Multiparty Session Nets

Luca Fossati[1], Raymond Hu[2], and Nobuko Yoshida[2]

[1]University of Cambridge [2]Imperial College London

Abstract. This paper introduces global session nets, an integration of multiparty session types (MPST) and Petri nets, for role-based choreographic specifications to verify distributed multiparty systems. The graphical representation of session nets enables more liberal combinations of branch, merge, fork and join patterns than the standard syntactic MPST. We use session net token dynamics to verify a flexible conformance between the graphical global net and syntactic endpoint types, and apply the conformance to ensure type-safety and progress of endpoint processes with channel mobility. We have implemented Java APIs for validating global session graph well-formedness and endpoint type conformance.

1 Introduction

Backgrounds and motivations In the early 2000s, there was an active debate on the ways in which various foundations could be applied to the description and verification of Web service standards, triggered by both researchers and developers working on Web services. Two of the major formalisms actively discussed are Petri nets and the π-calculus: the former can offer flexible graphical models of parallel workflows, while the latter can describe process interactions and mobility of channels in a textual format. A working group called Petri-and-Pi was led by Milner and van der Aalst in 2004 to seek meeting points. As a direction in a similar vein, this paper develops a new graphical formulation of multiparty session types (MPSTs) [12] based on Petri nets (PNs) that we call *session nets*. Our main motivations are (1) to offer graphical global specifications based on Petri nets that cannot be directly represented in MPST systems based on "linear" syntactic types [2, 4, 9, 12]; and (2) to apply Petri net token dynamics to a *conformance validation* which can guarantee independent endpoint processes satisfy safety and progress. We believe the resulting graphical representation, similar to notations used in BPMN [3] and UML [18], and accompanying token model will help engineers to write and understand MPST global protocols..

Session nets An MPST framework starts with global descriptions of the message passing protocols by which the participants should interact. In session nets (Figure 1), global protocols are specified by a combination of multiparty role ($\mathtt{A}, \mathtt{B}, \mathtt{C}, \ldots$) and message ($a, b, c, \ldots$) information over a PN control flow structure. Global session execution is modelled by standard PN token dynamics: branches and merges at places correspond to internal and external choices in the protocol flow at the specified roles; unlabelled transitions correspond to internal fork/join synchronisations; and labelled transitions to observable message I/O actions (e.g. $?a$ and $!a$). Decoupling I/O transitions gives a natural asynchronous model.

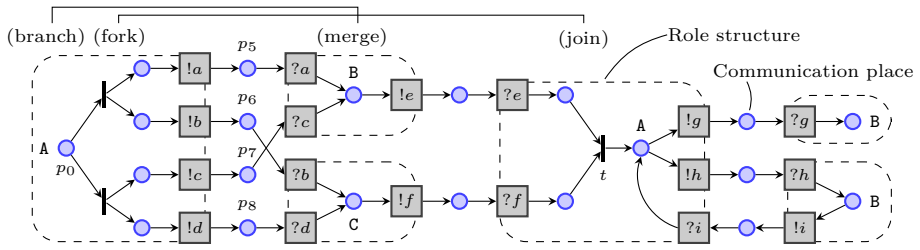


Fig. 1. Interleaved (i.e. non-nested) choice (branch-merge) and parallel (fork-join) structures with "criss-crossing" paths, leading to a recursive protocol segment.

The session net in Figure 1 cannot be represented by the global type syntax of [2, 4, 9, 12]. Firstly, because the "criss-crossing" of the middle two of the four paths from p_0 to t cannot be expressed in the tree structure of a linear syntax. Secondly, each of these paths flows from the initial branch through a fork, but then goes through a merge before the join. This interleaving of choice (branch-merge) and parallel (fork-join) structures is not supported by the nesting of choice and parallel constructors imposed by standard global type syntax. Appendix A shows session graphs of larger application protocols from [3].

Due to the flexibility of PN structures, a key design point in session nets is to characterise the nets that correspond to coherent protocols that are safely realisable as a system of distributed, asynchronous endpoints. In our framework, well-formed session graphs guarantee that net execution exhibits safety, in PN terminology (i.e. 1-boundedness), and an MPST-based form of progress. Safety states that no place is ever occupied by more than one token at a time. Progress means that every marking reachable from the initial marking enables a transition or is a terminal marking, in which tokens occupy only terminal places. § 2 defines session graphs, which are free-choice PNs by construction, their well-formedness conditions, and shows the above properties.

Conformance Unlike the typical top-down projection from global to local types in previous MPST systems [2, 4, 9, 12], we introduce a *conformance* relation between syntactic endpoint types and well-formed nets. § 3 shows our conformance allows each endpoint type to be validated against a net independently, while guaranteeing that their behaviour in composition respects the behaviour of the global net. Conformance between syntactic endpoints and global graphs is also motivated by practice: developers of Web services and other distributed applications often use expressive graphical patterns [3, 7, 23] for global specifications, but implement the endpoint programs using relatively primitive send/receive APIs, such as network socket or RPC interfaces. Our conformance accepts valid expansions of parallel specifications into a sequence of interleaved actions at the endpoint implementation level, and captures several session typing concepts, such as branch subtyping [8] and certain forms of asynchronous output permutations [6, 16].

Conformance is validated as a bidirectional I/O simulation between the (localised) net execution of a session graph and the behaviour of an individual role given by its type. Roughly speaking, conformance works by checking that every output specified by a local output type is accepted by the session net

(acting as an environment comprising the external roles), and that every message sent in the session net by another role to the local role is handled by a local input type. For example, T_1 and T_4 are different endpoint types that each conform to the session net in Figure 1 for the $\mathtt{A}$ role.

$$
\begin{array}{ll}
T_1 = !\{\mathtt{B}\langle a\rangle.!\mathtt{C}\langle b\rangle.T_2,\ \mathtt{C}\langle b\rangle.!\mathtt{B}\langle a\rangle.T_2, & T_4 = !\{\mathtt{B}\langle a\rangle.!\mathtt{C}\langle b\rangle.T_5,\ \mathtt{B}\langle c\rangle.!\mathtt{C}\langle d\rangle.T_5\} \\
\quad\ \mathtt{B}\langle c\rangle.!\mathtt{C}\langle d\rangle.T_2,\ \mathtt{C}\langle d\rangle.!\mathtt{B}\langle c\rangle.T_2\} & T_5 = ?\{\mathtt{B}\langle e\rangle.?\mathtt{C}\langle f\rangle.T_6,\ \mathtt{C}\langle f\rangle.?\mathtt{B}\langle e\rangle.T_6\} \\
T_2 = ?\{\mathtt{B}\langle e\rangle.?\mathtt{C}\langle f\rangle.T_3,\ \mathtt{C}\langle f\rangle.?\mathtt{B}\langle e\rangle.T_3\} & T_6 = !\{\mathtt{B}\langle h\rangle.?\{\mathtt{B}\langle i\rangle.!\{\mathtt{B}\langle g\rangle.\mathtt{end}\}\}\} \\
T_3 = \mu\,\mathtt{t}.!\{\mathtt{B}\langle g\rangle.\mathtt{end},\ \mathtt{B}\langle h\rangle.?\{\mathtt{B}\langle i\rangle.\mathtt{t}\}\} &
\end{array}
$$

The type $!\{\mathtt{B}\langle a\rangle.T, \mathtt{C}\langle b\rangle.T', \ldots\}$ denotes a choice between outputs $\mathtt{B}\langle a\rangle$ followed by T, $\mathtt{C}\langle b\rangle$ followed by T', etc.; dually $?\{...\}$ for input choice. For singleton choices, we can omit the curly braces, e.g. $!\mathtt{C}\langle b\rangle$. $\mu\mathtt{t}.T$ denotes a recursive type. T_1 is the endpoint type that corresponds most "directly" to the structure relevant to $\mathtt{A}$ in the graph. The parallel forks after p_0 to $\mathtt{B}$ and $\mathtt{C}$ are expanded into the sequential interleaved outputs (a, b and c, d) in each branch. This is followed in T_2 by the interleaved inputs (e, f) in the next part joining at t. Conformance prioritizes parallel outputs over inputs to prevent deadlocks (§ 3). T_3 conforms to the final part (after t) with a recursive type containing the branch by $\mathtt{A}$ to either enact the loop (g) or end the protocol (h). T_4 differs from T_1 by safely under-specifying a subset of the interleaved outputs (analogously to MPST output subtyping) in the first part, and performing only one specific trace of the recursive branch; replacing T_6 with $!\{\mathtt{B}\langle g\rangle.\mathtt{end}\}$ would also be conformant. T_1 and T_4 are each guaranteed compatible with any independently conformant $\mathtt{B}$ and $\mathtt{C}$ endpoints.

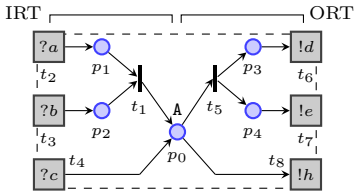
In § 3, we use conformant endpoint types to type check endpoint session processes, including channel passing and session delegations [12]. We show that safety and progress of a well-formed session net are reflected in the MPST safety and progress of a system of conformant, well-typed endpoint processes. This approach gives a natural application for our novel notion of progress in PNs.

The benefits of the session nets framework come from integrating PN and π-calculus models to support more advanced graphical MPST protocols, and to bridge the gap between high-level graphical specifications and lower-level endpoint code. To our knowledge, session nets are the first application of Petri nets to a static typing of processes featuring channel mobility. We have implemented Java APIs for validating session graph well-formedness and endpoint type conformance to demonstrate the tractability of our framework, which are available from [19]. The appendix contains additional related work, use cases [3] and full proofs.

2 Session Net Graphs

2.1 Role Structures and Session Net Graphs

We first define the *labelled Petri net graphs* that we have adapted to represent message passing protocols in the manner of multiparty session types (MPST). We then introduce *role structures*, which are labelled Petri net graphs given by a few simple construction rules. Role structures are interconnected by asynchronous *communication places* to form a complete session net graph.



$P = \{p_0, p_1, p_2, p_3, p_4\}$
$T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$
$F = \{(t_1, p_0), (t_4, p_0), (p_1, t_1), (p_2, t_1), (p_2, p_1), (t_3, p_2), (p_0, t_5), (p_0, t_8), (t_5, p_3), (t_5, p_4), (p_3, t_6), (p_4, t_7)\}$
$f = \{p_i \mapsto \mathtt{A}\}_{0 \leq i \leq 4}$
$g = \{t_2 \mapsto ?a, t_3 \mapsto ?b, t_4 \mapsto ?c, t_6 \mapsto !d, t_7 \mapsto !e, t_8 \mapsto !h\}$

Fig. 2. An example role structure

Petri net graphs We extend standard Petri net graphs with functions f and g to specify MPST roles and message labels [12]. A *labelled Petri net graph* (henceforth, *Petri net graph*) is a tuple $\boldsymbol{P} = \langle P, T, F, f, g \rangle$, where: $P = \{p_0, \ldots, p_n\}$ is a finite set of *places*; $T = \{t_0, \ldots, t_m\}$ is a finite set of *transitions*; $F \subseteq (P \times T) \cup (T \times P)$ is a set of *arcs* (the *flow relation*); $f : P \rightharpoonup R$ is a partial function which associates places to *role names* from the set $R = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \ldots\}$; and $g : T \rightharpoonup L$ is a partial injective function which associates transitions to *message labels* from the set $L = \{\dagger_1 a, \dagger_2 b, \dagger_3 c, \ldots\}$ where $\dagger = ? \mid !$ is an *I/O decoration*. Places and transitions are required to be disjoint ($P \cap T = \emptyset$) and their union, denoted by X, is required to be non-empty ($X = P \cup T \neq \emptyset$). The elements $(x, y, \ldots)$ of X are called *nodes*. The *pre-set* of $x \in X$ is $\bullet x = \{y \in X \mid (y, x) \in F\}$ and its *post-set* is $x\bullet = \{y \in X \mid (x, y) \in F\}$.

We represent places as circles and arcs as arrows between places and transitions, as in the standard graphical representation. We call *observable* the transitions in the domain of g and represent them as boxes. The other transitions are called *internal* and represented as narrow rectangles, as in Figure 1. Observable transitions are annotated according to the g labelling function: ?-decorated observables are referred to as *inputs*, and !-decorated observables as *outputs*. Places can be annotated according to the f function.

