\!]^{id'}_\emptyset \xrightarrow{\tau} \mathscr{N} \parallel s_1 : D_1' \parallel s_2 : D_2'$$

$$D_1' = \langle i\rangle f^d?.[\![P]\!]^!_\emptyset \mid [\![P]\!]^?_\emptyset \mid \mid [\![P]\!]^{id}_\emptyset$$

$$D_2' = \langle i\rangle f^d?.[\![P]\!]^!_\emptyset \mid [\![P]\!]^!_\emptyset \mid [\![P]\!]^?_\emptyset \mid [\![P]\!]^{id'}_\emptyset$$

Now, we need to show that $[\![\Delta'; [f^d]^o_i((id')P)]\!] \equiv \mathscr{N} \parallel s_1 : D_1' \parallel s_2 : D_2'$. By Table 8, we have that

$$[\![\Delta'; \overbrace{[f^d]^o_i((id')P)}^{Q'}]\!] = \mathscr{N} \parallel s_1' : [\![Q']\!]^?_\emptyset \mid [\![Q']\!]^{id}_\emptyset \parallel s_2' : [\![Q']\!]^?_\emptyset \mid [\![Q']\!]^{id}_\emptyset$$

By applying the projection function, we have that $D_1' \equiv [\![Q']\!]^?_\emptyset \mid [\![Q']\!]^{id}_\emptyset$ and $D_2' \equiv [\![Q']\!]^?_\emptyset \mid [\![Q']\!]^{id'}_\emptyset$ as required.

Case rules Brd and Loc are applied: can be proved in a similar manner.

Case rule IdSum is applied: We need to prove that if $\Delta; (id)(S_1 + S_2) \to \Delta'; S_1' + S_2$ then $[\![\Delta; (id)(S_1 + S_2)]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; (S_1' + S_2)]\!]$. From rule IdSum, we know that $\Delta; (id)(S_1 + S_2) \to \Delta'; S_1' + S_2$ if $\Delta; (id)S_1 \to \Delta'; S_1'$. By the induction hypothesis, we have that $[\![\Delta; (id)S_1]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; S_1']\!]$.

By the definition of the projection function, we can rewrite the projection with respect to a single node id' with $\Delta(id') = s_1$ as follows: $[\![\Delta; (id)(S_1 + S_2)]\!] = \mathscr{N} \parallel \Delta(id') : [\![(id)(S_1 + S_2)]\!]^?_\emptyset \mid [\![(id)(S_1 + S_2)]\!]^{id'}_\emptyset$

where $\mathscr{N}$ is the rest of the nodes. By the projection function $[\![(id)S_1]\!]^?_\emptyset = [\![S_1]\!]^?_\emptyset$ and thus the induction hypothesis can be written as:

$$\mathscr{N} \parallel \Delta(id') : [\![S_1]\!]^?_\emptyset \mid [\![(id)S_1]\!]^{id'}_\emptyset \xrightarrow{\hat{\lambda}}_{\equiv} \mathscr{N}' \parallel \Delta'(id') : [\![S'_1]\!]^?_\emptyset \mid [\![S'_1]\!]^{id'}_\emptyset$$

For a node id', we have two cases for the projection of $[\![\Delta; (id)(S_1 + S_2)]\!]$:

Case $id' \neq id$**:** The projection of $[\![\Delta'; S'_1 + S_2]\!]$ proceeds as follows:

$$[\![\Delta'; S'_1 + S_2]\!] = \mathscr{N}' \parallel \Delta'(id') : [\![S'_1]\!]^?_\emptyset \mid [\![S'_1]\!]^{id'}_\emptyset \mid [\![S_2]\!]^?_\emptyset \mid [\![S_2]\!]^{id'}_\emptyset$$

and

$$[\![\Delta; (id)(S_1 + S_2)]\!] = \mathscr{N} \parallel \Delta(id') : [\![S_1]\!]^?_\emptyset \mid [\![S_1]\!]^{id'}_\emptyset \mid [\![S_2]\!]^?_\emptyset \mid [\![S_2]\!]^{id'}_\emptyset$$

By the induction hypothesis, by the closure of LTS under definition context (see Appendix A, Lemma A.1), and finally by applying rule Com, we have that

$$\mathscr{N} \parallel \Delta(id') : [\![S_1]\!]^?_\emptyset \mid [\![S_1]\!]^{id'}_\emptyset \mid [\![S_2]\!]^?_\emptyset \mid [\![S_2]\!]^{id'}_\emptyset \xrightarrow{\hat{\lambda}}_{\equiv} \mathscr{N}' \parallel \Delta'(id') : [\![S'_1]\!]^?_\emptyset \mid [\![S'_1]\!]^{id'}_\emptyset \mid [\![S_2]\!]^?_\emptyset \mid [\![S_2]\!]^{id'}_\emptyset$$

and $\mathscr{N}' \parallel \Delta'(id') : [\![S'_1]\!]^?_\emptyset \mid [\![S'_1]\!]^{id'}_\emptyset \mid [\![S_2]\!]^?_\emptyset \mid [\![S_2]\!]^{id'}_\emptyset$ is exactly the projection of $\Delta'; S'_1 + S_2$ as required.

Case $id' = id$**:** This case follows in a similar way.

Case rule Synch is applied: We need to prove that if $\Delta; [f^d]^o_i P \rightarrow \Delta'; [f^d]^o_i P'$ then $[\![\Delta; [f^d]^o_i P]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; [f^d]^o_i P']\!]$. From rule Synch, we know that $\Delta; [f^d]^o_i P \rightarrow \Delta'; [f^d]^o_i P'$ if $\Delta; P \rightarrow \Delta'; P'$. By the induction hypothesis, we have that $[\![\Delta; P]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; P']\!]$. By relying on the definition of the projection function, we have that:

$$\overbrace{\Pi_{\forall id \in dom(\Delta)}(\Delta(id) : [\![P]\!]^?_\emptyset \mid [\![P]\!]^{id}_\emptyset)}^{[\![\Delta;P]\!]} \xrightarrow{\hat{\lambda}}_{\equiv} \overbrace{\Pi_{\forall id \in dom(\Delta')}(\Delta'(id) : [\![P']\!]^?_\emptyset \mid [\![P']\!]^{id}_\emptyset)}^{[\![\Delta';P']\!]}$$

We also know by definition that

$$[\![\Delta; [f^d]^o_i P]\!] = \Pi_{\forall id \in dom(\Delta)}(\Delta(id) : \langle i \rangle f^d?.[\![P]\!]^!_\emptyset \mid [\![P]\!]^?_\emptyset \mid [\![P]\!]^{id}_\emptyset)$$

and

$$[\![\Delta'; [f^d]^o_i P']\!] = \Pi_{\forall id \in dom(\Delta')}(\Delta'(id) : \langle i \rangle f^d?.[\![P']\!]^!_\emptyset \mid [\![P']\!]^?_\emptyset \mid [\![P']\!]^{id}_\emptyset)$$

By Lemma 5.6, we have that $[\![P]\!]^!_\emptyset \equiv [\![P']\!]^!_\emptyset$ and thus we conclude the proof by the induction hypothesis, by closure of LTS under definition context (see Appendix A, Lemma A.1) and by applying rule Com.

Case rule Id is applied: We need to prove that if $\Delta; (id)P \rightarrow \Delta'; (id)P'$ then $[\![\Delta; (id)P]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; (id)P']\!]$. From rule Id, we know that $\Delta; (id)P \rightarrow \Delta'; (id)P'$ if $\Delta; P \rightarrow \Delta'; P'$. By the induction hypothesis, we have that $[\![\Delta; P]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; P']\!]$.

By the definition of the projection function, we can rewrite the projection with respect to a single node id' with $\Delta(id') = s_1$ as follows: $[\![\Delta; (id)P]\!] = \mathscr{N} \parallel \Delta(id') : [\![(id)P]\!]^?_\emptyset \mid [\![(id)P]\!]^{id'}_\emptyset$ where $\mathscr{N}$ is the rest of the nodes. By the induction hypothesis we have that:

$$\mathscr{N} \parallel \Delta(id') : [\![P]\!]^?_\emptyset \mid [\![P]\!]^{id'}_\emptyset \xrightarrow{\hat{\lambda}} \mathscr{N}' \parallel \Delta'(id') : [\![P']\!]^?_\emptyset \mid [\![P']\!]^{id'}_\emptyset$$

We have two cases for the projection of $[\![\Delta; (id)P]\!]$:

Case $id' \neq id$**:**

$$[\![\Delta; (id)P]\!] = \mathscr{N} \parallel \Delta(id') : [\![P]\!]^{?}_{\emptyset} \mid [\![P]\!]^{id'}_{\emptyset}$$

and

$$[\![\Delta'; (id)P']\!] = \mathscr{N}' \parallel \Delta'(id') : [\![P']\!]^{?}_{\emptyset} \mid [\![P']\!]^{id'}_{\emptyset}$$

Clearly, this case follows directly by the induction hypothesis.