Role structures An *inbound role tree* (IRT) is a Petri net graph $\boldsymbol{P} = \langle P, T, F, f, g \rangle$ with $\mathsf{dom}(f) = P$, which forms a directed tree rooted at a place, with set of nodes X and edges F, and such that: (1) every arc is directed towards the root (the root is reachable from every node); (2) every observable transition is an input; (3) if $|X| > 1$, the set of inputs contains all and only leaves. An *outbound role tree* (ORT) is defined dually, but permits observables in non-leaf positions: (1) every arc is directed away from the root; (2) every observable transition is an output; (3) if $|X| > 1$, every leaf is an output. An IRT or ORT with $|X| = 1$ is just a single root place.

Using common terminology [3, 18], we refer to: a place in an IRT as a *merge*, and in an ORT as a *branch*; and a transition in an IRT as a *join*, and in an ORT as a *fork*. Intuitively, an IRT represents the internal synchronisations within a role after receiving control through the arrival of external messages (the input leaf nodes). An ORT represents the decisions leading to the transfer of control to other roles by dispatching external messages (the output leaf nodes). Their asymmetry reflects the I/O asymmetry of the conformance approach (see § 3).

A *role structure* consists of an IRT and an ORT, rooted at a shared place and disjoint elsewhere, for a single role. Figure 2 as a whole shows a RS for role `A` with core place p_0. We often annotate only the core place in each RS. Formally:

Definition 2.1 (Role structures). Let $\boldsymbol{P}_1 = \langle P_1, T_1, F_1, f_1, g_1 \rangle$ be an IRT and $\boldsymbol{P}_2 = \langle P_2, T_2, F_2, f_2, g_2 \rangle$ an ORT. Then $\boldsymbol{P} = \langle P_1 \cup P_2, T_1 \cup T_2, F_1 \cup F_2, f_1 \cup f_2, g_1 \cup g_2 \rangle$ is a *role structure* (RS) iff: (1) $P_1 \cap P_2 = \{p\}$ and p, called the *core place*, is the root of P_1 and P_2; (2) $T_1 \cap T_2 = \emptyset$; (3) $f_1(p_1) = f_2(p_2) = \{\mathtt{A}\}$ for all $p_1 \in P_1, p_2 \in P_2$ and some $\mathtt{A} \in R$. We use $\boldsymbol{R}_1$, $\boldsymbol{R}_2$,... to denote role structures.

Session net graphs We construct *session net graphs*, using *communication places* to compose role structures by connecting their input and output transitions.

Definition 2.2 (Session net graphs). A *session net graph* (*session graph* or SG for short) $\boldsymbol{G}$ is a Petri net graph generated by the following cases:

1. $\boldsymbol{G} = \boldsymbol{R}$ is a role structure;
2. $\boldsymbol{G} = \langle P_1 \cup P_2, T_1 \cup T_2, F_1 \cup F_2, f_1 \cup f_2, g_1 \cup g_2 \rangle$ is the union of disjoint session graphs $\boldsymbol{G}_1 = \langle P_1, T_1, F_1, f_1, g_1 \rangle$ and $\boldsymbol{G}_2 = \langle P_2, T_2, F_2, f_2, g_2 \rangle$;
3. $\boldsymbol{G} = \langle P \cup \{p\}, T, F \cup \{(t_!, p), (p, t_?)\}, f, g \rangle$ where $\langle P, T, F, f, g \rangle$ is a session graph, $p \notin P$ is a *communication place*, $t_!$ is an output and $t_?$ is an input and: (1) $\exists_{a \in L}(g(t_!) = !a \wedge g(t_?) = ?a)$; (2) $\nexists_{p' \in P \setminus \mathsf{dom}(f)}((t_!, p') \in F \vee (p', t_?) \in F)$.

Communication places represent asynchronous message dependencies between the roles of the connected RSs. Condition 3 prevents connecting any observable transition to more than one communication place ($P \setminus \mathsf{dom}(f)$ gives the set of communication places). In Figure 1, communication places p_5 and p_7 connect the leftmost `A` and `B` RSs, while p_6 and p_8 connect the leftmost `A` and `C` RSs.

The behaviour of a role in an SG protocol is given by all the RSs for that role and the message causalities with other RSs. Each RS represents a control point in the protocol where an internal decision by the role is activated by incoming messages, leading to the dispatch of subsequent messages. This decision may then be handled as an external choice distributed over multiple RSs downstream. In Figure 1, `A`'s internal choice to send g or h is handled by `B` over the two right-most `B`-RSs. Recursive protocols are also formed from the composition of RSs.

Free-choice graphs [10] are a well-known class of Petri net graphs, where complexity is limited by structurally preventing conflicts. A Petri net graph is *free-choice* if, for any arc from a place p to a transition t, either $\bullet t = \{p\}$ or $p\bullet = \{t\}$. The following states that every SG is *free-choice*.

Proposition 2.1. *If $\boldsymbol{G}$ is an SG, then $\boldsymbol{G}$ is a free-choice Petri net graph.*

2.2 Well-formedness of Session Net Graphs

Paths, cycles and diamonds Let $\boldsymbol{G} = \langle P, T, F, f, g \rangle$ and $X = P \cup T$. A node $x \in X$ is *initial* if $\bullet x = \emptyset$ and *terminal* if $x\bullet = \emptyset$. We write $Term(\boldsymbol{G})$ for the set of terminal nodes in $\boldsymbol{G}$. $F_{-x} = \{(x', x'') \mid x' \in X \setminus \{x\}, x'' \in X \setminus \{x\}, (x', x'') \in F\}$ denotes the restriction of F to $X \setminus \{x\}$. We extend this definition to a set of nodes in the natural way, where we omit set parenthesis, e.g. we write $F_{-x,y} = F_{-\{x,y\}}$. The reflexive and transitive closure of a relation $\Re$ is denoted $\Re^*$.

A *path* in $\boldsymbol{P}$ is a finite, non-empty sequence of nodes $x_0..x_n$ such that $(x_i, x_{i+1})_{0 \leq i \leq n-1} \in F$. We let $\sigma, \sigma', \ldots$ range over the set of paths augmented by

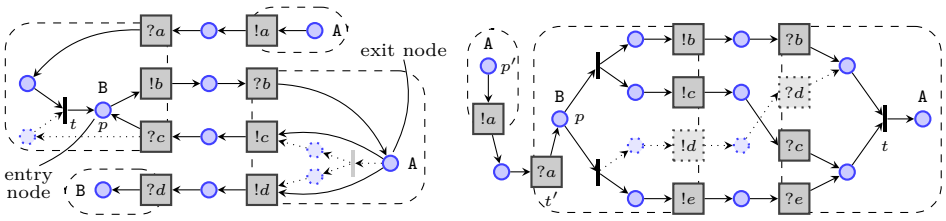


Fig. 3. Illustration of cycles and diamonds

the empty sequence ϵ; $\sigma\sigma'$ denotes the concatenation of σ and σ'. We sometimes treat σ as the set of nodes occurring in it, e.g. we write $\sigma \cup \sigma'$. We say σ is a *simple path* iff every $x \in \sigma$ occurs exactly once; σ *contains* a node x if $x \in \sigma$.

A *cycle* φ is a path $x\,x_1..x_n\,x$ where $x\,x_1..x_n$ is a simple path. A node x is: an *entry node of* φ iff there is a path σ from an initial node to x and $\sigma \cap \varphi = \{x\}$; an *exit node of* φ iff there is a path σ' from x to a terminal node and $\sigma' \cap \varphi = \{x\}$. Figure 3 (left) shows a cycle, along with its entry and exit nodes.

A *diamond* δ from *start* x to *end* y, $x \neq y$, is a pair of paths $\delta = \langle x\,\sigma_1\,y,\ x\,\sigma_2\,y\rangle$, where $\sigma_1 \cap \sigma_2 = \emptyset$, $\sigma_1 \cup \sigma_2 \neq \emptyset$ and $x, y \notin \sigma_1 \cup \sigma_2$. δ is *pre-cross-free* if for all $z' \in \sigma_1$ and $z'' \in \sigma_2$, $(z', z'') \notin F^*_{-x,y}$ or $(z'', z') \notin F^*_{-x,y}$. Informally, δ is pre-cross-free if it does not feature a pair of criss-crossing paths between its two sides. In Figure 3 (right), the diamond with start p and end t is pre-cross-free when the dotted part is ignored. Finally, δ is *cross-free* if it is pre-cross-free in the graph obtained by removing the nodes of a path, if any, from an initial node to each $z \in \bullet x$. That is, a cross-free diamond has an entry path via each $z \in \bullet x$ that does not overlap the diamond. The p–t diamond in Figure 3 (right) is cross-free due to the path from p' to t'.

The conditions for an SG to be well-formed are as follows.

Definition 2.3 (Well-formed session graph). An SG $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ is *well-formed* if it is a connected graph that respects the following conditions:

(Reachability) (R1) There is exactly one initial node: place $p_I \in P$
(R2) All terminal nodes are core places
(R3) $\forall_{x \in X}\ ((p_I, x) \in F^* \wedge \exists_{y \in Term(\boldsymbol{G})}\ ((x, y) \in F^*))$
(Labels) (L1) $\forall_{p \in P}, \forall_{t,t' \in p\bullet}\ (\{f(p') \mid (t, p') \in F^*\} = \{f(p') \mid (t', p') \in F^*\})$
(L2) $\forall_{t \in \mathsf{dom}(g)}, \exists_{p \in P \setminus \mathsf{dom}(f)}((p, t) \in F \vee (t, p) \in F)$
(Cycles) (C1) If x is an entry node for some cycle φ, $x \in P$
(C2) If x is an exit node for some cycle φ, $x \in P$
(Diamonds) (D1) If $\langle x\sigma_1 y, x\sigma_2 y\rangle$ is a diamond, then $x \in T \Rightarrow y \in T$
(D2) If $\langle x\sigma_1 y, x\sigma_2 y\rangle$ is cross-free, then $x \in P \Rightarrow y \in P$
(D3) If $\langle t\sigma_1 y, t\sigma_2 y\rangle$ is cross-free, then for all $p \in \sigma_1$ and $t' \in p\bullet$, there is a σ_1' such that $t' \in \sigma_1' y$ and $\langle t\sigma_1' y, t\sigma_2 y\rangle$ is cross-free

The first five conditions correspond to basic properties of MPST global types. (R1)–(R3) ensure that every node is reachable from the initial place and a terminal place is reachable from them. (L1) checks that the sets of roles involved in each case of a branch are equal (branch mergeability [4, 9]). (L2) ensures that the SG construction has connected every input and output to a communication place.

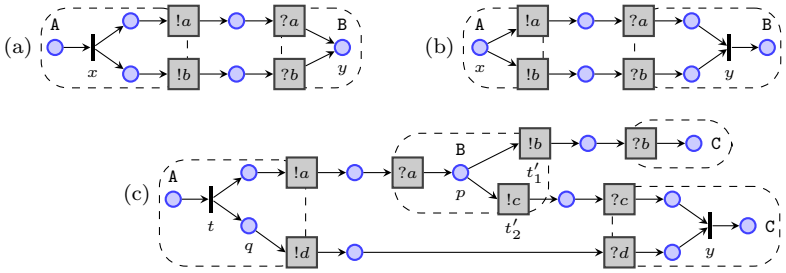


Fig. 4. Badly-formed session graphs illustrating conditions (D1)–(D3)

The remaining conditions ensure safety and progress of token dynamics, by constraining the composition of branch-merge, fork-join and recursive structures to be a realisable MPST protocol. (C1) and (C2) state that an entry or exit node of any cycle is a place. (D1) requires a diamond starting at a transition to also end at a transition. (D2) imposes a dual condition only on cross-free diamonds. (D3) checks that branches along a cross-free transition-start diamond are re-merged before the diamond ends. Cross-free diamonds represent the "minimal" diamond structures for which these latter constraints need hold. Checking these conditions on cross-free diamonds only (i.e. not all diamonds) permits a larger set of well-formed SGs, e.g. the p_0–t diamond in Figure 1 is not checked for (D2).

We illustrate conditions (C1), (C2) and (D1)–(D3) by examples. In Figure 3 (left), if the dotted structure in the top-left RS for B is added, the transition t would be the entry node of a cycle, violating (C1): the net execution from the initial place would be immediately stuck. If instead the dotted structure in the bottom-right RS for A is added, the greyed-out internal transition would be an exit node, violating (C2): the net execution would be unsafe, allowing an unbounded number of tokens to accumulate within the cycle. Figure 4 (a)–(c) give badly-formed SGs that violate conditions (D1)–(D3), respectively. In (a), the diamond opened by a fork but closed by a place is unsafe (not 1-bounded). In (b), the (cross-free) diamond opened by a branch but closed by a transition will be stuck. Note that it is not necessary to apply this condition to non-cross-free diamonds, e.g. the p_0–t diamond in Figure 1. In (c), the branch at p along the upper side of the t–y diamond will prevent the net from terminating if t'_1 is chosen by B.

Proposition 2.2. *For any SG $\boldsymbol{G}$, well-formedness is decidable.*

Deciding well-formedness conditions (R1) to (C2) is straightforward from their definitions. For (D1) and (D2), we can show that if the properties hold for any SG diamond comprised of simple paths, they hold for all general diamonds that may be derived from the "simple diamond" by performing some number of cycles along its sides. The case of (D3) is similarly decided by checking only the diamonds restricted to simple paths from the start to p and from p to the end.

2.3 Session Nets

A *Petri net* $\langle \boldsymbol{P}, M \rangle$ is a Petri net graph $\boldsymbol{P} = \langle P, T, F, f, g \rangle$ with a *marking* $M : P \to \mathbb{N}_0$. The following is standard terminology. A place $p \in P$ contains n

tokens in M, if $M(p) = n$. A transition $t \in T$ is *enabled* at M (written $\langle \boldsymbol{P}, M\rangle \xrightarrow{t}$) when $M(p) > 0$ for every $p \in \bullet t$. When t is enabled it may *fire*, yielding a new marking M' (written $\langle \boldsymbol{P}, M\rangle \xrightarrow{t} \langle \boldsymbol{P}, M'\rangle$) such that: $M'(p) = M(p) - 1$, for all $p \in (\bullet t \setminus t\bullet)$; $M'(p) = M(p)+1$, for all $p \in (t\bullet \setminus \bullet t)$; $M'(p) = M(p)$, otherwise. We may omit $\boldsymbol{P}$ if it is clear from the context. A *firing sequence* $M_0 \xrightarrow{t_1} M_1 \ldots \xrightarrow{t_n} M_n$ can also be written $\phi : \langle \boldsymbol{P}, M_0\rangle \xrightarrow{s} \langle \boldsymbol{P}, M_n\rangle$, where $s = t_1 \ldots t_n$. A marking M' is *reachable* from M in $\boldsymbol{P}$ if there is a firing sequence ϕ from $\langle \boldsymbol{P}, M\rangle$ to $\langle \boldsymbol{P}, M'\rangle$.

Definition 2.4 (Session nets). *Let $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ be a well-formed SG with initial place $p_I \in P$. A Petri net $\boldsymbol{N} = \langle \boldsymbol{G}, M\rangle$ is: 1) an* initial session net *and M is the* initial marking *for $\boldsymbol{G}$ iff $M(p_I) = 1$, and $M(p) = 0$ for all $p \in P \setminus \{p_I\}$; 2) a* session net *iff M is reachable from the initial marking M_0 for $\boldsymbol{G}$.*

Session nets satisfy the standard safety of Petri nets [10, 17], i.e. no place contains more than one token in any marking reachable from the initial marking. Formally: a Petri net $\langle \boldsymbol{P}, M\rangle$ is *safe* iff $M'(p) \leq 1$, for all p and M' reachable from M in $\boldsymbol{P}$.

Theorem 2.1 (Safety). *Every initial session net is safe.*

We want to ensure that sessions can always terminate successfully [2, 12, 20]. Standard Petri nets liveness [10, 17] asks for continuous execution in a system such that no part ever becomes redundant, which is not practical for general sessions. Deadlock-freedom instead requires that every reachable marking enables some transition, which does not ensure the progress of all session participants.

Let $\langle \boldsymbol{G}, M\rangle$ be a session net for $\boldsymbol{G} = \langle P, T, F, f, g\rangle$. The marking M is *terminal* in $\boldsymbol{G}$ just when, for all $p \in P$, $M(p) > 0$ implies $p \in Term(\boldsymbol{G})$, i.e. only terminal places contain tokens. Progress asks for some terminal marking to be reachable:

Theorem 2.2 (Progress). *Let $\boldsymbol{N} = \langle \boldsymbol{G}, M\rangle$ be a session net. Then there is a terminal marking M' which is reachable from M in $\boldsymbol{G}$.*

The proofs of decidability of well-formedness (Proposition 2.2), safety (Theorem 2.1) and progress (Theorem 2.2) are based on a conspicuous set of basic properties of diamonds and cycles in well-formed SGs (see appendix).

We sketch the critical part of the proof of progress, specifying which condition of Definition 2.3 is mostly responsible for proving each step. Given a reachable marking M and a transition t such that there are two places $p, q \in \bullet t$, where $M(p) > 0$ and $M(q) = 0$, we need to show that a marking M' is reachable from M, where t is enabled. First, we go back through the history of firings which led to adding a token into p, until we find a fork which rejoins at t via q (we use (D2)). Then we go forward in the firing history again, all the way to M. Meanwhile, we show the invariant that a token is "on its way" to q, i.e. that there is a simple path σ from a non-empty place to q (we use (D3)). Finally, we show that there exists a sequence of firings which starts from M, contains all the transitions in σ and ends by inserting a token into q (we use (C1)).

We comment on the use of other well-formedness conditions. (D1) and (C2) are fundamental for safety: (D1) prevents incorrect merging of parallel token flows

generated by a fork; (C2) rules out cycle patterns that could generate unboundedly many tokens. (R1) and (R3) allow to derive basic results upon which all properties rely. (R2), (L1) and (L2) ensure session types compatibility (§ 3 and § 3).

3 Endpoint Types and Conformance

Endpoint types represent the local view of a global protocol from the perspective of a role. This section defines *conformance* between well-formed SGs and syntactic endpoint types. Using the results of § 2, we show the key property of our framework: executing a system of independently conformant endpoints preserves conformance to the corresponding global net execution, thereby ensuring safety and progress.

Endpoint multiparty session types Syntactic endpoint types provide a more programmatic specification for implementation, to be verified by type checking (as shown in § 3) or type inference (along the line of [22]). We define their syntax and LTS with message buffers for asynchronous FIFO communication.

Endpoint types are defined as follows:

$$T ::= \quad ?\{\mathtt{r}_i\langle a_i\rangle.T_i\}_{i\in I} \mid \quad !\{\mathtt{r}_i\langle a_i\rangle.T_i\}_{i\in I} \;\mid\; \mu\,\mathtt{t}.T \;\mid\; \mathtt{t} \;\mid\; \mathsf{end}$$

Input choice ($?\{\mathtt{r}_i\langle a_i\rangle.T_i\}_{i\in I}$) is an external choice, receiving one of the I-indexed messages labelled a_i from role $\mathtt{r}_i$ ($\mathtt{A}, \mathtt{B}, \dots$). Dually, *output choice* ($!\{\mathtt{r}_i\langle a_i\rangle.T_i\}_{i\in I}$) internally chooses one of the a_i messages to send to $\mathtt{r}_i$. $\mathtt{t}$ is a recursion variable, $\mu\,\mathtt{t}.T$ is a recursive type that binds $\mathtt{t}$ in T, and end is the terminated type. We assume that all labels in types are distinct and recursive types are guarded, taking an equi-recursive view of types [2, 12]. Let R be a set of roles, then:

$$C ::= (\vec{T}, \vec{w}) \qquad \vec{T} = (T_{\mathtt{r}})_{\mathtt{r}\in R} \qquad \vec{w} = (w_{\mathtt{r}\mathtt{r}'})_{\mathtt{r}\neq\mathtt{r}'\in R} \qquad w ::= \vec{a}$$

where C denotes *configurations* and w denotes *buffers*. Let m denote the *actions* $m ::= \mathtt{r}!\mathtt{r}'\langle a\rangle \mid \mathtt{r}?\mathtt{r}'\langle a\rangle$. We write $!m$ to stand for $\mathtt{r}!\mathtt{r}'\langle a\rangle$ for some $\mathtt{r}$, $\mathtt{r}'$ and a; similarly for $?m$. The relation $T \xrightarrow{m} T'$, on endpoint types for role $\mathtt{r}$, is given by:

$$!\{\mathtt{r}'_i\langle a_i\rangle.T_i\}_{i\in I} \xrightarrow{\mathtt{r}!\mathtt{r}'_i\langle a_i\rangle} T_i \qquad ?\{\mathtt{r}'_i\langle a_i\rangle.T_i\}_{i\in I} \xrightarrow{\mathtt{r}?\mathtt{r}'_i\langle a_i\rangle} T_i \qquad \frac{T[\mu\,\mathtt{t}.T/\mathtt{t}] \xrightarrow{m} T'}{\mu\,\mathtt{t}.T \xrightarrow{m} T'}$$

We write $T \xrightarrow{m}$ iff $T \xrightarrow{m} T'$ for some T'. Lastly, $(\vec{T}, \vec{w}) \xrightarrow{\mathtt{r}\dagger\mathtt{r}'\langle a\rangle} (\vec{T}', \vec{w}')$ iff:

$$\dagger = ! \implies (T_{\mathtt{r}} \xrightarrow{\mathtt{r}!\mathtt{r}'\langle a\rangle} T'_{\mathtt{r}} \wedge (i \neq \mathtt{r} \Rightarrow T'_i = T_i) \wedge w_{\mathtt{r}\mathtt{r}'} \cdot a = w'_{\mathtt{r}\mathtt{r}'} \wedge (ij \neq \mathtt{r}\mathtt{r}' \Rightarrow w_{ij} = w'_{ij}))$$

$$\dagger = ? \implies (T_{\mathtt{r}} \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a\rangle} T'_{\mathtt{r}} \wedge (i \neq \mathtt{r} \Rightarrow T'_i = T_i) \wedge w_{\mathtt{r}'\mathtt{r}} = a \cdot w'_{\mathtt{r}'\mathtt{r}} \wedge (ij \neq \mathtt{r}'\mathtt{r} \Rightarrow w_{ij} = w'_{ij}))$$

Output by $\mathtt{r}$ to $\mathtt{r}'$ enqueues a message in the buffer. Input by $\mathtt{r}'$ consumes messages in the same order, checking that the label matches one of those expected.

Conformance Conformance replaces the usual projection found in MPST systems [12]. Similarly to safe projections and session type subtyping [8], conformance relates the local protocol behaviour of a role to the global specification. Unlike projection, it uses the global behavioural model (i.e. net dynamics) to validate each local behaviour at endpoint level.

The functions $\mathtt{local}(t)$ and $\mathtt{remote}(t)$ lookup the *local* and the *remote role* of an observable transition t, respectively. Given $\boldsymbol{G} = \langle P, T, F, f, g\rangle$, let $t \in \mathsf{dom}(g)$.

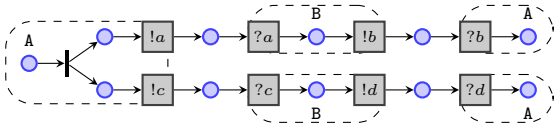


Fig. 5. Motivation for output priority in independent conformance to parallel SG flows

Then $\mathtt{local}(t) = f(p)$, for $p \in \mathsf{dom}(f)$ such that $((p,t) \in F \vee (t,p) \in F)$. Similarly, $\mathtt{remote}(t) = f(p)$, for $p \in \mathsf{dom}(f)$ such that there are $p' \notin \mathsf{dom}(f)$ and t', where either: $\{(p,t'),(t',p'),(p',t)\} \subseteq F$ if $g(t) = ?a$; or $\{(t,p'),(p',t'),(t',p)\} \subseteq F$ if $g(t) = !a$. We define the *projected LTS* on session nets for a role $\mathtt{r}$ as follows:

1. $\langle \boldsymbol{G}, M\rangle \xrightarrow{\mathtt{r}\dagger\mathtt{r}'\langle a\rangle} \langle \boldsymbol{G}, M'\rangle$ if $M \xrightarrow{t} M'$, $g(t) = \dagger a$, $\mathtt{local}(t) = \mathtt{r}$ and $\mathtt{remote}(t) = \mathtt{r}'$
2. $\langle \boldsymbol{G}, M\rangle \xrightarrow{\tau_{\mathtt{r}}} \langle \boldsymbol{G}, M'\rangle$ if $M \xrightarrow{t} M'$ and $\mathtt{local}(t) \neq \mathtt{r}$.
3. $\langle \boldsymbol{G}, M\rangle \xrightarrow{\tau} \langle \boldsymbol{G}, M'\rangle$ if $M \xrightarrow{t} M'$ and $t \notin \mathsf{dom}(g)$.

We write $\xrightarrow{\tau^*}$ for the reflexive and transitive closure of $\xrightarrow{\tau}$, and $\Rightarrow$ for the reflexive and transitive closure of $\xrightarrow{\tau} \cup \xrightarrow{\tau_{\mathtt{r}}}$. Conformance is defined as follows.