Case $id' = id$**:** the projection proceeds as follows:

$$[\![\Delta; (id)P]\!] = \mathscr{N} \parallel \Delta(id') : [\![P]\!]^{?}_{\emptyset} \mid [\![P]\!]^{!}_{\emptyset} \mid [\![P]\!]^{id'}_{\emptyset}$$

and

$$[\![\Delta'; (id)P']\!] = \mathscr{N}' \parallel \Delta'(id') : [\![P']\!]^{?}_{\emptyset} \mid [\![P']\!]^{!}_{\emptyset} \mid [\![P']\!]^{id'}_{\emptyset}$$

By Lemma 5.6, we have that $[\![P]\!]^{!}_{\emptyset} \equiv [\![P']\!]^{!}_{\emptyset}$ and thus we conclude the proof by the induction hypothesis, by closure of LTS under definition context (see Appendix A, Lemma A.1) and by applying rule Com.

Case rule Sum is applied: We need to prove that if $\Delta; P_1 + P_2 \rightarrow \Delta'; P_1' + P_2$ then $[\![\Delta; P_1 + P_2]\!] \xrightarrow{\hat{\lambda}} \equiv [\![\Delta'; P_1' + P_2]\!]$. From rule Sum, we know that $\Delta; P_1 + P_2 \rightarrow \Delta'; P_1' + P_2$ if $\Delta; P_1 \rightarrow \Delta'; P_1'$. By the induction hypothesis, we have that $[\![\Delta; P_1]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; P_1']\!]$.

By the definition of the projection function, we can rewrite the projection with respect to a single node id' with $\Delta(id') = s_1$ as follows: $[\![\Delta; P_1 + P_2]\!] = \mathscr{N} \parallel \Delta(id') : [\![P_1 + P_2]\!]^{?}_{\emptyset} \mid [\![P_1 + P_2]\!]^{id'}_{\emptyset}$ where $\mathscr{N}$ is the rest of the nodes. By the induction hypothesis we have that:

$$\mathscr{N} \parallel \Delta(id') : [\![P_1]\!]^{?}_{\emptyset} \mid [\![P_1]\!]^{id'}_{\emptyset} \xrightarrow{\hat{\lambda}} \mathscr{N}' \parallel \Delta'(id') : [\![P_1']\!]^{?}_{\emptyset} \mid [\![P_1']\!]^{id'}_{\emptyset}$$

The projection of $[\![\Delta; P_1 + P_2]\!]$ proceeds as follows:

$$[\![\Delta; P_1 + P_2]\!] = \mathscr{N} \parallel \Delta(id') : [\![P_1]\!]^{?}_{\emptyset} \mid [\![P_1]\!]^{id'}_{\emptyset} \mid [\![P_2]\!]^{?}_{\emptyset} \mid [\![P_2]\!]^{id'}_{\emptyset}$$

and

$$[\![\Delta'; P_1' + P_2]\!] = \mathscr{N}' \parallel \Delta'(id') : [\![P_1']\!]^{?}_{\emptyset} \mid [\![P_1']\!]^{id'}_{\emptyset} \mid [\![P_2]\!]^{?}_{\emptyset} \mid [\![P_2]\!]^{id'}_{\emptyset}$$

We conclude the proof by the induction hypothesis, by closure of LTS under definition context (see Appendix A, Lemma A.1) and by applying rule Com.

Case rule Par is applied: We need to prove that if $\Delta; P_1 \mid P_2 \rightarrow \Delta'; P_1' \mid P_2$ then it is the case that $[\![\Delta; P_1 \mid P_2]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; P_1' \mid P_2]\!]$. This case can be proved in a similar manner to the previous case but by using rule Par instead of rule Sum.

Case rule Struct is applied: We need to prove that if $\Delta; P \rightarrow \Delta'; Q$ then $[\![\Delta; P]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; Q]\!]$.

From rule Struct, we have that $\Delta; P \rightarrow \Delta'; Q$ if $\Delta; P' \rightarrow \Delta'; Q'$ where $P' \equiv P$ and $Q' \equiv Q$. By relying on Lemma 5.5 (lifted to configurations in Appendix A, Lemma A.8), we have that $[\![\Delta; P]\!] \equiv [\![\Delta; P']\!]$ and $[\![\Delta; Q]\!] \equiv [\![\Delta; Q']\!]$ and by the induction hypothesis we have that $[\![\Delta; P]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; Q]\!]$ as required.

□

Lemma 5.9 (Soundness) says that each behaviour exhibited by the yielded projection has a correspondent step in the source configuration. The structure of the proof is as follows: first we use inversion results (see Appendix A, Lemma A.11 up to Lemma A.18) that allow us to characterise the structure of the definitions yielded by the projection, namely to identify the interacting input/output definitions. Then, we relate the output (choice) to a synchronisation action in the protocol (Projection Inversion, Lemma 5.7). We also have that the identified synchronisation action in the protocol must have a unique input counterpart in the projection (see Appendix A, Lemma A.9 and Lemma A.10), hence we may then relate the synchronisation action with the input involved in the interaction. Having identified the related synchronisation action we are able to derive the reduction, at which point we proceed to prove the relation between the arrival configuration with the network resulting from the interaction step. Crucial to this part of the proof is the characterisation of the Active Node Projection (Lemma 5.4) since reduction in configurations, as previously mentioned, is a change of active nodes: first the enabling node is active then the reacting nodes are activated. By means of Lemma 5.4 we are able to pinpoint that the projection of the enabling node differs only in the presence of the output (choice) before interaction, absent after interaction. We are also able to pinpoint that the projection of the reacting node differs only in the activation of the reaction in the continuation of the input after interaction, not active before interaction. Lemma 5.4 also allows to conclude that nodes not involved in the interaction have identical projections before and after the interaction. The precise characterisation allows us to relate the network resulting from the interaction step to the projection of the arrival configuration.

Lemma 5.9 (Soundness)**.** *If* $[\![\Delta; P]\!] \xrightarrow{\hat{\lambda}} \mathscr{N}$ *then* $\Delta; P \to \Delta'; Q$ *and* $\mathscr{N} \equiv [\![\Delta'; Q]\!]$.

Proof. (Sketch) We detail the proof for the case $\hat{\lambda} = \tau$ (in particular for child to parent binary interaction), remaining cases follow similar lines (namely for broadcast interaction when $\hat{\lambda} = id!\star^f$ which proof builds on Lemma A.17 and Lemma A.18).

Case $\hat{\lambda} = \tau$: By inversion on $[\![\Delta; P]\!] \xrightarrow{\tau} \mathscr{N}$ (Lemma A.11) we have that

$$[\![\Delta; P]\!] \equiv \mathscr{N}_1 \parallel \mathscr{N}_2 \text{ and } \mathscr{N}_1 \xrightarrow{id_1 \to {id_2}^f} \mathscr{N}_1', \mathscr{N}_2 \xrightarrow{id_2 \leftarrow {id_1}^f} \mathscr{N}_2', \text{ and } \mathscr{N} \equiv \mathscr{N}_1' \parallel \mathscr{N}_2' \tag{1}$$

or

$$[\![\Delta; P]\!] \equiv s : D \parallel \mathscr{N}'' \text{ and } s : D \xrightarrow{\tau} s : D' \text{ and } \mathscr{N} \equiv s : D' \parallel \mathscr{N}'' \tag{2}$$

or

$$[\![\Delta; P]\!] \equiv s : D \text{ and } s : D \xrightarrow{\tau} s : D' \text{ and } \mathscr{N} \equiv s : D' \tag{3}$$

We detail the proof for (1) and remark that the proof of (2) and (3) follow similar lines (inversion on Loc and the use of Lemma A.16 are the main differences).