Definition 3.1 (Conformance). An endpoint type $T_{\mathtt{r}}$ *conforms to a session net* $\langle \boldsymbol{G}, M\rangle$, written $T_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M\rangle$, if the following conditions are satisfied:

1. (a) if $T_{\mathtt{r}} \xrightarrow{\mathtt{r}!\mathtt{r}'\langle a\rangle} T'_{\mathtt{r}}$, then $\langle \boldsymbol{G}, M\rangle \Rightarrow \xrightarrow{\mathtt{r}!\mathtt{r}'\langle a\rangle} \langle \boldsymbol{G}, M'\rangle$ and $T'_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M'\rangle$
   (b) if $T_{\mathtt{r}} \xrightarrow{?m}$, then $\langle \boldsymbol{G}, M\rangle \Rightarrow \xrightarrow{?m'}$ for some $?m'$
2. (a) if $\langle \boldsymbol{G}, M\rangle \Rightarrow \xrightarrow{\mathtt{r}!\mathtt{r}'\langle a\rangle}$, then $T_{\mathtt{r}} \xrightarrow{!m}$ for some m
   (b) if $\langle \boldsymbol{G}, M\rangle \Rightarrow \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a\rangle} \langle \boldsymbol{G}, M'\rangle$, then:
      – $T_{\mathtt{r}} \xrightarrow{\vec{m}} \xrightarrow{?m}$, for some $?m$ and sequence of output actions $\vec{m}$
      – if $T_{\mathtt{r}} \xrightarrow{?m}$ for some $?m$, then $T_{\mathtt{r}} \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a\rangle} T'_{\mathtt{r}}$ and $T'_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M'\rangle$
3. if $\langle \boldsymbol{G}, M\rangle \xrightarrow{\tau_{\mathtt{r}}} \langle \boldsymbol{G}, M'\rangle$, then $T_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M'\rangle$

$T_{\mathtt{r}} \asymp \boldsymbol{G}$ ($T_{\mathtt{r}}$ *conforms to* $\boldsymbol{G}$) if $T_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M_0\rangle$ where M_0 is the initial marking.

The asymmetry between cases 1 and 2 is due to choice subtyping [8], and the omission of parallel endpoint types. In 1(a), every endpoint output must be simulated by the session net. In 2(a), an endpoint only has to perform *some* output when the net outputs. Thus endpoint outputs may safely underspecify the global model. Dually, endpoint inputs may be overspecified. In 2(b) and 1(b), the endpoint simulates every input by the net, but not vice versa. In 2(b), we allow the endpoint to output before simulating an input: this is sound because the net can do the same outputs without disabling the original input. Note that the subtyping [8] is included in the conformance: if $T_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M\rangle$ and $T'_{\mathtt{r}} \leqslant T_{\mathtt{r}}$ where $\leqslant$ is defined as in [8, Definition 8], then $T'_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M\rangle$ (Lemma J.1).

Conformant endpoint types for the SG in Figure 1 were explained in § 1. Figure 5 shows a SG between roles $\mathtt{A}$ and $\mathtt{B}$, and endpoint types $T_{\mathtt{A}}^{bad}$ and $T_{\mathtt{B}}^{bad}$ for $\mathtt{A}$ and $\mathtt{B}$, respectively (using the abbreviated notation described in § 1). Note that these types do not independently conform to the SG: $T_{\mathtt{A}}^{bad}$ refines the global

$$
\begin{array}{rcll}
P & ::= & \overline{u}[\mathtt{r}_1,..,\mathtt{r}_n](c).P & \text{Request} \\
& | & u[\mathtt{r}](c).P & \text{Accept} \\
& | & c!\,\mathtt{r} : l\langle v\rangle; P & \text{Select} \\
& | & c?\{\mathtt{r}_i : l_i(z_i).P_i\}_{i\in I} & \text{Branch} \\
& | & P \mid Q \;\mid 0 & \text{Parallel, Nil} \\
& | & \mu X.P \;\mid X & \text{Recursion} \\
& | & (\boldsymbol{\nu} a)P \;\mid (\boldsymbol{\nu} s)P & \text{Hiding} \\
& | & s[\mathtt{r},\mathtt{r}'] : h & \text{Queue}
\end{array}
\qquad
\begin{array}{rcll}
h & ::= & \epsilon \mid h \cdot l\langle v\rangle \mid h \cdot s[\mathtt{r}] & \\
v & ::= & a \mid s[\mathtt{r}] \mid x & \text{(values)} \\
u & ::= & a \mid x & \text{(identifiers)} \\
c & ::= & x \mid s[\mathtt{r}] & \text{(sessions)} \\
s, s', ... & & & \text{(session names)} \\
a, b, ... & & & \text{(shared names)} \\
x, y, z, ... & & & \text{(variables)}
\end{array}
$$

Fig. 6. Syntax of processes

protocol by forcing a process to wait for an acknowledgement to a (message b), before sending c; similarly, $T_{\mathtt{B}}^{bad}$ mandates to wait for both a and c before doing any output. When composed together, they get stuck in a deadlock. Conformance is designed to prioritise outputs over inputs, thus ruling out incorrect protocols as $T_{\mathtt{A}}^{bad}$ and $T_{\mathtt{B}}^{bad}$. If output priority was to be relaxed, both $T_{\mathtt{A}}^{bad}$ and $T_{\mathtt{B}}^{bad}$ would be conformant and deadlocks would not be prevented.

Weak transition sequences of a net are finite; hence we have:

Proposition 3.1. *For any endpoint type $T_{\mathtt{r}}$ and SG $\boldsymbol{G}$, conformance is decidable.*

Theorem 3.1 (Soundness). *Let $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ have initial marking M_0 and f have range R. Let $C_0 = (\vec{T_0}, \vec{\epsilon})$ be an* initial *configuration such that $T_{0\mathtt{r}} \asymp \langle \boldsymbol{G}, M_0\rangle$, for all $\mathtt{r} \in R$. Let also $C_0 \xrightarrow{m_1} C_1 \ldots \xrightarrow{m_n} C_n$ be such that $C_i = (\vec{T_i}, \vec{w_i})$, for all $i \in \{1, \ldots, n\}$. Then $\langle \boldsymbol{G}, M_0\rangle \xrightarrow{\tau^*}\xrightarrow{m_1} \langle \boldsymbol{G}, M_1\rangle \ldots \xrightarrow{\tau^*}\xrightarrow{m_n} \langle \boldsymbol{G}, M_n\rangle$, for some $M_1 \ldots M_n$; such that $T_{i\mathtt{r}} \asymp \langle \boldsymbol{G}, M_i\rangle$, for all $i \in \{1, \ldots, n\}$ and $\mathtt{r} \in R$.*

We now define the safety properties of a configuration C, following those in communicating automata [9, § 3]. We say C is terminal if $C = (\vec{\mathsf{end}}, \vec{\epsilon})$.

1. C is a *deadlock configuration* if $\vec{w} = \vec{\epsilon}$, while C is not terminal and no $T_{\mathtt{r}}$ is an output type, i.e. some types are blocked, waiting for messages.
2. C is an *orphan message configuration* if all $T_{\mathtt{r}} \in \vec{T}$ are end but $\vec{w} \neq \emptyset$, i.e. there is at least an orphan message in a buffer.
3. C is an *unspecified reception configuration* if there is $\mathtt{r} \in R$ such that $T_{\mathtt{r}}$ is an input and, for all $\mathtt{r}' \in R$ and a, $T_{\mathtt{r}} \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a\rangle} T'_{\mathtt{r}}$ implies that $|w_{\mathtt{r}'\mathtt{r}}| > 0$ and $w_{\mathtt{r}'\mathtt{r}} \neq a \cdot w$, i.e $T_{\mathtt{r}}$ is prevented from receiving any message from buffer $\mathtt{r}'\mathtt{r}$.

We say C is *deadlock-free* (resp. *orphan message-free*, *reception error-free*) if no C' such that $C \xrightarrow{\vec{m}} C'$ is a deadlock (resp. orphan message, unspecified reception) configuration. C is *safe* if it is deadlock-free, orphan-free and reception error-free.

Theorem 3.2 (Safety and Progress). *Let $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ be a well-formed SG, where the range of f is R. Let $C_0 = (\vec{T_0}, \vec{\epsilon})$ be an initial configuration such that $T_{0\mathtt{r}} \asymp \boldsymbol{G}$, for all $\mathtt{r} \in R$. Then* (1) *C_0 is safe; and* (2) *for all C such that $C_0 \xrightarrow{\vec{m}} C$, either C is terminal or $C \xrightarrow{m'} C'$, for some action m'.*

$$[\textsc{Req}]\frac{T_{\mathtt{r}_1} \asymp \boldsymbol{G} = \langle P,T,F,f,g\rangle \quad \Gamma \vdash u : \boldsymbol{G} \qquad \mathtt{range}(f) = \{\mathtt{r}_1,..,\mathtt{r}_n\} \quad \Gamma \vdash Q \triangleright \Delta, x : T_{\mathtt{r}_1}}{\Gamma \vdash \overline{u}[\mathtt{r}_1,..,\mathtt{r}_n](x).Q \triangleright \Delta} \qquad [\textsc{Acc}]\frac{T_{\mathtt{r}_i} \asymp \boldsymbol{G} \quad \Gamma \vdash u : \boldsymbol{G} \qquad \Gamma \vdash Q \triangleright \Delta, x : T_{\mathtt{r}_i} \quad i \neq 1}{\Gamma \vdash u[\mathtt{r}_i](x).Q \triangleright \Delta}$$

$$[\textsc{Sel}]\frac{j \in I \quad \Gamma \vdash P \triangleright \Delta, c : T_j \quad \Gamma \vdash u : \boldsymbol{G}_j}{\Gamma \vdash c!\,\mathtt{r}_j : l_j\langle u\rangle; P \triangleright \Delta, c :!\{\mathtt{r}_i\langle l_i\langle \boldsymbol{G}_i\rangle\rangle.T_i\}_{i\in I}}$$

$$[\textsc{Bra}]\frac{\forall i \in I \quad \Gamma, z_i : \boldsymbol{G}_i \vdash P_i \triangleright \Delta, c : T_i}{\Gamma \vdash c?\{\mathtt{r}_i : l_i(z_i).P_i\}_{i\in I} \triangleright \Delta, c :?\{\mathtt{r}_i\langle l_i\langle \boldsymbol{G}_i\rangle\rangle.T_i\}_{i\in I}}$$

$$[\textsc{SSel}]\frac{j \in I \quad \Gamma \vdash P \triangleright \Delta, c : T_j}{\Gamma \vdash c!\,\mathtt{r}_j : l_j\langle c'\rangle; P \triangleright \Delta, c :!\{\mathtt{r}_i\langle l_i\langle T'_i\rangle\rangle.T_i\}_{i\in I}, c' : T'_j}$$

$$[\textsc{SBra}]\frac{\forall i \in I \quad \Gamma \vdash P_i \triangleright \Delta, c : T_i, z_i : T'_i}{\Gamma \vdash c?\{\mathtt{r}_i : l_i(z_i).P_i\}_{i\in I} \triangleright \Delta, c :?\{\mathtt{r}_i\langle l_i\langle T'_i\rangle\rangle.T_i\}_{i\in I}}$$

Fig. 7. Process typing for conformant endpoints

4 Multiparty Asynchronous Session Calculus

Safety and progress are reflected from session graphs onto processes through type conformance. The syntax (Figure 6) is extended from [2], allowing communication with different roles within a single branch.It supports channel mobility and session delegation (i.e. passing and hiding shared/session channels). We summarise the semantics adapted from [2]. ⌊Link⌋ creates a new session s with bidirectional queues, where $\mathsf{fn}(P)$ is the set of free names of P; ⌊Sel⌋ enqueues and ⌊Bra⌋ dequeues a message. Other rules give the closure under $|$, ν and structural equivalence $\equiv$ (including $(\boldsymbol{\nu} s)(s[\mathtt{r}_1,\mathtt{r}'_1] : \epsilon \mid .. \mid s[\mathtt{r}_n,\mathtt{r}'_n] : \epsilon) \equiv 0$).

$$\lfloor\textsc{Link}\rfloor \quad \overline{a}[\mathtt{r}_1,..,\mathtt{r}_n](x).P_1 \mid a[\mathtt{r}_2](x).P_2 \mid \cdots \mid a[\mathtt{r}_n](x).P_n \longrightarrow (\boldsymbol{\nu} s)(\Pi_{i\in\{1,..,n\}}(P_i[s[\mathtt{r}_i]/x] \mid \Pi_{j\in\{1,..,n\}\setminus i} s[\mathtt{r}_i,\mathtt{r}_j] : \epsilon)) \quad s \notin \mathsf{fn}(P_i)$$

$$\lfloor\textsc{Sel}\rfloor \quad s[\mathtt{r}]!\,\mathtt{r}' : l\langle v\rangle; P \mid s[\mathtt{r},\mathtt{r}'] : h \longrightarrow P \mid s[\mathtt{r},\mathtt{r}'] : h \cdot l\langle v\rangle$$

$$\lfloor\textsc{Bra}\rfloor \quad s[\mathtt{r}]?\{\mathtt{r}'_i : l_i(z_i).P_i\}_{i\in J} \mid s[\mathtt{r}'_j,\mathtt{r}] : l_j\langle v\rangle \cdot h \longrightarrow P_j[v/z_j] \mid s[\mathtt{r}'_j,\mathtt{r}] : h$$

Conformance replaces the usual endpoint type projection [12]. Type environments use well-formed SGs $\boldsymbol{G}$ and endpoint types T from the previous sections:

$$\Gamma \ ::= \ \emptyset \ \mid \ \Gamma \cdot u{:}\boldsymbol{G} \ \mid \ \Gamma \cdot X{:}\Delta \qquad\qquad \Delta \ ::= \ \emptyset \ \mid \ \Delta \cdot c{:}T$$

SG/endpoint type messages are injectively mapped to pairs of process labels $l_1, l_2, \ldots$ and $\boldsymbol{G}$ or T, e.g. $?\{\mathtt{r}_i\langle l_i\langle S_i\rangle\rangle.T_i\}_{i\in I}$ where each S is either $\boldsymbol{G}$ (shared channel passing) or T (session delegation). $X : \Delta$ types a recursive process. $\Gamma \vdash P \triangleright \Delta$ is a typing judgement.

Figure 7 lists the key rules, adapted from [2], for typing conformant endpoint processes in the session net setting; the omitted rules are as in [2]. Rule [Req] types a session initiation request by checking that the endpoint type for the session body conforms to the $\boldsymbol{G}$ associated to the shared channel for role $\mathtt{r}_1$; [Acc] types an initiation accept in the dual manner. Rules [Sel] and [Bra] type selection and branching with shared channel passing (i.e. passing SG-typed messages). Rules [SSel] and [SBra] similarly type selection and branching with session delegation (i.e. linear communication of endpoint-typed messages).

Processes typed by endpoint types T_4, T_5 and T_6 in § 1 are given as follows:

$P_4 = \overline{a}[\mathtt{r}_\mathtt{A},\mathtt{r}_\mathtt{B},\mathtt{r}_\mathtt{C}](x).\mathsf{if}\ v\ \mathsf{then}\ x!\,\mathtt{r}_\mathtt{B}{:}a; x!\,\mathtt{r}_\mathtt{C}{:}b; P_5\ \mathsf{else}\ x!\,\mathtt{r}_\mathtt{B}{:}c; x!\,\mathtt{r}_\mathtt{C}{:}d; P_5$
$P_5 = x?\{\mathtt{r}_\mathtt{B}{:}e.x?\{\mathtt{r}_\mathtt{C}{:}f.P_6\}, \mathtt{r}_\mathtt{C}{:}f.x?\{\mathtt{r}_\mathtt{B}{:}e.P_6\}\} \qquad P_6 \ = \ x!\,\mathtt{r}_\mathtt{B}{:}h.x?\{\mathtt{r}_\mathtt{B}{:}i.x!\,\mathtt{r}_\mathtt{B}{:}g.0\}$

where we use the if statement which is encodable in the current syntax [15, § 2]. Without explicit subsumption typing rules, conformance still enables the typing of processes with branch/select and recursive subtype behaviours [8] and permutation of selections [16], via parallel expansion. By Theorem 3.1, we have the following subject reduction theorem, from which the safety properties for processes are derived as a corollary [12].

Theorem 4.1. *Suppose* $\Gamma \vdash P \triangleright \emptyset$ *and* $P \longrightarrow^* P'$. *Then* $\Gamma \vdash P' \triangleright \emptyset$.

Session net progress (Theorem 2.2) corresponds to the following progress property for processes within a single session [12] (a session net, as any individual global type, models a single protocol). We say $P_0 = \overline{a}[\mathtt{r}_1, .., \mathtt{r}_n](x).P_1 \mid a[\mathtt{r}_2](x).P_2 \mid \cdots \mid a[\mathtt{r}_n](x).P_n$ is *simple* if $a : \boldsymbol{G} \vdash P_0 \triangleright \emptyset$, P_i does not contain session delegation, accept, request and hiding, and $\boldsymbol{G} = \langle P, T, F, f, g \rangle$ where $\mathtt{range}(f) = \{\mathtt{r}_1, .., \mathtt{r}_n\}$.

Theorem 4.2 (Progress). *Let* $a : \boldsymbol{G} \vdash P_0 \triangleright \emptyset$ *and let* P_0 *be simple. Then for all* P *such that* $P_0 \longrightarrow^* P$, *either* $P \equiv 0$ *or* $P \longrightarrow P'$, *for some* P'.

Thus safety and progress of a well-formed net ensure those of the conforming, well-typed processes. Progress across separate sessions can be obtained by using advanced typing systems, e.g. [2], at the top of the typing systems in Figure 7.

5 Implementation and Related Work

Implementation We have implemented Java APIs for validating session graph well-formedness and endpoint type conformance to demonstrate the tractability of our framework. The code and implementations of all the examples in this paper (including Appendix A) are available at [19]. We plan to integrate this framework into an extension of Session Java [13], using these well-formedness and conformance APIs to extend the type system following § 3.

Related Work Workflow nets [20] (WFNs) are a class of Petri nets originally introduced to describe the operation of business processes. A WFN is an abstraction of a global system on which Petri net techniques are used to verify properties such as dead-lock freedom and proper termination. Session nets differ firstly by specifying multiparty role and message details that WFNs are not concerned with. A sound WFN is a good single, self-contained system, whereas a well-formed session net further ensures that the global protocol, given by the configuration of roles and messages on the structure of the net, is safely realisable as a set of independent, distributed endpoints. Secondly, as an MPST framework, session nets bridge from the global graph to syntactic endpoint specifications (via conformance), that are then used to type-check endpoint code.

Open WF-nets (oWFNs) [14] are an endpoint-oriented adaptation of WF-nets to distributed systems, that starts from constructing a separate net for each endpoint. In contrast, session nets start from the global-oriented SG model of a protocol against which each endpoint is checked for conformance. In oWFNs, the final system properties depend on the specific endpoint composition (effectively treating the complete system as a standalone WF-net), whereas in session nets,

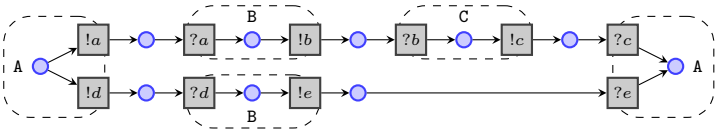


Fig. 8. A badly-formed SG due to multiparty role conditions (Def. 2.3 (L1))

any endpoints that are independently conformant to an SG are guaranteed to give a good composition. Like basic WFNs, oWFNs do not explicitly specify or validate multiparty protocol details.

Although Petri nets classes such as WFNs can be interpreted in a communications setting (e.g. in [21], the validation of a sequence of I/O action sequences is subsumed under the general task of accepting traces of fired transitions), they do not explicitly describe communication protocols. The multiparty protocol information captured by an SG and their associated well-formedness is crucial in the design of session nets, allowing us to validate the safe decomposition of the global system into distributed endpoints. Without these concerns, it is not necessary to consider as many structural constraints for WFNs as for well-formed SGs. (A basic WFN requires only (1) one initial and one terminal place, and (2) that any transition is contained in a path from the initial to the terminal place; an SG with a single terminal node is thus a WFN.)

As an example, Figure 8 shows a SG whose underlying Petri net satisfies safety and progress, but not the conditions on role labelling (specifically, Def. 2.3 (L1)). This global protocol cannot be safely realised between the distributed endpoints at the implementation level. If an `A` endpoint chooses to send d in an instance of this protocol, the `C` endpoint will not receive any message. However, this means `C` cannot locally determine whether `A` has indeed selected $!d$, or whether `A` actually selected $!a$ and `C` should wait (indefinitely) to do $?b$. A simple way to amend this SG is to ensure that `C` is also present along the lower branch (not necessarily in the same order), so that the initial internal choice by `A` is explicitly communicated to `C` in all eventualities. Other cases of incoherent message labelling, but otherwise safe in terms of the underlying Petri net, are similarly ruled out by well-formedness, e.g. race conditions in parallel protocol flows.

In [21], WF-nets are used to implement tools for checking the conformance of an executed process to a BPEL specification. Their conformance checking, however, is done at run-time and is used to verify the execution trace of a process, via e.g. a logging service or runtime monitor. Our notion of conformance is different, as it is used to statically check the local correctness of each endpoint type by relating them all to an agreed SG. A well-typed system of endpoint processes is guaranteed to behave safely for all executions.

Session nets, as in [2, 4, 9, 12] and other type structures for Web services (e.g. [1, 5, 11]), abstract from specific data types so that data typing can be integrated orthogonally. Recently there have been several works to bridge communicating automata with choreographies or session types [1, 9]. The main focus of [1, 4, 9] are projectability conditions for more general forms of global specifications. The unit of their specifications is an input-output relation between two roles (i.e. $\mathtt{A} \to \mathtt{B}$),

whereas a main new feature of session nets is the explicit representation of the internal decision structures of participants to produce outputs in response to inputs. This enables more flexible well-formed global types than those in [1, 4, 9]. None of these works proposed conformance as we have developed for session nets.

References

1. S. Basu, T. Bultan, and M. Ouederni. Deciding choreography realizability. In *POPL*, pages 191–202. ACM, 2012.
2. L. Bettini, M. Coppo, L. D'Antoni, M. D. Luca, M. Dezani-Ciancaglini, and N. Yoshida. Global progress in dynamically interleaved multiparty sessions. In *CONCUR*, volume 5201 of *LNCS*, pages 418–433. Springer, 2008.
3. Business Process Model and Notation. `http://www.bpmn.org`.
4. G. Castagna, M. Dezani-Ciancaglini, and L. Padovani. On global types and multi-party sessions. *Logical Methods in Computer Science*, 8(1), 2012.
5. G. Castagna, N. Gesbert, and L. Padovani. A theory of contracts for web services. In *POPL*, pages 261–272. ACM, 2008.
6. T. Chen and K. Honda. Specifying stateful asynchronous properties for distributed programs. In *CONCUR*, volume 7454 of *LNCS*, pages 209–224. Springer, 2012.
7. CPN Homepage. `http://cpntools.org/`.
8. R. Demangeon and K. Honda. Full abstraction in a subtyped pi-calculus with linear types. In *CONCUR*, volume 6901 of *LNCS*, pages 280–296, 2011.
9. P.-M. Deniélou and N. Yoshida. Multiparty session types meet communicating automata. In *ESOP*, volume 7211 of *LNCS*, pages 194–213, 2012.
10. J. Desel and J. Esparza. *Free Choice Petri Nets (Cambridge Tracts in Theoretical Computer Science)*. Cambridge University Press, 1995.
11. S. Hallé, T. Bultan, G. Hughes, M. Alkhalaf, and R. Villemaire. Runtime verification of web service interface contracts. *Computer*, 43(3):59–66, Mar. 2010.
12. K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *POPL*, pages 273–284. ACM, 2008.
13. R. Hu, N. Yoshida, and K. Honda. Session-Based Distributed Programming in Java. In *ECOOP'08*, volume 5142 of *LNCS*, pages 516–541. Springer, 2008.
14. N. Lohmann, P. Massuthe, C. Stahl, and D. Weinberg. Analyzing interacting bpel processes. In *Business Process Management*, volume 4102 of *LNCS*, pages 17–32. Springer, 2006.
15. D. Mostrous and V. Vasconcelos. Affine Sessions. In *Coordination*, LNCS, 2014.
16. D. Mostrous, N. Yoshida, and K. Honda. Global principal typing in partially commutative asynchronous sessions. In *ESOP*, volume 5502 of *LNCS*, pages 316–332. Springer, 2009.
17. T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
18. OMG. Unified Modelling Language, Version 2.0, 2004.
19. Java APIs for session nets. `http://www.doc.ic.ac.uk/~rhu/session_nets.html`.
20. W. M. P. van der Aalst. Verification of workflow nets. In *ICATPN*, volume 1248 of *LNCS*, pages 407–426. Springer, 1997.
21. W. M. P. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and H. M. W. Verbeek. Choreography conformance checking: An approach based on bpel and petri nets. In *The Role of Business Processes in Service Oriented Architectures*. IBFI, 2006.
22. V. T. Vasconcelos and K. Honda. Principal typing scheme for polyadic π-calculus. In *Proc. CONCUR'93*, 1993.
23. Workflow Patterns homepage. `http://www.workflowpatterns.com/`.