Case (1): By inversion on $\mathscr{N}_1 \xrightarrow{id_1 \to {id_2}^f} \mathscr{N}_1'$ (Lemma A.13) we have that

$$\mathscr{N}_1 \equiv s_1 : D_1 \mid \langle o \rangle f^d! + C \parallel \mathscr{N}_1'' \text{ and } \mathscr{N}_1' \equiv s_1' : D_1 \parallel \mathscr{N}_1''$$

or

$$\mathscr{N}_1 \equiv s_1 : D_1 \mid \langle o \rangle f^d! + C \text{ and } \mathscr{N}_1' \equiv s_1' : D_1$$

and in both cases $s_1' = f^d!(s_1, id_2)$, $s_1 \models o$ and $id(s_1) = id_1$. Furthermore if $d = \blacktriangleright$ then $id_2 \in n(s_1)$ and if $d = \blacktriangle$ then $id_2 = i(s_1)$. We consider the case when $\mathscr{N}_1 \equiv s_1 : D_1 \mid \langle o \rangle f^d! + C \parallel \mathscr{N}_1''$ and $\mathscr{N}_1' \equiv s_1' : D_1 \parallel \mathscr{N}_1''$ and when $d = \blacktriangle$ hence $id_2 = i(s_1)$.
By definition of projection, we have that

$$[\![P]\!]^?_\emptyset \mid [\![P]\!]^{id_1}_\emptyset \equiv D_1 \mid \langle o \rangle f^{\blacktriangle}! + C \tag{4}$$

Considering Lemma 5.2 we have that there are D_1' and R_1 such that

$$D_1 \equiv D_1' \mid R_1 \text{ and } [\![P]\!]^?_\emptyset \equiv D_1' \text{ and } [\![P]\!]^{id_1}_\emptyset \equiv R_1 \mid \langle o \rangle f^{\blacktriangle}! + C \tag{5}$$

From Lemma 5.7 we conclude

$$P \equiv \mathscr{C}[(id_1)S] \text{ and } [\![S]\!]^!_\emptyset \equiv \langle o\rangle f^{\blacktriangle}! + C, \text{ hence } S \equiv [f^{\blacktriangle}]^o_i P' + S' \tag{6}$$

By inversion on $\mathscr{N}_2 \xrightarrow{id_2 \leftarrow id_1 f} \mathscr{N}_2'$ (Lemma A.15) we have that

$$\mathscr{N}_2 \equiv s_2 : D_2 \,|\, \langle i'\rangle f^{d'}?.R \,\|\, \mathscr{N}_2'' \text{ and } \mathscr{N}_2' \equiv s_2' : D_2 \,|\, \langle i'\rangle f^{d'}?.R \,|\, R \,\|\, \mathscr{N}_2''$$

or

$$\mathscr{N}_2 \equiv s_2 : D_2 \,|\, \langle i'\rangle f^{d'}?.R \text{ and } \mathscr{N}_2' \equiv s_2' : D_2 \,|\, \langle i'\rangle f^{d'}?.R \,|\, R$$

and in both cases $s_2' = f^{d'}?(s_2, id_1)$, $s_2 \models i'$, and $id(s_2) = id_2$. We consider the case when $\mathscr{N}_2 \equiv s_2 : D_2 \,|\, \langle i'\rangle f^{d'}?.R \,\|\, \mathscr{N}_2''$ and $\mathscr{N}_2' \equiv s_2' : D_2 \,|\, \langle i'\rangle f^{d'}?.R \,|\, R \,\|\, \mathscr{N}_2''$.
By definition of projection, we have that

$$[\![P]\!]^?_\emptyset \,|\, [\![P]\!]^{id_2}_\emptyset \equiv D_2 \,|\, \langle i'\rangle f^{d'}?.R \tag{7}$$

Considering Lemma 5.2 we have that there are D_2' and R_2 such that

$$D_2 \equiv D_2' \,|\, R_2 \text{ and } [\![P]\!]^?_\emptyset \equiv D_2' \,|\, \langle i'\rangle f^{d'}?.R \text{ and } [\![P]\!]^{id_2}_\emptyset \equiv R_2 \tag{8}$$

From (6) and considering an auxiliary result (Lemma A.9) we conclude $[\![P]\!]^?_\emptyset \equiv D' \,|\, \langle i\rangle f^{\blacktriangle}?.R'$ and $[\![P']\!]^!_\emptyset \equiv R'$. From input uniqueness (Lemma A.10) we conclude $i = i'$, $d' = \blacktriangle$ and $R \equiv R'$, hence $[\![P']\!]^!_\emptyset \equiv R$.
We then have that $s_1 \models o$, $id_2 = i(s_1)$, $s_2 \models i$ and we may obtain Δ' by considering $s_1' = f^{\blacktriangle}!(s_1, id_2)$ and $s_2' = f^{\blacktriangle}?(s_2, id_1)$. Considering Lemma 2.6 and by rules Bin and IdSum we conclude

$$\Delta; \mathscr{C}[(id_1)[f^{\blacktriangle}]^o_i P' + S'] \to \Delta'; \mathscr{C}[[f^{\blacktriangle}]^o_i((id_2)P') + S']$$

which proves there exists a reduction in the global model, hence completing the first part of the proof.
We now turn to proving it is a matching reduction. From $\mathscr{N} \equiv \mathscr{N}_1' \,\|\, \mathscr{N}_2'$ we conclude

$$\mathscr{N} \equiv s_1' : D_1 \,\|\, \mathscr{N}_1'' \,\|\, s_2' : D_2 \,|\, \langle i\rangle f^{\blacktriangle}?.R \,|\, R \,\|\, \mathscr{N}_2'' \tag{9}$$

From (4), (5) and (6) by Lemma 5.4 we conclude that

$$[\![P]\!]^{id_1}_\emptyset \equiv [\![\mathscr{C}[[f^{\blacktriangle}]^o_i P' + S']]\!]^{id_1}_\emptyset \,|\, [\![[f^{\blacktriangle}]^o_i P' + S']\!]^!_\emptyset \tag{10}$$

and also that

$$[\![P]\!]^?_\emptyset \equiv [\![\mathscr{C}[[f^{\blacktriangle}]^o_i P' + S']]\!]^?_\emptyset \tag{11}$$

Again by Lemma 5.4 and considering $id_2 \neq id_1$ (and $? \neq id_2$) we conclude

$$[\![\mathscr{C}[[f^{\blacktriangle}]^o_i P' + S']]\!]^{id_1}_\emptyset \equiv [\![\mathscr{C}[[f^{\blacktriangle}]^o_i((id_2)P') + S']]\!]^{id_1}_\emptyset$$

and

$$[\![\mathscr{C}[[f^{\blacktriangle}]^o_i P' + S']]\!]^?_\emptyset \equiv [\![\mathscr{C}[[f^{\blacktriangle}]^o_i((id_2)P') + S']]\!]^?_\emptyset$$

From the latter and considering (5), in particular $[\![P]\!]^?_\emptyset \equiv D_1'$, and considering (11) we conclude

$$[\![\mathscr{C}[[f^{\blacktriangle}]^o_i((id_2)P') + S']]\!]^?_\emptyset \equiv D_1' \tag{12}$$

From (5), in particular $[\![P]\!]^{id_1}_\emptyset \equiv R_1 \,|\, \langle o\rangle f^{\blacktriangle}! + C$, and (10) and considering $[\![[f^{\blacktriangle}]^o_i P' + S']\!]^!_\emptyset \equiv \langle o\rangle f^{\blacktriangle}! + C$ from (6) we conclude $[\![\mathscr{C}[[f^{\blacktriangle}]^o_i P' + S']]\!]^{id_1}_\emptyset \equiv R_1$. Then, by Lemma 5.4 we conclude

$$[\![\mathscr{C}[[f^{\blacktriangle}]^o_i((id_2)P') + S']]\!]^{id_1}_\emptyset \equiv R_1 \tag{13}$$

since $id_2 \neq id_1$. From (12) and (13) and considering $D_1 \equiv D_1' \mid R_1$, we conclude

$$s_1' : D_1 \equiv s_1' : [\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^? \mid [\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^{id_1} \tag{14}$$

From (7) and (8), considering that $i = i'$ and $d' = \blacktriangle$, and (6), by Lemma 5.4, since $? \neq id_1$ and $? \neq id_2$, we conclude

$$[\![\mathscr{C}[(id_1)[f^{\blacktriangle}]_i^o P' + S']]\!]_\emptyset^? \equiv [\![\mathscr{C}[[f^{\blacktriangle}]_i^o P' + S']]\!]_\emptyset^? \equiv [\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^?$$

and hence we have that

$$[\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^? \equiv D_2' \mid \langle i \rangle f^{\blacktriangle}?.R$$

From Lemma 5.4 we conclude

$$[\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^{id_2} \equiv [\![\mathscr{C}[[f^{\blacktriangle}]_i^o P' + S']]\!]_\emptyset^{id_2} \mid [\![P']\!]_\emptyset^!$$

Since $[\![P']\!]_\emptyset^! \equiv R$ we conclude

$$[\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^{id_2} \equiv [\![\mathscr{C}[[f^{\blacktriangle}]_i^o P' + S']]\!]_\emptyset^{id_2} \mid R$$

and since $id_1 \neq id_2$, again considering Lemma 5.4, we conclude

$$[\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^{id_2} \equiv [\![\mathscr{C}[(id_1)[f^{\blacktriangle}]_i^o P' + S']]\!]_\emptyset^{id_2} \mid R$$

Since $[\![P]\!]_\emptyset^{id_2} \equiv R_2$ we conclude

$$[\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^{id_2} \equiv R_2 \mid R$$

We then have that

$$[\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^? \equiv D_2' \mid \langle i \rangle f^{\blacktriangle}?.R \text{ and } [\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^{id_2} \equiv R_2 \mid R$$

and hence, considering $D_2 \equiv D_2' \mid R_2$ we conclude

$$s_2' : D_2 \mid \langle i \rangle f^{\blacktriangle}?.R \mid R \equiv s_2' : [\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^? \mid [\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^{id_2} \tag{15}$$

Considering Lemma 5.4, for any id' such that $id' \neq id_1$ and $id' \neq id_2$, we conclude

$$[\![P]\!]_\emptyset^? \mid [\![P]\!]_\emptyset^{id'} \equiv [\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^? \mid [\![\mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]_\emptyset^{id'}$$

Since also the changes in Δ' are localised to s_1' and s_2' we may show that $\mathscr{N}_1''$ and $\mathscr{N}_2''$ may be obtained correspondingly. Considering (9), (14) and (15) we may then conclude

$$\mathscr{N} \equiv [\![\Delta'; \mathscr{C}[[f^{\blacktriangle}]_i^o((id_2)P') + S']]\!]$$

□

We may now state our operational correspondence result which follows directly from Lemma 5.8 and Lemma 5.9. Theorem 5.10 states a one-to-one correspondence in the behaviours of global specifications and their translations. Notice that the possible transition labels ($\hat{\lambda}$) exclusively refer to synchronisations, either broadcast or binary, and local actions (hence $\hat{\lambda} = \tau$ or $\hat{\lambda} = id!\star^f$).

Theorem 5.10 (Operational Correspondence)**.** *We have that:*

1. *If* $\Delta; P \to \Delta'; Q$ *then* $[\![\Delta; P]\!] \xrightarrow{\hat{\lambda}}_{\equiv} [\![\Delta'; Q]\!]$.
2. *If* $[\![\Delta; P]\!] \xrightarrow{\hat{\lambda}} \mathscr{N}$ *then* $\Delta; P \to \Delta'; Q$ *and* $\mathscr{N} \equiv [\![\Delta'; Q]\!]$.

The operational correspondence result is key to allow for the development of protocols to be carried out in the global language, as any reasoning for the global model can be conveyed to the (automatically generated) implementation.

6. Concluding Remarks

In this paper we propose a language for specifying the coordination of operation control for power distribution grids. More precisely, we show how to program operation control protocols governing the behaviour of a power gird as a whole from a global perspective. Our language embeds the notion of yielding control in synchronisations and of communication driven by topological and state based-information by means of novel constructs. We formalise how global model specifications can be used to automatically synthesise individual controllers of the grid substations, yielding an operationally correct distributed implementation. Noticeably, the generated controller code is based on reactive definitions that are continuously able to react to incoming messages, up to state conditions. This suggests that the code running in a controller can be updated on-the-fly (e.g., to support a new protocol) by adding more definitions without modifying the already active code. Furthermore, the projection of static protocol specifications may be considered for any node since it is not node-specific, contrarily to network state and active projection. This suggests that nodes may be added on-the-fly to the concrete network, by installing the previously generated controller code for the static protocol while retaining the operation safety. We also present a non-trivial fault management scenario from the realm of power girds so as to illustrate the programming flavour of our global language.

We remark that the design principles of our model target power distribution grids specifically, namely considering that interaction is confined to closely follow the network topology. However, the principles showcased by our development can be used when considering different sorts of topologies. Conceivably, our principles can also be conveyed to other settings where operation protocols involve yielding control as a consequence of synchronisations. The notion of yielding control in communications is not new (e.g., token ring), but protocol languages based on this idea are lacking. We believe that our protocol language can be useful to program network behaviours in highly decentralised settings, such as wireless sensor and actuator networks (cf. [26]), where events trigger behaviours that involve network coordination.

We conclude this paper by discussing related approaches, focusing on formal models in particular, and by mentioning some directions for future work. Global protocol specifications can be found in the session type literature, spawning from the work of Honda et al. [27]. Such specifications are typically used to verify programs or guide their development. Other proposals enrich the global protocol specifications so as to allow programming to be carried out directly in the global language (e.g., [20, 21]). We insert our development in this latter context, since we also provide a global specification language where operation control protocols can be programmed. The distinguishing features of our approach naturally stem from the targeted setting. In particular, we consider that protocol specifications are role-agnostic and that the interacting parties are established operationally in the following sense: emitting/enabling parties correspond to active nodes and potential receiving/reacting parties are the ones that have a communication link with the active node, determined exclusively by the message direction and by the network configuration, and where message reception causes activation. We are unaware of related approaches that encompass this notion of yielding control in synchronisations, i.e., where the activated receiver is dynamically selected by the combination of who is the sender, the message direction and the network configuration.

In order to provide some intuition on this distinction, consider the following protocol:

$$\mathbf{rec}\ X.[\mathsf{Ping}^{\blacktriangleright}]^{\mathsf{tt}}_{\mathsf{tt}}X$$

that can be used to capture a "pingpong" like interaction between two (neighbouring) parties that exchange sender and receiver roles at each moment in time: consider a state Δ that specifies $\Delta(\mathbf{1}) = \mathbf{1}[z, 1, \{2\}, 0, 0, 0]$ and $\Delta(\mathbf{2}) = \mathbf{2}[z, 1, \{1\}, 0, 0, 0]$ and the evolution of configuration $\Delta; (\mathbf{1})(\mathbf{rec}\ \mathbf{X}.[\mathsf{Ping}^{\blacktriangleright}]^{\mathsf{tt}}_{\mathsf{tt}}\mathbf{X})$ to the configuration $\Delta; [\mathsf{Ping}^{\blacktriangleright}]^{\mathsf{tt}}_{\mathsf{tt}}(\mathbf{2})(\mathbf{rec}\ \mathbf{X}.[\mathsf{Ping}^{\blacktriangleright}]^{\mathsf{tt}}_{\mathsf{tt}}\mathbf{X})$. Notice that initially node **1** has the capability to send Ping and node **2** to receive, while after the evolution the association is switched (**2** can send Ping and **1** can receive). Hence, the protocol does not statically specify which node is activated by the communication, instead the activation results from the capability of receiving a message (determined by the $\blacktriangleright$ direction and by the fact that node **2** is a neighbour of **1** according to $\Delta(\mathbf{1})$). Notice also that the exact same protocol can be used to capture the interaction between three parties by considering a different network configuration (e.g., adding a node **3** to Δ such that $\Delta(\mathbf{3}) = \mathbf{3}[z, 1, \{1, 2\}, 0, 0, 0]$ and updating the neighbours of **2** and **3** accordingly, at which point if a node is initially active then either one of its neighbours can receive and become active and so forth).

In the realm of approaches that can also be placed in the context of [20, 21], we identify two proposals that are more closely related to our work, namely [19, 28]. The approach presented in [19] is related to ours both in goal and in approach, targeting power distribution grids in particular and relying on a global choreographic model (introduced in [22]). The focus is on specifying the operation control logic accounting for failures by means of specialised language principles. In contrast to our work, the association between processes and roles, once established, is fixed and cannot dynamically evolve. Hence, we do not see how to model the Ping protocol scenario above (both for two participants or for three). We believe accounting for failures is an interesting aspect to consider in the specification of the operation control logic, so integrating the language principles exploited in [19] in our work can prove to be worthwhile.

The language proposal presented in [28] supports the on-the-fly instantiation of roles by processes upon (choreographic-)procedure call. So, the Ping protocol above capturing the interaction of two participants can be modelled by considering the appropriate parameter switching (e.g., by means of a procedure defined along the lines of $\mathsf{Ping}(\mathsf{p},\mathsf{q}) = \mathsf{p} \to \mathsf{q}; \mathsf{Ping}(\mathsf{q},\mathsf{p})$ where the order of the parameters in the recursive call is switched). However, notice that such an implementation cannot capture the interaction between three participants like in our case. Noticeably, the primitive that allows to introduce participants proposed in [28] allows for dynamic reconfiguration of the communication topology without channel passing, and we believe adopting a related primitive is an interesting direction to extend our work.

In contrast with protocol specification languages such as session types [29], that give support for static verification/specification purposes, our approach is intended to serve as a programming mechanism. We nevertheless distinguish our language with respect to previous session-type based approaches following lines similar to the comparison above. Protocols specified in session types typically include role annotations that are impersonated by parties at runtime when agreeing to collaborate on a protocol. This yields a one-to-one correspondence between communication capabilities and parties, since only one role can send/receive a specific message (potentially an infinite number of times). Other approaches handle roles differently but retain a fixed (not subject to change at runtime) association between communication capabilities and parties: conversation types [30] are role-agnostic specifications like our (static) protocols; the approach presented in [31] includes role annotations where the association between roles and parties is not one-to-one; the approach presented in [32] allows for several parties to impersonate a single role. Our approach differs since communication capabilities are not assigned to nodes in a fixed way and can change at runtime.

With respect to related programming models, we distinguish *AbC* [25, 33] where communicating partners are also dynamically identified based on their states (of attributes). We remark that *AbC* does not support programming from a global point of view, but rather from a local distributed implementation view. Furthermore, in our approach the conditions specified in synchronisation actions refer to the local state of individual nodes (cf. the distributed implementation), while in *AbC* conditions refer to the communication partner. We also remark that the conversation calculus [34] introduces message directions similar to ours, but are there used to refer to the syntactic structure of processes (capturing, e.g., the caller-callee relation) and do not refer to the physical communication infrastructure like in our work.