Appendix Contents

A Appendix: Additional Examples from Use Cases

This section gives two examples that demonstrate how session nets can be applied to more realistic and concrete multiparty application protocols. The first example is a fork-join extension of the Buyer-Seller Web services examples found in [2, 12]. The second example is a choreography from the OMG Business Process Model and Notation (BPMN) specification [3]. The source code for constructing these session graphs and validating their well-formedness using our Java APIs is available at [19].

Multiparty Buyer-Seller with fork-join Figure 9 illustrates the application protocol as a BPMN choreography. The `Buyer` (`B`) makes a combined order request, in this example for two items, to the `Seller` (`S`). In BPMN, the light background participant in each interaction box is the interaction initiator, dark participant the non-initiating participant, white envelope the message from the initiator to non-initiatior, and optional shaded envelope an acknowledgement message. The diamonds containing a + symbol are parallel gateways (used to denote both

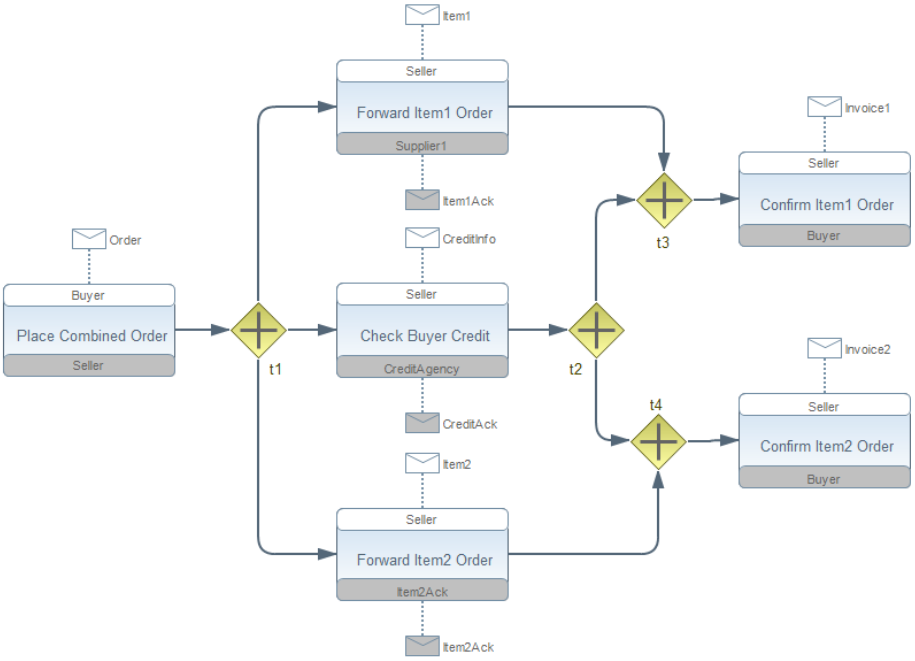


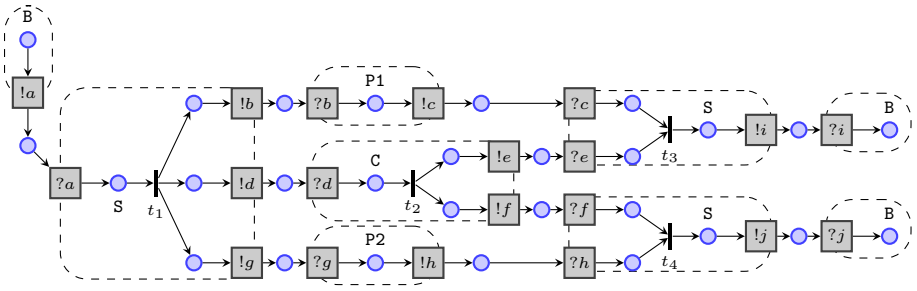
Fig. 9. BPMN choreography for a Buyer-Seller application featuring multiparty fork-join

forks and joins). After receiving the order, `S` forks (t_1) the protocol into three threads. The top and bottom threads forward the order for each item to the two different `Supplier`s (`P1` and `P2`), who acknowledge the order. In the middle thread, `S` performs a credit check on the `B` via `CreditAgency` (`C`)). Taking advantage of the graphical representation, the protocol is specified so that the join (t_3) of the top and middle threads is independent of the join (t_4) of the middle and bottom threads. As long as the `C` has acknowledged the credit check, this allows the first item order to be confirmed independently of the second (i.e. concurrently).

Figure 10 gives the session net representation of this protocol (with the map from labels to the application data types). This multiparty fork-join pattern cannot be represented using the global MPST type syntax of [2, 4, 9, 12]. Intuitively, the tree-structured syntax in these works would require either all three of the middle threads to be joined before sending either invoice message (i.e. a second parallel constructor for the two invoice threads is syntactically sequenced after the preceding three threads), or else one pair of middle threads would be joined before the other (i.e. a second parallel is syntactically nested within the first).

BPMN Logistics use case Figure 11 reproduces the choreography diagram for the "Logistics" use case from the BPMN 2.0 specification document[1] (§ 11.1). Figure 12 gives the well-formed session net representation. In addition to branching and looping (implicit in the "loop" marked tasks), this protocol features a

[1] `http://www.omg.org/spec/BPMN/2.0/PDF/`



$g = \{a \mapsto \texttt{Order}, b \mapsto \texttt{Item1}, c \mapsto \texttt{Item1Ack}, d \mapsto \texttt{CreditInfo}, e \mapsto \texttt{CreditAck1},$
$f \mapsto \texttt{CreditAck2}, g \mapsto \texttt{Item2}, h \mapsto \texttt{Item2Ack}, i \mapsto \texttt{Invoice1}, j \mapsto \texttt{Invoice2}\}$

Fig. 10. The session net representation of the protocol in Figure 9

fork-join segment in which the `Supplier-Shipper` and `Supplier-Consignee` threads loop independently to perform some number of repeated message exchanges before synchronising at the join. Note that the asymmetry between `Shipper` (`H`) and `Consignee` (`C`) threads prior to the join is due to the fact that, unlike `C`, `H` is not involved in the continuation of the protocol following the join. `Supplier` (`S`) informs `H` when the g–f (i.e. `Provide`/`Deliver Item`) loop is ended by message u, whereas it is sufficient for `C` to be informed via the message q (`Confirmation of Delivery Schedule`) later on in the protocol. For this example, we have chosen to model the `Update PO and Delivery Schedule` synchronisation at the BPMN join using two messages (h and l) to join at `Retailer` (`R`), but alternatively we could extend the graph to have `S` first perform an internal join and send a single message to `R`. The latter part of the protocol, although represented in the BPMN as a linear task sequence, involves a second fork-join pattern (the parallel behaviour is implicit in the configuration of roles along the sequence), which the session net makes explicit.

B Appendix: Additional Comments on Related Work

This section gives some additional comments on the related work discussed in § 5.

The standard notion of liveness for Petri nets [10] states that from any marking reachable from the initial marking, there is a firing sequence that will enable any transition in the net. Liveness is thus only directly suited to modelling continuously running systems, often too strong a property for communication sessions, which may feature mutually exclusive choices and eventually terminate. On the other hand, deadlock-freedom in Petri nets states that every reachable marking enables some transition; this property is too weak for modelling multiparty sessions, where progress is required of all participants.

Session nets and WFNs share the above motivations for developing more practical progress properties. Session net progress (shown for all well-formed SGs) corresponds to the proper termination conditions for WFN soundness (e.g. [20], Def. 7(i),(ii)) generalised to multiple terminal places. Soundness of a free-choice WFN can be verified in polynomial time, by verifying liveness of

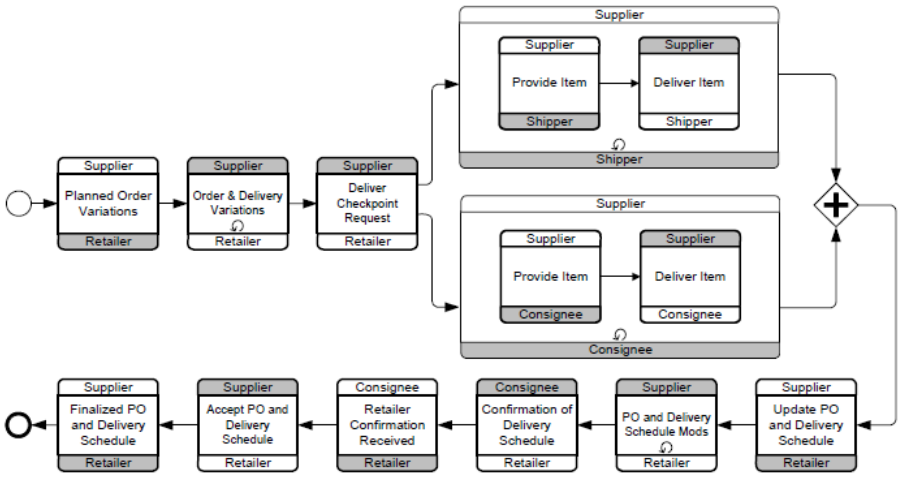
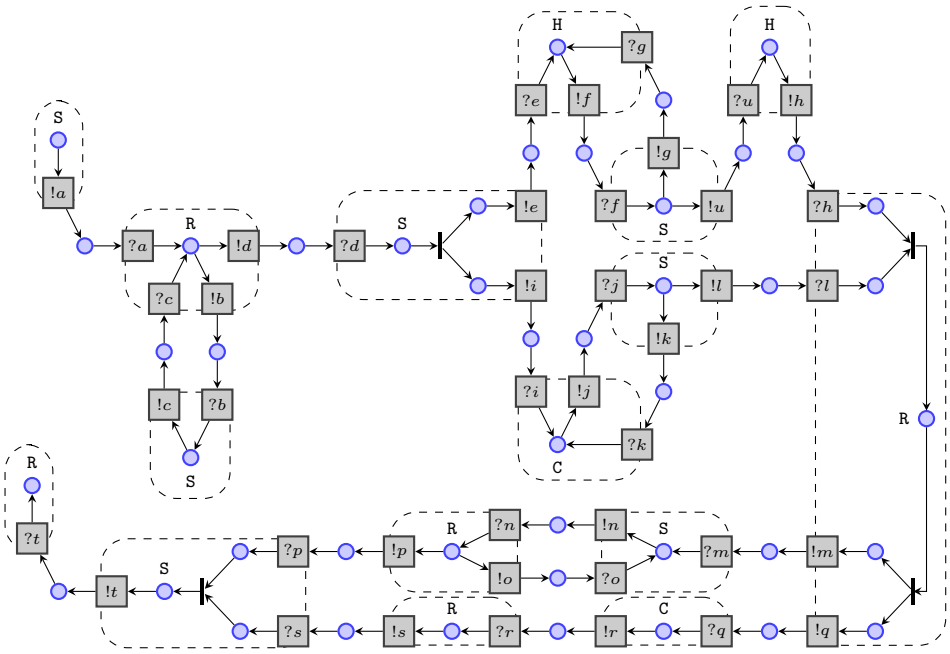


Fig. 11. "Logistics" use case from the BPMN 2.0 specification[1] (Figure 11.4, page 319)



$g = \{a \mapsto \texttt{Planned}, b \mapsto \texttt{Order}, c \mapsto \texttt{OrderEnd}, d \mapsto \texttt{Checkpoint}, e \mapsto \texttt{Provide},$
$f \mapsto \texttt{Deliver}, g \mapsto \texttt{Provide}, h \mapsto \texttt{Update}_1, i \mapsto \texttt{Provide}, j \mapsto \texttt{Deliver}, k \mapsto \texttt{Provide},$
$l \mapsto \texttt{Update}_2, m \mapsto \texttt{PO}, n \mapsto \texttt{POAck}, o \mapsto \texttt{PO}, p \mapsto \texttt{Accept}_1, q \mapsto \texttt{Confirmation},$
$r \mapsto \texttt{Retailer}, s \mapsto \texttt{Accept}_2, t \mapsto \texttt{Finalized}, u \mapsto \texttt{ProvideEnd}\}$

Fig. 12. The session net representation of the "Logistics" use case in Figure 11

a lightly extended version of the net [20]. This technique can be extended to SGs by transforming multiple terminal places to a single one, but we note the transformation may be non-trivial in the general case. Appendix H shows that the no dead transitions property of sound WFNs also holds for well-formed SGs (Theorem H.1). Regarding safety, sound WFNs are bounded, while well-formed SGs are more strictly 1-bounded. The stronger latter property is related to the preservation of linearity of communications [12] (e.g. precluding parallel message races), which is essential to guarantee session type safety.