For future work, we have already identified some language extensions that may prove to be worthwhile pursuing. One would be to lift the restriction on well-formed protocols that considers the active node construct may only appear (initially) top-level, in particular by allowing it to appear in (the body of) recursive protocols. This would allow to model nodes that are persistently active to synchronise on an action, up to satisfying a specific condition, hence potentially capturing sensor like behaviours that are "waiting" for a condition to hold so as to trigger behaviour. This would require extending the target model with persistent outputs so as to retain operational correspondence. Another extension would be to consider value passing so as to cope with dynamic reconfigurations that require more information is exchanged between nodes.

More importantly, it is definitely of interest to develop verification techniques for system level safety properties, and it seems natural to specify and check such properties considering the global model. To reason on protocol correctness and certify operation on all possible configurations, such verification techniques should focus on the static specification and abstract from the concrete network configuration, relying on some form of abstract interpretation (cf. [35, 36]). Thanks to the operational correspondence result that relates the global and distributed models, we can ensure that any property that holds for the global specification also holds for the distributed one. The reactive style of the distributed model suggests an implementation

based on the actor model [37, 38] (in, e.g., Erlang [39]). Conceivably, such reactive descriptions can be useful in other realms, in particular when addressing interaction models based on immediate reactions to external stimulus, such as the wireless sensor and actuator networks mentioned previously.

A. Operational Correspondence - Auxiliary Results

The following Lemma ensures that the parallel composition is merely interleaving and does not influence the behaviour of any of its sub-definitions, namely if a definition is able to take a step when isolated then this step will be possible also when put in parallel with any other definition.

Lemma A.1 (Closure of LTS Under Definition Context)**.** *If* $s : D_1 \xrightarrow{\hat{\lambda}} s' : D_2$ *then* $s : D_1 \mid D_3 \xrightarrow{\hat{\lambda}} s' : D_2 \mid D_3$ *for any* D_3*.*

Proof. By induction on the shape of the derivation of $s : D_1 \mid D_3 \xrightarrow{\hat{\lambda}} s' : D_2 \mid D_3$. □

The following results are used in the proof that structural congruence is preserved by projection, in particular regarding recursion unfolding. We start by addressing properties of the !-projection, namely, Lemma A.2 shows that it is preserved under any environment and process substitution since only the initial actions are relevant.

Lemma A.2 (Preservation of !-Projection)**.** *Let* P *be a protocol where recursion is guarded. We have that*

1. $[\![P]\!]^!_\sigma \equiv [\![P]\!]^!_{\sigma'}$ *for any* σ, σ'*.*
2. $[\![P[Q/X]]\!]^!_\sigma \equiv [\![P]\!]^!_\sigma$*.*

Proof. By induction on the structure of P following expected lines. Notice that !-projection addresses only immediate synchronisation actions, hence neither the mapping σ nor the substitution affect the projection given that recursion is guarded in P. □

Lemma A.2 is used directly in the proof of recursion unfolding for the !-projection and also to prove properties of the ?-projection. Lemma A.3 is also auxiliary to the case of ?-projection, showing the correspondence between environment and process substitution used in !-projection.

Lemma A.3 (Soundness of Mapping for !-Projection)**.** *Let* Q *be a protocol where recursion in guarded. We have that* $[\![P]\!]^!_{\sigma[X\mapsto Q]} \equiv [\![P[Q/X]]\!]^!_\sigma$*.*

Proof. By induction on the structure of P following expected lines. Notice that when P is X then we obtain $[\![Q]\!]^!_{\sigma[X\mapsto Q]}$ and $[\![Q]\!]^!_\sigma$ which may be equated considering Lemma A.2(*1*). □

We now address properties that directly regard the ?-projection, starting by Lemma A.4 that shows that processes in the environment are interchangeable as long as their !-projection is equivalent.

Lemma A.4 (Preservation of ?-Projection)**.** *Let* Q_1 *and* Q_2 *be protocols where recursion is guarded such that* $[\![Q_1]\!]^!_\sigma \equiv [\![Q_2]\!]^!_\sigma$*. We have that* $[\![P]\!]^?_{\sigma[X\mapsto Q_1]} \equiv [\![P]\!]^?_{\sigma[X\mapsto Q_2]}$*.*

Proof. By induction on the structure of P following expected lines. Notice that when P is X then we obtain $[\![Q_1]\!]^!_{\sigma[X\mapsto Q_1]}$ and $[\![Q_2]\!]^!_{\sigma[X\mapsto Q_2]}$ which may be equated considering Lemma A.2(*1*) and the hypothesis. □

The following result (Lemma A.5) is key to the proof of the unfolding, as it shows that two copies of a ?-projection are equivalent to one.

Lemma A.5 (Replicability of ?-Projection)**.** *We have that* $[\![P]\!]^?_\sigma \mid [\![P]\!]^?_\sigma \equiv [\![P]\!]^?_\sigma$*.*

Proof. By induction on the structure of P. The proof follows by induction hypothesis in expected lines for all cases except for the synchronisation action which also relies on axiom $\langle c\rangle f^d?.R \mid \langle c\rangle f^d?.R \equiv \langle c\rangle f^d?.R$. □

The main auxiliary properties for the case of recursion unfolding regarding ?-projection are given by the next result. Lemma A.6(*1*) equates the substitution with the environment, considering the involved process is in context. Lemma A.6(*2*) equates the substitution with the environment, considering a context with some (input) residua.

Lemma A.6 (Soundness of Mapping for ?-Projection)**.** *Let Q be a protocol where recursion is guarded.*

1. *We have that* $[\![P]\!]^?_{\sigma[X\mapsto Q]} \mid [\![Q]\!]^?_\sigma \equiv [\![P[Q/X]]\!]^?_\sigma \mid [\![Q]\!]^?_\sigma$.
2. *We have that* $[\![P[Q/X]]\!]^?_\sigma \equiv [\![P]\!]^?_{\sigma[X\mapsto Q]} \mid \Pi_{i\in I}\langle c_i\rangle f_i^{d_i}?.R_i$.

Proof. By induction on the structure of P. We sketch the proof of *1.*, the proof of *2.* follows similar lines.

Case P is $\mathbf{0}$: We have that $\mathbf{0} \mid [\![Q]\!]^?_\sigma \equiv \mathbf{0} \mid [\![Q]\!]^?_\sigma$.

Case P is X: Since $[\![X]\!]^?_{\sigma[X\mapsto Q]}$ by definition is $\mathbf{0}$, we have that $\mathbf{0} \mid [\![Q]\!]^?_\sigma \equiv [\![Q]\!]^?_\sigma \mid [\![Q]\!]^?_\sigma$ and the proof follows by considering Lemma A.5.

Case P is $(id)P'$ or $P_1 \mid P_2$ or $S_1 + S_2$: The proof follows from the induction hypothesis in expected lines.

Case P is $[f^d]^o_i P'$: Since $[\![[f^d]^o_i P']\!]^?_{\sigma[X\mapsto Q]}$ by definition is $\langle i\rangle f^d?.[\![P']\!]^!_{\sigma[X\mapsto Q]} \mid [\![P']\!]^?_{\sigma[X\mapsto Q]}$ we have that (i) $[\![[f^d]^o_i P']\!]^?_{\sigma[X\mapsto Q]} \mid [\![Q]\!]^?_\sigma \equiv \langle i\rangle f^d?.[\![P']\!]^!_{\sigma[X\mapsto Q]} \mid [\![P']\!]^?_{\sigma[X\mapsto Q]} \mid [\![Q]\!]^?_\sigma$. Considering Lemma A.3 we have that (ii) $\langle i\rangle f^d?.[\![P']\!]^!_{\sigma[X\mapsto Q]} \equiv \langle i\rangle f^d?.[\![P'[Q/X]]\!]^!_\sigma$. By induction hypothesis we conclude that (iii) $[\![P']\!]^?_{\sigma[X\mapsto Q]} \mid [\![Q]\!]^?_\sigma \equiv [\![P'[Q/X]]\!]^?_\sigma \mid [\![Q]\!]^?_\sigma$. Considering both (ii) and (iii) we conclude that (iv)

$$\langle i\rangle f^d?.[\![P']\!]^!_{\sigma[X\mapsto Q]} \mid [\![P']\!]^?_{\sigma[X\mapsto Q]} \mid [\![Q]\!]^?_\sigma \equiv \langle i\rangle f^d?.[\![P'[Q/X]]\!]^!_\sigma \mid [\![P'[Q/X]]\!]^?_\sigma \mid [\![Q]\!]^?_\sigma$$

By definition we have that $[\![[f^d]^o_i P'[Q/X]]\!]^?_\sigma \mid [\![Q]\!]^?_\sigma \equiv \langle i\rangle f^d?.[\![P'[Q/X]]\!]^!_\sigma \mid [\![P'[Q/X]]\!]^?_\sigma \mid [\![Q]\!]^?_\sigma$ which considering (i) and (iv) completes the proof.

Case P is rec $X.P'$: By definition we have that $[\![\mathbf{rec}\ X.P']\!]^?_{\sigma[X'\mapsto Q]} \mid [\![Q]\!]^?_\sigma \equiv [\![P']\!]^?_{\sigma[X'\mapsto Q][X\mapsto P']} \mid [\![Q]\!]^?_\sigma$. By induction hypothesis we have that $[\![P']\!]^?_{\sigma[X'\mapsto Q][X\mapsto P']} \mid [\![Q]\!]^?_\sigma \equiv [\![P'[Q/X']]\!]^?_{\sigma[X\mapsto P']} \mid [\![Q]\!]^?_\sigma$. Considering Lemma A.2(*2*) since recursion is guarded in P' we have that $[\![P'[Q/X']]\!]^!_\sigma \equiv [\![P']\!]^!_\sigma$, from which, considering Lemma A.4, we conclude $[\![P'[Q/X']]\!]^?_{\sigma[X\mapsto P']} \mid [\![Q]\!]^?_\sigma \equiv [\![P'[Q/X']]\!]^?_{\sigma[X\mapsto (P'[Q/X'])]} \mid [\![Q]\!]^?_\sigma$. The latter concludes the proof since by definition $[\![\mathbf{rec}\ X.P'[Q/X]]\!]^?_\sigma \mid [\![Q]\!]^?_\sigma \equiv [\![P'[Q/X']]\!]^?_{\sigma[X\mapsto (P'[Q/X'])]} \mid [\![Q]\!]^?_\sigma$.

□

We may now prove that projection is preserved under recursion unfolding. This lemma will be used in the proof of Lemma 5.5, where we want to show that the projection function is invariant to structural congruent protocols.

Lemma A.7 (Preservation of Projection under Unfolding)**.** *Let P be a static protocol where recursion is guarded. We have that* $[\![\mathbf{rec}\ X.P]\!]^{\mathtt{r}}_\emptyset \equiv [\![P[\mathbf{rec}\ X.P/X]]\!]^{\mathtt{r}}_\emptyset$ *for any* $\mathtt{r}$.

Proof. We prove the three cases separately.

Case $\mathtt{r} = !$: By definition we have $[\![\mathbf{rec}\ X.P]\!]^!_\emptyset \triangleq [\![P]\!]^!_{[X\mapsto P]}$. Since recursion is guarded in P, from Lemma A.2(*2*) we conclude $[\![P]\!]^!_{[X\mapsto P]} \equiv [\![P[\mathbf{rec}\ X.P/X]\!]^!_{[X\mapsto P]}$, and from Lemma A.2(*1*) we have $[\![P[\mathbf{rec}\ X.P/X]\!]^!_{[X\mapsto P]} \equiv [\![P[\mathbf{rec}\ X.P/X]\!]^!_\emptyset$.

Case $\mathtt{r} = id$: Since P is an (id)-absent protocol we have that $P[\mathbf{rec}\ X.P/X]$ is an (id)-absent protocol, hence the result follows immediately from Lemma 5.3 from which we conclude $[\![\mathbf{rec}\ X.P]\!]^{id}_\emptyset \equiv \mathbf{0}$ and $[\![P[\mathbf{rec}\ X.P/X]]\!]^{id}_\emptyset \equiv \mathbf{0}$.

Case $\mathtt{r} = ?$: From Lemma A.6(*1*) we have that

$$[\![P]\!]^{?}_{[X\mapsto \mathbf{rec}\ X.P]} \mid [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \equiv [\![P[\mathbf{rec}\ X.P/X]]\!]^{?}_{\emptyset} \mid [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset}$$

By definition we have that $[\![\mathbf{rec}\ X.P]\!]^{!}_{\emptyset} \equiv [\![P]\!]^{!}_{[X\mapsto P]}$ and since recursion is guarded in P from Lemma A.2(*1*) we have that $[\![P]\!]^{!}_{[X\mapsto P]} \equiv [\![P]\!]^{!}_{\emptyset}$ hence $[\![\mathbf{rec}\ X.P]\!]^{!}_{\emptyset} \equiv [\![P]\!]^{!}_{\emptyset}$. From this fact, considering Lemma A.4 we conclude

$$[\![P]\!]^{?}_{[X\mapsto P]} \mid [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \equiv [\![P[\mathbf{rec}\ X.P/X]]\!]^{?}_{\emptyset} \mid [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset}$$

By definition we have that $[\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \equiv [\![P]\!]^{?}_{[X\mapsto P]}$, hence

$$[\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \mid [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \equiv [\![P[\mathbf{rec}\ X.P/X]]\!]^{?}_{\emptyset} \mid [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset}$$

From Lemma A.5 we have that $[\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \mid [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \equiv [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset}$, hence (i)

$$[\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \equiv [\![P[\mathbf{rec}\ X.P/X]]\!]^{?}_{\emptyset} \mid [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset}$$

From Lemma A.6(*2*) we have that

$$[\![P[\mathbf{rec}\ X.P/X]]\!]^{?}_{\sigma} \equiv [\![P]\!]^{?}_{\sigma[X\mapsto \mathbf{rec}\ X.P]} \mid \Pi_{l\in L}\langle c_l\rangle f_l^{d_l}?.R_l$$

As before, from $[\![\mathbf{rec}\ X.P]\!]^{!}_{\emptyset} \equiv [\![P]\!]^{!}_{\emptyset}$ and considering Lemma A.4 we conclude

$$[\![P[\mathbf{rec}\ X.P/X]]\!]^{?}_{\sigma} \equiv [\![P]\!]^{?}_{\sigma[X\mapsto P]} \mid \Pi_{l\in L}\langle c_l\rangle f_l^{d_l}?.R_l$$

and, again as before, since by definition $[\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \equiv [\![P]\!]^{?}_{[X\mapsto P]}$ we conclude (ii)

$$[\![P[\mathbf{rec}\ X.P/X]]\!]^{?}_{\sigma} \equiv [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \mid \Pi_{l\in L}\langle c_l\rangle f_l^{d_l}?.R_l$$

which together with (i)

$$[\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \equiv [\![P[\mathbf{rec}\ X.P/X]]\!]^{?}_{\emptyset} \mid [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset}$$

allows us to conclude

$$[\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \equiv [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \mid \Pi_{l\in L}\langle c_l\rangle f_l^{d_l}?.R_l \mid [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset}$$

From this fact and considering Lemma A.5 we conclude

$$[\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \equiv [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \mid \Pi_{l\in L}\langle c_l\rangle f_l^{d_l}?.R_l$$

which together with (ii)

$$[\![P[\mathbf{rec}\ X.P/X]]\!]^{?}_{\sigma} \equiv [\![\mathbf{rec}\ X.P]\!]^{?}_{\emptyset} \mid \Pi_{l\in L}\langle c_l\rangle f_l^{d_l}?.R_l$$

completes the proof.

□

Lemma A.8 ensures that structurally equivalent protocols have structurally equivalent projections under any possible network state Δ.

Lemma A.8 (Preservation of Structural Congruence Under Network Projection)**.** *If $P \equiv Q$ then $[\![\Delta; P]\!] \equiv [\![\Delta; Q]\!]$ for any Δ.*

Proof. The proof proceeds by relying on the definition of the projection function and Lemma 5.5. By definition we have that:

$$[\![\Delta; P]\!] \triangleq \Pi_{\forall id \in dom(\Delta)}(\Delta(id) : [\![P]\!]^{?}_{\emptyset} \mid [\![P]\!]^{id}_{\emptyset})$$

$$[\![\Delta; Q]\!] \triangleq \Pi_{\forall id \in dom(\Delta)}(\Delta(id) : [\![Q]\!]^{?}_{\emptyset} \mid [\![Q]\!]^{id}_{\emptyset})$$

Since $P \equiv Q$, we have that, by Lemma 5.5 and regardless of Δ, $[\![P]\!]^{?}_{\emptyset} \equiv [\![Q]\!]^{?}_{\emptyset}$ and $[\![P]\!]^{id}_{\emptyset} \equiv [\![Q]\!]^{id}_{\emptyset}$. Thus $[\![\Delta; P]\!] \equiv [\![\Delta; Q]\!]$ as required. □

The following result states that for each synchronisation action in the protocol there is a corresponding input in the respective ?-projection, and that the continuation of the input is the reaction obtained by !-projecting the continuation of the synchronisation action.

Lemma A.9 (Synchronisation Action Projection)**.** *If $P \equiv \mathscr{C}[[f^d]^o_i Q]$ then $[\![P]\!]^{?}_{\emptyset} \equiv D \mid \langle i \rangle f^d?.R$ and $[\![Q]\!]^{!}_{\emptyset} \equiv R$.*

Proof. By induction on the structure of $\mathscr{C}[\cdot]$. Follows expected lines (cf. Lemma 5.4). □

The next result says ?-projections can be described as a parallel composition of distinguished inputs.