Some of the well-formedness conditions of Definition 2.3 are naturally similar to certain characterisations of sound WFNs: e.g. [21] requires *all* diamonds to have matching splits and joins. SG well-formedness is more general, e.g. it imposes the place-to-place (branch-merge) requirement on cross-free diamonds only (permitting diamonds such as the p_0–t diamond in Figure 1). We believe our well-formed SG characterisation can be adapted to provide intuitive user feedback in MPST tool implementations based on session nets, e.g. to report why a global type/graph is badly-formed and why it would not be a valid MPST protocol. More abstract net characterisations, e.g. based on S-coverability [21] and linear dependencies [10], may not provide as direct guidance in terms of protocol design and engineering.

Session nets have so far focused on a programming methodology that steps from the graphical global specification to syntactic endpoint specifications. A possibility for integrating session nets and WFNs would be to project (or relate by conformance) global session nets to WFN endpoints extended with asynchronous I/O actions. Adapting the projection of synthesised Petri nets into communication automata [20] may offer one approach.

Additional References

20. E. Badouel, B. Caillaud, and P. Darondeau. Distributing finite automata through petri net synthesis. *Formal Aspects of Computing*, 13(6):447–470, 2002.
21. W. M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

C Appendix: Structures in Petri net Graphs

In this section we present some general results on structures that occur in Petri net graphs of all kinds.

C.1 Simple Paths

We define the function $\mathsf{simple}(-)$, which takes a path, removes the cycles and returns a simple path. Then we show that it satisfies the expected properties.

For any path σ, we define $\mathsf{simple}(\sigma)$ as follows:

- $\mathsf{simple}(\epsilon) = \epsilon$;
- if $\sigma = x \cdot \sigma'$ and $x \notin \sigma'$, then $\mathsf{simple}(\sigma) = x \cdot (\mathsf{simple}(\sigma'))$;
- if $\sigma = x \cdot \sigma' \cdot x \cdot \sigma''$ and $x \notin \sigma''$, then $\mathsf{simple}(\sigma) = x \cdot (\mathsf{simple}(\sigma''))$.

The first result shows that the function $\mathsf{simple}(-)$ indeed returns a simple path, with the same start and end nodes as the original path.

Lemma C.1. *Let σ be a path from x to y in a Petri net graph $\boldsymbol{P}$. Then $\mathsf{simple}(\sigma)$ is a simple path from x to y.*

Proof. We do the proof by induction on the length of σ. Since σ is an actual path, it cannot be ϵ. Then, by definition of $\mathsf{simple}(-)$, there are σ', σ'' and x such that $\sigma = \sigma' \cdot x \cdot \sigma''$, $x \notin \sigma''$ and either $\sigma' = \epsilon$ or σ' is a path whose first node is x. Either way, the above implies $\mathsf{simple}(\sigma) = x \cdot \mathsf{simple}(\sigma'')$.

If $\sigma'' = \epsilon$, σ is a path from x to x and $\mathsf{simple}(\sigma) = x$ is a simple path from x to x. Otherwise there are nodes $z \in x\bullet$ and y, such that σ is a path from x to y and σ'' is a path from z to y. By induction hypothesis, $\mathsf{simple}(\sigma'')$ is a simple path from z to y. Then $\mathsf{simple}(\sigma)$ is a simple path from x to y. □

The next result shows the other direction of the above.

Lemma C.2. *Let σ be such that $\mathsf{simple}(\sigma)$ is a path from x to y. Then σ is also a path from x to y.*

Proof. By definition of $\mathsf{simple}(-)$, $\sigma = \epsilon$ if and only if $\mathsf{simple}(\sigma) = \epsilon$. Since the latter contains at least one node, so does the former. Then σ is a path. Let z be the first node of σ. Then by Lemma C.1, z is also the first node of $\mathsf{simple}(\sigma)$. Then $z = x$. We use the same reasoning to show that y is the last node of σ. □

The following result gives a useful factorisation for the function $\mathsf{simple}(-)$.

Lemma C.3. *Let σ be a path and $x \in \mathsf{simple}(\sigma)$. Then there are σ' and σ'' such that $\sigma = \sigma' \cdot x \cdot \sigma''$ and $\mathsf{simple}(\sigma) = \mathsf{simple}(\sigma') \cdot \mathsf{simple}(x \cdot \sigma'')$.*

Proof. Since $x \in \mathsf{simple}(\sigma)$, the length of $\mathsf{simple}(\sigma)$ is greater than or equal to 1. If the length is 1, then $\mathsf{simple}(\sigma) = x$. By definition of the function $\mathsf{simple}(-)$, there is σ_x such that $\sigma = \epsilon \cdot x \cdot \sigma_x$ (note that σ_x is possibly ϵ). Then:

$$\mathsf{simple}(\sigma) \;=\; \mathsf{simple}(\epsilon \cdot x \cdot \sigma_x) \;=\; \epsilon \cdot \mathsf{simple}(x \cdot \sigma_x) \;=\; \mathsf{simple}(\epsilon) \cdot \mathsf{simple}(x \cdot \sigma_x)$$

and the claim is satisfied.

Suppose that the claim holds for length $n \geq 1$ and let $\mathsf{simple}(\sigma)$ have length $n + 1$. Hence, there are a node y and a path σ' of length n such that:

$$\mathsf{simple}(\sigma) \;=\; y \cdot \sigma'$$

First let $y = x$. By definition of $\mathsf{simple}(-)$, σ is of the form $\sigma = \epsilon \cdot x \cdot \sigma_x$, for some σ_x. Then, following the same reasoning given in the base case, we have $\mathsf{simple}(\sigma) = \mathsf{simple}(\epsilon) \cdot \mathsf{simple}(x \cdot \sigma_x)$, which concludes the proof.

Now let $y \neq x$. By definition of $\mathsf{simple}(-)$, σ is of the form $\sigma = \sigma_y \cdot \sigma'_y$, for some σ_y and σ'_y such that y is the first and last node of σ_y, $y \notin \sigma'_y$ and $\sigma' = \mathsf{simple}(\sigma'_y)$. Then $\mathsf{simple}(\sigma_y) = y$ and:

$$\mathsf{simple}(\sigma) \;=\; y \cdot \sigma' \;=\; \mathsf{simple}(\sigma_y) \cdot \mathsf{simple}(\sigma'_y)$$

$x \neq y$ implies $x \in \sigma'$. By induction hypothesis, there are σ_x and σ'_x such that $\sigma'_y = \sigma_x \cdot x \cdot \sigma'_x$ and $\sigma' = \mathsf{simple}(\sigma'_y) = \mathsf{simple}(\sigma_x) \cdot \mathsf{simple}(x \cdot \sigma'_x)$. Then $\sigma = (\sigma_y \cdot \sigma_x) \cdot x \cdot \sigma'_x$. Moreover, $y \notin \sigma'_y$ implies $y \notin \sigma_x$. Hence, $\mathsf{simple}(\sigma_y \cdot \sigma_x) = y \cdot \mathsf{simple}(\sigma_x) = \mathsf{simple}(\sigma_y) \cdot \mathsf{simple}(\sigma_x)$. Then:

$$\begin{aligned} \mathsf{simple}(\sigma) \;=\; \mathsf{simple}(\sigma_y) \cdot \mathsf{simple}(\sigma'_y) \;&=\; \mathsf{simple}(\sigma_y) \cdot \mathsf{simple}(\sigma_x) \cdot \mathsf{simple}(x \cdot \sigma'_x) \\ &=\; \mathsf{simple}(\sigma_y \cdot \sigma_x) \cdot \mathsf{simple}(x \cdot \sigma'_x) \end{aligned}$$

which concludes the proof. □

It follows as a corollary that every node contained in the result path is also contained in the original path.

Corollary C.1. *Let σ be a path in a Petri net graph $\boldsymbol{P}$ and let $x \in \mathsf{simple}(\sigma)$. Then $x \in \sigma$.*

C.2 Diamonds

We start by introducing some technical notions related to diamonds that are used throughout the Appendix, then we present the results.

Diamonds: technical notions. We formalise the notion of entry path we informally used in § 2 to define cross-free diamonds. Coincidentally, we introduce the new notion of *straight diamond*, by relaxing the cross-free condition: like cross-free diamonds, straight diamonds require an entry path; but unlike cross-free diamonds, they allow criss-crossing paths between the two sides of the diamond.

Definition C.1 (Entry path of a diamond, straight diamond). *Let $\delta = \langle \sigma_1, \sigma_2 \rangle$ be a diamond with start node x. Then a path σ from an initial node y to some node $z \in {\bullet x}$ is an* entry path *of δ if and only if $\sigma \cap (\sigma_1 \cup \sigma_2) = \emptyset$. If such is the case, we say that δ is a* straight diamond *with entry path σ.*

Checking the cross-free condition of a diamond, requires some tedious quantifications over its two paths and all the paths that go from a node in the first path to a node in the second path. Sometimes, it is possible to abstract from those quantifications, by using the following two families of sets. Let $\sigma_1, \ldots \sigma_n$ be paths and let $X' \subseteq X$ be a subset of the set of all nodes X. Then we define:

$$S^{X'}_{i,j} = \{x \in \sigma_i | \exists_{y \in \sigma_j} : (x, y) \in F^*_{-X'}\} \tag{1a}$$

$$\bar{S}^{X'}_{i,j} = \{y \in \sigma_j | \exists_{x \in \sigma_i} : (x, y) \in F^*_{-X'}\} \tag{1b}$$

where $i, j \in \{1, \ldots n\}$. Generally, we assume that $X' = \sigma \cup \{t, t'\}$, for some path σ and transitions t and t', such that there exists some diamond from t to t' with σ as entry path. We also assume that $t\sigma_i t'$ is a path, for all $i \in \{1, \ldots, n\}$. If σ, t and t' are clear from the context, we leave $X' = \sigma \cup \{t, t'\}$ implicit, e.g. we write $S_{i,j}$ and $\bar{S}_{i,j}$ to stand for $S^{\sigma \cup \{t,t'\}}_{i,j}$ and $\bar{S}^{\sigma \cup \{t,t'\}}_{i,j}$, respectively.

Note that $S_{i,j} \neq \emptyset$ if and only if $\bar{S}_{i,j} \neq \emptyset$. Let also $\leq_\sigma$ be the total order induced by a path σ on its nodes, as follows. Given $x, y \in \sigma$, $x \leq_\sigma y$ if and only if either $x = y$ or the latest occurrence of x appears before the latest occurrence of y in σ. In the following proofs, we may refer to *the maximal element of* $S_{i,j}$ and to *the minimal element of* $\bar{S}_{i,j}$, where: maximality in $S_{i,j}$ is with respect to $\leq_{\sigma_i}$; whereas minimality in $\bar{S}_{i,j}$ is with respect $\leq_{\sigma_j}$. Given $x, y \in \sigma_i \setminus \sigma$, we have that: $x \leq_{\sigma_i} y$ and $y \in S^{\sigma}_{i,j}$ imply $x \in S^{\sigma}_{i,j}$; while $x \leq_{\sigma_i} y$ and $x \in \bar{S}^{\sigma}_{j,i}$ imply $y \in \bar{S}^{\sigma}_{j,i}$. In other words, a non-empty $S_{i,j}$ is bijectively mapped to its maximal element, while a non-empty $\bar{S}_{i,j}$ is bijectively mapped to its minimal element.

Diamonds: results. The following result shows that $\mathsf{simple}(-)$ applied to the entry path of a diamond returns an entry path for the same diamond again. It follows that if a diamond has an entry path, then it has also a simple entry path.

Lemma C.4. *Let σ be an entry path for a diamond δ, then $\mathsf{simple}(\sigma)$ is also an entry path for δ.*

Proof. Let $\delta = \langle \sigma_1, \sigma_2 \rangle$. Suppose that there exists $x \in \mathsf{simple}(\sigma) \cap (\sigma_1 \cup \sigma_2)$. Then $x \in \mathsf{simple}(\sigma)$. By Corollary C.1, this implies $x \in \sigma$. Since x is also contained in $\sigma_1 \cup \sigma_2$, we have $x \in \sigma \cap (\sigma_1 \cup \sigma_2)$. But this is impossible since σ is an entry path of δ. Then $\mathsf{simple}(\sigma) \cap (\sigma_1 \cup \sigma_2) = \emptyset$. Then $\mathsf{simple}(\sigma)$ is an entry path of δ. □

The next result shows a nice factorisation of the function $\mathsf{simple}(-)$, when applied to a path on either side of a diamond.