Lemma A.10 (Input Uniqueness)**.** *If P is a well-formed protocol then $[\![P]\!]^{?}_{\sigma} \equiv \Pi_{i \in I} \langle c_i \rangle f_i^{d_i}?.R_i$ where $f_j \neq f_l$ for all $j, l \in I$ such that $j \neq l$.*

Proof. (Sketch) Since well-formed protocol protocols originate from specifications where all action labels are distinct and we have that ? projection is invariant under reduction (Lemma 5.4) we may consider wlog that all action labels are distinct in P. The proof then follows by induction on the structure of P in expected lines (cf. Lemma 5.2(*3*)), by preserving the invariant that the set of labels $\{f_i \mid i \in I\}$ is the set of labels of P. □

The following results characterise the structure of networks and definitions given a specific observable behaviour.

Lemma A.11 (Inversion on Network Internal Step)**.** *If $\mathscr{N} \xrightarrow{\tau} \mathscr{N}'$ then either*

- *$\mathscr{N} \equiv \mathscr{N}_1 \parallel \mathscr{N}_2$ and $\mathscr{N}_1 \xrightarrow{id_1 \to id_2 \, f} \mathscr{N}_1'$, $\mathscr{N}_2 \xrightarrow{id_2 \leftarrow id_1 \, f} \mathscr{N}_2'$, and $\mathscr{N}' \equiv \mathscr{N}_1' \parallel \mathscr{N}_2'$.*
- *$\mathscr{N} \equiv s : D \parallel \mathscr{N}''$ and $s : D \xrightarrow{\tau} s : D'$ and $\mathscr{N}' \equiv s : D' \parallel \mathscr{N}''$.*
- *$\mathscr{N} \equiv s : D$ and $s : D \xrightarrow{\tau} s : D'$ and $\mathscr{N}' \equiv s : D'$.*

Proof. By induction on the shape of the derivation of $\mathscr{N} \xrightarrow{\tau} \mathscr{N}'$ following expected lines. □

Lemma A.12 (Inversion on Definition Output)**.** *If $D \xrightarrow{\langle c \rangle f^d!} D'$ then $D \equiv D' \mid \langle c \rangle f^d! + C$.*

Proof. By induction on the shape of the derivation of $D \xrightarrow{\langle c \rangle f^d!} D'$ following expected lines. □

Lemma A.13 (Inversion on Network Output)**.** *If $\mathscr{N} \xrightarrow{id_1 \to id_2 \, f} \mathscr{N}'$ then $\mathscr{N} \equiv s : D \mid \langle o \rangle f^d! + C \parallel \mathscr{N}''$ or $\mathscr{N} \equiv s : D \mid \langle o \rangle f^d! + C$, $\mathscr{N}' \equiv s' : D \parallel \mathscr{N}''$ or $\mathscr{N}' \equiv s' : D$ respectively, $s' = f^d!(s, id_2)$, $s \models o$ and $id(s) = id$. Furthermore if $d = \blacktriangleright$ then $id_2 \in n(s)$ and if $d = \blacktriangle$ then $id_2 = i(s)$.*

Proof. By induction on the shape of the derivation of $\mathscr{N} \xrightarrow{id_1 \to id_2 \, f} \mathscr{N}'$, where the base case follows by inversion on oBinR and oBinU and by considering Lemma A.12. □

Lemma A.14 (Inversion on Definition Input)**.** *If $D \xrightarrow{\langle c \rangle f^d?} D'$ then $D \equiv D'' \mid \langle c \rangle f^d?.R$ and $D' \equiv D'' \mid \langle c \rangle f^d?.R \mid R$.*

Proof. By induction on the shape of the derivation of $D \xrightarrow{\langle c \rangle f^{d}?} D'$ following expected lines. □

Lemma A.15 (Inversion on Network Input)**.** *If* $\mathscr{N} \xrightarrow{id_1 \leftarrow id_2{}^{f}} \mathscr{N}'$ *then* $\mathscr{N} \equiv s : D \,|\, \langle i \rangle f^{d}?.R \,\|\, \mathscr{N}''$ *or* $\mathscr{N} \equiv s : D \,|\, \langle i \rangle f^{d}?.R$, $\mathscr{N}' \equiv s' : D \,|\, \langle i \rangle f^{d}?.R \,|\, R \,\|\, \mathscr{N}''$ *or* $\mathscr{N}' \equiv s' : D \,|\, \langle i \rangle f^{d}?.R \,|\, R$ *respectively,* $s' = f^{d}?(s, id_2)$, $s \models i$, *and* $id(s) = id_1$.

Proof. By induction on the shape of the derivation of $\mathscr{N} \xrightarrow{id_1 \rightarrow id_2{}^{f}} \mathscr{N}'$, where the base case follows by inversion on iBin and by considering Lemma A.14. □

Lemma A.16 (Inversion on Definition Internal Step)**.** *If* $D \xrightarrow{\langle c \rangle f} D'$ *then* $D \equiv \langle i \rangle f^{\bullet}?.R \,|\, \langle o \rangle f^{\bullet}! + C \,|\, D''$ *and* $D' \equiv \langle i \rangle f^{\bullet}?.R \,|\, R \,|\, D''$ *and* $c = i \wedge o$.

Proof. By induction on the shape of the derivation of $D \xrightarrow{\langle c \rangle f} D'$ where the base case follows by inversion on Self and by considering Lemma A.12 and Lemma A.14. □

Lemma A.17 (Inversion on Network Broadcast Input)**.** *If* $\mathscr{N} \xrightarrow{id?\star^{f}} \mathscr{N}'$ *then* $\mathscr{N} \equiv \Pi_{j \in J}(s_j : D_j)$ *and* $\mathscr{N}' \equiv \Pi_{j \in J}(s_j : D'_j)$ *and* $\forall j \in J$ *it is the case that either (i)* $D_j \equiv D''_j \,|\, \langle i \rangle f^{\star}?.R_j$ *and* $s_j \models i$ *and* $i(s_j) = id$ *and* $D'_j \equiv D''_j \,|\, \langle i \rangle f^{\star}?.R_j \,|\, R_j$ *or (ii)* $discard(s_j : D_j, id?\star^{f})$ *and* $D'_j \equiv D_j$.

Proof. By induction on the shape of the derivation of $\mathscr{N} \xrightarrow{id?\star^{f}} \mathscr{N}'$, where the base case follows by inversion on iBrd and by considering Lemma A.14 or by inversion on dBrd. □

Lemma A.18 (Inversion on Network Broadcast Output)**.** *If* $\mathscr{N} \xrightarrow{id!\star^{f}} \mathscr{N}'$ *then* $\mathscr{N} \equiv s : D \mid \langle o \rangle f^{\star}! + C \,\|\, \mathscr{N}_1$ *or* $\mathscr{N} \equiv s : D \mid \langle o \rangle f^{\star}! + C$, $\mathscr{N}' \equiv s : D \,\|\, \mathscr{N}_2$ *and* $\mathscr{N}_1 \xrightarrow{id?\star^{f}} \mathscr{N}_2$ *or* $\mathscr{N}' \equiv s : D$, *respectively,* $s \models o$ *and* $id(s) = id$.

Proof. By induction on the shape of the derivation of $\mathscr{N} \xrightarrow{id!\star^{f}} \mathscr{N}'$, where the base case follows by inversion on oBrd and by considering Lemma A.12. □

References

[1] S. M. Amin, B. F. Wollenberg, Toward a smart grid: power delivery for the 21st century, IEEE Power and Energy Magazine 3 (5) (2005) 34–41. doi:10.1109/MPAE.2005.1507024.

[2] F. Andren, T. Strasser, W. Kastner, Towards a common modeling approach for smart grid automation, in: IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society, 2013, pp. 5340–5346. doi:10.1109/IECON.2013.6700004.

[3] SmartGrids: Vision and strategy for europe's electricity networks of the future, https://www.etip-snet.eu.

[4] IntelliGrid: Smart power for the 21^{st} century, http://smartgrid.epri.com/IntelliGrid.aspx.

[5] X. Han, K. Heussen, O. Gehrke, H. W. Bindner, B. Kroposki, Taxonomy for evaluation of distributed control strategies for distributed energy resources, IEEE Trans. Smart Grid 9 (5) (2018) 5185–5195. doi:10.1109/TSG.2017.2682924.

[6] T. I. Strasser, F. Andren, J. Kathan, C. Cecati, C. Buccella, P. Siano, P. Leitão, G. Zhabelova, V. Vyatkin, P. Vrba, V. Marík, A review of architectures and concepts for intelligence in future electric energy systems, IEEE Trans. Industrial Electronics 62 (4) (2015) 2424–2438. doi:10.1109/TIE.2014.2361486.

[7] W. Zhang, W. Liu, X. Wang, L. Liu, F. T. Ferrese, Distributed multiple agent system based online optimal reactive power control for smart grids, IEEE Trans. Smart Grid 5 (5) (2014) 2421–2431. doi:10.1109/TSG.2014.2327478.

[8] P. Vrba, V. Marík, P. Siano, P. Leitão, G. Zhabelova, V. Vyatkin, T. I. Strasser, A review of agent and service-oriented concepts applied to intelligent energy systems, IEEE Trans. Industrial Informatics 10 (3) (2014) 1890–1903. doi:10.1109/TII.2014.2326411.

[9] E. Ciancamerla, M. Minichino, M. C. Falvo, L. Martirano, Active distribution grids: A matlab-simulink tool for energy performance analysis, in: 2016 AEIT International Annual Conference (AEIT), 2016, pp. 1–5. doi:10.23919/AEIT.2016.7892749.

[10] C. Steinbrink, S. Lehnhoff, S. Rohjans, T. I. Strasser, E. Widl, C. Moyo, G. Lauss, F. Lehfuss, M. Faschang, P. Palensky, A. A. van der Meer, K. Heussen, O. Gehrke, E. Guillo-Sansano, M. H. Syed, A. Emhemed, R. Brandl, V. H. Nguyen, A. M. Khavari, Q. T. Tran, P. Kotsampopoulos, N. D. Hatziargyriou, N. Akroud, E. Rikos, M. Z. Degefa, Simulation-based validation of smart grids - status quo and future research trends, in: Industrial Applications of Holonic and Multi-Agent Systems - 8th International Conference, HoloMAS 2017, Proceedings, Vol. 10444 of LNCS, Springer, 2017, pp. 171–185. doi:10.1007/978-3-319-64635-0_13.

[11] R. Brandl, P. Kotsampopoulos, G. Lauss, M. Maniatopoulos, M. Nuschke, J. Montoya, T. I. Strasser, D. Strauss-Mincu, Advanced testing chain supporting the validation of smart grid systems and technologies, in: 2018 IEEE Workshop on Complexity in Engineering, COMPENG 2018, Florence, Italy, October 10-12, 2018, IEEE, 2018, pp. 1–6. doi:10.1109/CompEng.2018.8536223.

[12] A. Barbato, A. Capone, L. Chen, F. Martignon, S. Paris, A distributed demand-side management framework for the smart grid, Computer Communications 57 (2015) 13–24. doi:10.1016/j.comcom.2014.11.001.

[13] W. Tushar, B. Chai, C. Yuen, D. B. Smith, K. L. Wood, Z. Yang, H. V. Poor, Three-party energy management with distributed energy resources in smart grid, IEEE Trans. Industrial Electronics 62 (4) (2015) 2487–2498. doi:10.1109/TIE.2014.2341556.

[14] D. Papadaskalopoulos, D. Pudjianto, G. Strbac, Decentralized coordination of microgrids with flexible demand and energy storage, IEEE Transactions on Sustainable Energy 5 (4) (2014) 1406–1414. doi:10.1109/TSTE.2014.2311499.

[15] E. Dall'Anese, S. V. Dhople, B. B. Johnson, G. B. Giannakis, Decentralized optimal dispatch of photovoltaic inverters in residential distribution systems, IEEE Transactions on Energy Conversion 29 (4) (2014) 957–967.

[16] M. Yazdanian, A. Mehrizi-Sani, Distributed control techniques in microgrids, IEEE Trans. Smart Grid 5 (6) (2014) 2901–2909. doi:10.1109/TSG.2014.2337838.

[17] J. von Appen, T. Stetz, M. Braun, A. Schmiegel, Local voltage control strategies for PV storage systems in distribution grids, IEEE Trans. Smart Grid 5 (2) (2014) 1002–1009. doi:10.1109/TSG.2013.2291116.

[18] B. M. Eid, N. A. Rahim, J. Selvaraj, A. H. E. Khateb, Control methods and objectives for electronically coupled distributed energy resources in microgrids: A review, IEEE Systems Journal 10 (2) (2016) 446–458. doi:10.1109/JSYST.2013.2296075.

[19] H. A. López, K. Heussen, Choreographing cyber-physical distributed control systems for the energy sector, in: Proceedings of the 32nd ACM SIGAPP Symposium on Applied Computing, SAC 2017, ACM, 2017, pp. 437–443. doi:10.1145/3019612.3019656.

[20] M. Carbone, K. Honda, N. Yoshida, Structured communication-centered programming for web services, ACM Trans. Program. Lang. Syst. 34 (2) (2012) 8:1–8:78. doi:10.1145/2220365.2220367.

[21] M. Carbone, F. Montesi, Deadlock-freedom-by-design: multiparty asynchronous global programming, in: The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2013, ACM, 2013, pp. 263–274. doi:10.1145/2429069.2429101.

[22] H. A. López, F. Nielson, H. R. Nielson, Enforcing availability in failure-aware communicating systems, in: Formal Techniques for Distributed Objects, Components, and Systems - 36th IFIP WG 6.1 International Conference, FORTE 2016, Proceedings, Vol. 9688 of LNCS, Springer, 2016, pp. 195–211. doi:10.1007/978-3-319-39570-8_13.

[23] I. N. Kouveliotis-Lysikatos, D. Koukoula, N. D. Hatziargyriou, A double-layered fully distributed voltage control method for active distribution networks, IEEE Trans. Smart Grid 10 (2) (2019) 1465–1476. doi:10.1109/TSG.2017.2768239.

[24] M. Hennessy, J. Rathke, Bisimulations for a calculus of broadcasting systems, Theor. Comput. Sci. 200 (1-2) (1998) 225–260. doi:10.1016/S0304-3975(97)00261-2.

[25] Y. A. Alrahman, R. De Nicola, M. Loreti, A calculus for collective-adaptive systems and its behavioural theory, Information and Computation (2019) 104457doi:10.1016/j.ic.2019.104457.

[26] E. Khamespanah, M. Sirjani, K. Mechitov, G. Agha, Modeling and analyzing real-time wireless sensor and actuator networks using actors and model checking, STTT 20 (5) (2018) 547–561. doi:10.1007/s10009-017-0480-3.

[27] K. Honda, N. Yoshida, M. Carbone, Multiparty asynchronous session types, J. ACM 63 (1) (2016) 9:1–9:67. doi:10.1145/2827695.

[28] L. Cruz-Filipe, F. Montesi, Procedural choreographic programming, in: Formal Techniques for Distributed Objects, Components, and Systems - 37th IFIP WG 6.1 International Conference, FORTE 2017, Proceedings, Vol. 10321 of LNCS, Springer, 2017, pp. 92–107. doi:10.1007/978-3-319-60225-7_7.

[29] H. Hüttel, I. Lanese, V. T. Vasconcelos, L. Caires, M. Carbone, P. Deniélou, D. Mostrous, L. Padovani, A. Ravara, E. Tuosto, H. T. Vieira, G. Zavattaro, Foundations of session types and behavioural contracts, ACM Comput. Surv. 49 (1) (2016) 3:1–3:36. doi:10.1145/2873052.

[30] L. Caires, H. T. Vieira, Conversation types, Theor. Comput. Sci. 411 (51-52) (2010) 4399–4440. doi:10.1016/j.tcs.2010.09.010.

[31] P. Baltazar, L. Caires, V. T. Vasconcelos, H. T. Vieira, A type system for flexible role assignment in multiparty communicating systems, in: Trustworthy Global Computing - 7th International Symposium, TGC 2012, Revised Selected Papers, Vol. 8191 of LNCS, Springer, 2012, pp. 82–96. doi:10.1007/978-3-642-41157-1_6.

[32] P. Deniélou, N. Yoshida, Dynamic multirole session types, in: Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, ACM, 2011, pp. 435–446. doi:10.1145/1926385.1926435.

[33] Y. A. Alrahman, R. De Nicola, M. Loreti, F. Tiezzi, R. Vigo, A calculus for attribute-based communication, in: Proceedings of the 30th ACM SIGAPP Symposium on Applied Computing, SAC 2015, ACM, 2015, pp. 1840–1845. doi:10.1145/2695664.2695668.

[34] H. T. Vieira, L. Caires, J. C. Seco, The conversation calculus: A model of service-oriented computation, in: Programming Languages and Systems, 17th European Symposium on Programming, ESOP 2008, Proceedings, Vol. 4960 of LNCS, Springer, 2008, pp. 269–283. doi:10.1007/978-3-540-78739-6_21.

[35] P. Cousot, R. Cousot, Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints, in: Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, POPL 1977, ACM, 1977, pp. 238–252. doi:10.1145/512950.512973.

[36] P. Cousot, R. Cousot, Systematic design of program analysis frameworks, in: Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages, POPL 1979, ACM Press, 1979, pp. 269–282. doi:10.1145/

567752.567778.
[37] G. A. Agha, ACTORS - a model of concurrent computation in distributed systems, MIT Press series in artificial intelligence, MIT Press, 1990.
[38] C. J. Callsen, G. Agha, Open heterogeneous computing in actor space, J. Parallel Distrib. Comput. 21 (3) (1994) 289–300. doi:10.1006/jpdc.1994.1060.
[39] J. Armstrong, Programming Erlang: Software for a Concurrent World, Pragmatic Bookshelf, 2007.