Lemma C.5. *Let $\langle x \cdot \sigma \cdot y,\ x \cdot \sigma' \cdot y \rangle$ be a diamond from a node x to a node y. Then $\mathsf{simple}(x \cdot \sigma \cdot y) = x \cdot \mathsf{simple}(\sigma) \cdot y$ and $\mathsf{simple}(x \cdot \sigma' \cdot y) = x \cdot \mathsf{simple}(\sigma') \cdot y$.*

Proof. We do the proof for σ, that for σ' follows the same reasoning. The node x appears only once in $x \cdot \sigma \cdot y$, then $x \notin \sigma \cdot y$. This implies $\mathsf{simple}(x \cdot \sigma \cdot y) = x \cdot \mathsf{simple}(\sigma \cdot y)$. By Lemma C.1, $\mathsf{simple}(\sigma \cdot y)$ is a simple path that ends at y. Then y does appear in $\mathsf{simple}(\sigma \cdot y)$. By Lemma C.3, there are σ' and σ'' such that $\sigma \cdot y = \sigma' \cdot y \cdot \sigma''$ and $\mathsf{simple}(\sigma \cdot y) = \mathsf{simple}(\sigma') \cdot \mathsf{simple}(y \cdot \sigma'')$. Since y appears only once in $\sigma \cdot y = \sigma' \cdot y \cdot \sigma''$, we have $\sigma = \sigma'$ and $\sigma'' = \epsilon$. Then $\mathsf{simple}(\sigma \cdot y) = \mathsf{simple}(\sigma) \cdot y$ and $\mathsf{simple}(x \cdot \sigma \cdot y) = x \cdot \mathsf{simple}(\sigma) \cdot y$. □

The following result justifies the definitions of diamond and cross-free diamond.

Lemma C.6. *Let $\delta = \langle x\sigma y,\ x\sigma' y \rangle$ be a diamond from a node x to a node y. Then $\delta' = \langle \mathsf{simple}(x\sigma y), \mathsf{simple}(x\sigma' y) \rangle$ is a diamond from x to y. Moreover, if δ is a cross-free diamond, then δ' is a cross-free diamond.*

Proof. By Lemma C.1, $\mathsf{simple}(x\sigma y)$ and $\mathsf{simple}(x\sigma' y)$ are both simple paths which start from x and end at y. Let $z \in \mathsf{simple}(x\sigma y) \cap \mathsf{simple}(x\sigma' y)$. By Corollary C.1, $z \in (x\sigma y) \cap (x\sigma' y)$. Since $\sigma \cap \sigma' = \emptyset$, either $z = x$ or $z = y$. Then δ' is a diamond from x to y.

Now, suppose that δ is cross-free and let $\bullet x = \{z_1, \ldots, z_n\}$. By definition of cross-free diamond, there are an initial node z_0 and paths $\sigma_0, \ldots, \sigma_n$ such

that σ_i starts from z_0 and ends at z_i (for all $i \in \{1, \ldots, n\}$) and δ is pre-cross-free in the graph obtained by removing σ_1, $\ldots \sigma_{n-1}$ and σ_n. Without loss of generality, we assume that $\forall_{z \in \sigma, z' \in \sigma'} (z, z') \notin F^*_{-\sigma_1 \cup \ldots \cup \sigma_n \cup \{x,y\}}$. By Lemma C.5, $\mathsf{simple}(x\sigma y) = x \cdot \mathsf{simple}(\sigma) \cdot y$ and $\mathsf{simple}(x\sigma' y) = x \cdot \mathsf{simple}(\sigma') \cdot y$. By Corollary C.1, any $z \in \mathsf{simple}(\sigma)$ and $z' \in \mathsf{simple}(\sigma')$ are such that $z \in \sigma$ and $z' \in \sigma'$. Then $(z, z') \notin F^*_{-\sigma_1 \cup \ldots \cup \sigma_n \cup \{x,y\}}$. Then δ' is a cross-free diamond from x to y. □

The following result shows that any straight diamond "subsumes" some cross-free diamond.

Lemma C.7. *Let $\delta = \langle x\sigma_1 y,\ x\sigma_2 y\rangle$ be a straight diamond from a node x to a node y with entry path σ_0, and let z be the last node of σ_1. Then there is a cross-free diamond $\delta' = \langle x'\sigma'_1 y,\ x'\sigma'_2 y\rangle$ with entry path σ'_0, where σ'_1 is a path ending with z, $x'\sigma'_2$ is a suffix of $x\sigma_2$ and $\sigma_0 x$ is a prefix of $\sigma'_0 x'$.*

Proof. The trivial case is when $S^{\sigma_0 \cup \{x,y\}}_{2,1} = \emptyset$, as it implies directly that $\delta = \langle x\sigma_1 y,\ x\sigma_2 y\rangle$ is a cross-free diamond with entry path σ_0.

Suppose that there is $x_2 \in S^{\sigma_0 \cup \{x,y\}}_{2,1}$. Then there is a path $\sigma_{2,1}$ from x_2 to z such that $\sigma_{2,1} \cap (\sigma_0 \cup \{x, y\}) = \emptyset$. Note that $\sigma_{2,1} \cap \sigma_2 \neq \emptyset$, since $x_2 \in \sigma_2$. Then there are a node $x' \in \sigma_2$ and a path σ'_1 such that $x'\sigma'_1$ is a suffix of $\sigma_{2,1}$ and $\sigma'_1 \cap \sigma_2 = \emptyset$ (i.e. x' is the last node of $\sigma_{2,1}$ which occurs also in σ_2). Then $\sigma_2 = \sigma_{xx'} x' \sigma'_2$, for some paths $\sigma_{xx'}$ and σ'_2 such that $x' \notin \sigma'_2$. Note that $\delta' = \langle x'\sigma'_1 y,\ x'\sigma'_2 y\rangle$ is a straight diamond with entry path $\sigma'_0 = \sigma_0 x \sigma_{xx'}$. Note also that z is the last node of σ'_1 and $x'\sigma'_2$ is a suffix of σ_2.

Now, if δ' is cross-free we are done; otherwise we apply the reasoning above recursively, until we find a cross-free diamond which satisfies the requirements, where the number of recursive applications is bounded by the length of σ_2. □

D Appendix: Session Graphs

In this section we prove some results on session graphs. They apply to both well-formed and non-well-formed SGs.

We introduce *S-paths*, which will be used in Lemma D.3, and later in the proof of safety. A path σ is an S-path if and only if, for all $t \in T$ and $p, q \in P$:

$$(((t,p) \in \sigma \wedge (t,q) \in \sigma) \ \vee \ ((p,t) \in \sigma \wedge (q,t) \in \sigma)) \quad \Rightarrow \quad p = q$$

This is a new definition, however it is inspired by the relatively standard definition of *S-net* [10] (also called *state-machine* in, e.g. [17]): intuitively, the graph underlying an S-path is an S-net, since the place that precedes any occurrence of a transition t in σ is unique; the same for the place that follows t.

The proofs for the first three results are rather straightforward checks on the definition of IRTs, ORTs and their compositions. The first one says that SGs satisfy the free-choice property stated in the main section.

Proof of Proposition 2.1. An RS consists of two trees sharing only the root and with arcs going towards the root in one case and away in the other. Then it

satisfies free-choice. Moreover, input transitions may only occur in the IRT part of their RS, they only have incoming arcs from places with a unique outgoing arc. In Def. 2.2, we can form SGs by connecting an output transition of a RS with an input transition of another RS through a communication place. Each communication place has exactly one outgoing arc. Then SGs also satisfy free-choice. □

In our proofs, we also use the following dual property of free-choice: for any arc from a transition t to a place p, either $\bullet p = \{t\}$ or $t\bullet = \{p\}$. To ensure this property, we consider a slight variant of SG, whereby each IRT that contains some output transition t is such that: given $p \in t\bullet$, the unique successor of t in the IRT, we have $\bullet p = \{t\}$. Note that the above already holds in ORTs, where every place has at most one incoming arc. We could have included the above condition in the original definition of IRT. However, it is quite pedantic, while also being an extremely minor requirement: we can apply a canonical transformation that inserts a dummy sequence of place-then-transition, after each output transition occurring in an IRT, and makes the condition hold for every SG. Up to this transformation, then we state the following.

Lemma D.1. *For any transition t in some SG and $p \in t\bullet$, either $\bullet p = \{t\}$ or $t\bullet = \{p\}$.*

Proof. An RS consists of two trees sharing only the root and with arcs going towards the root in one case and away in the other. Then it satisfies the stated property. Moreover, output transitions only have outgoing arcs to places with a unique incoming arc, both in the IRT and in the ORT parts of their RS. In Def. 2.2, we can form SGs by connecting an output transition of a RS with an input transition of another RS through a communication place. Each communication place has exactly one incoming arc. Then SGs also satisfy the stated property. □

The following is another structural result which follows from the way RSs are defined. The proof follows from the same observations given for the above two results, hence we omit it.

Lemma D.2. *For any transition t in some SG, $|\bullet t| > 1$ implies $|t\bullet| = 1$ and $|t\bullet| > 1$ implies $|\bullet t| = 1$.*

The following is a result on S-paths that is used in the proof of safety.

Lemma D.3. *Let $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ be an SG with initial place p_I. Let $t \in T$, let $p \in t\bullet$ and let σ be an S-path from p_I. Then:*

(1) $p \in \sigma \;\Rightarrow\; t\bullet \cap \sigma = \{p\}$ (2) $p, t \notin \sigma \;\Rightarrow\; t\bullet \cap \sigma = \emptyset$

Proof. The proof is immediate in the case of $t\bullet = \{p\}$. Then let $|t\bullet| > 1$, which means that there exists $p' \in t\bullet$ such that $p' \neq p$. By Lemma D.1, $\bullet p = \bullet p' = \{t\}$.

For the proof of (1), note that neither p nor p' are initial. Hence, any occurrence of p or of p' in *sigma* must follow an occurrence of t. Conversely, since σ is an

S-path, any occurrence of t be followed by an occurrence of the same place. Then either p or p' does not occur in σ. Since p does occur in σ, p' does not.

The proof of (2) is easier. As stated above, any occurrence of p' in σ must follow an occurrence of t. But t does not occur in σ. Then neither does p'. □

E Appendix: Well-Formed Session Graphs

This section shows some properties of well-formed SGS, which we use in later sections. We conclude by confirming that well-formedness is decidable (Proposition 2.2).

E.1 Properties of Well-formed Session Graphs

The first result says that every transition following the initial place has no other incoming arc, except for the one from the initial place.

Lemma E.1. *Let $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ be a well-formed SG with initial place p_I. Let $t \in p_I\bullet$. Then $\bullet t = \{p_I\}$.*

Proof. By contradiction, let there be $q \in \bullet t$ such that $q \neq p_I$. By Definition 2.3(R3), $(p_I, q) \in F^*$.

Suppose that $(p_I, q) \in F^*_{-t}$. Then $(t, q) \in F^*$ and, therefore, there is a cycle φ which contains the arc (q, t). Since $p_I \in \bullet t \setminus \varphi$, Definition 2.3(C1) applies. An implication is that $(t, p_I) \in F^*$. But that is impossible, since $\bullet p_I = \emptyset$ by definition of initial place. Then we have a contradiction.

Now suppose that $(p_I, q) \notin F^*_{-t}$. Since $(p_I, q) \in F^*$, there must be $t' \in p_I\bullet$ such that $t' \neq t$. But by Proposition 2.1, $p_I\bullet = \{t\}$ since $|\bullet t| > 1$. Again, we have a contradiction. Then $\bullet t = \{p_I\}$ □

The following result says that whenever there is an arc from a place p to a transition t, then there is a path from the initial place to p which does not visit t.

Lemma E.2. *Let $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ be a well-formed SG with initial place p_I. Let $p \in P$ and $t \in T$ such that $(p, t) \in F$. Then $(p_I, p) \in F^*_{-t}$.*

Proof. By contradiction, suppose that $(p_I, p) \notin F^*_{-t}$. By Definition 2.3(R3), $(p_I, p) \in F^*$. Then there is a simple path σ from p_I to p which visits t. Then $\sigma = \sigma_t t \sigma_{tp}$, for some paths σ_t and σ_{tp}. Since $p \in \bullet t$, there is a cycle $\varphi = t\sigma_{tp}t$. By simplicity of σ, we have $\sigma_t \cap \sigma_{tp} = \emptyset$. Then t is an entry node of φ. By Definition 2.3(C1), $t \in P$. Contradiction. □

The following result says that, if a transition has more than one incoming arc, then it may not be part of a cycle of only two nodes.

Lemma E.3. *Let $t \in T$ be a transition of a well-formed SG $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ such that $|\bullet t| > 1$. Then $\bullet t \cap t\bullet = \emptyset$.*

Proof. By contradiction, let $p \in {\bullet t} \cap t{\bullet}$ and let $q \in {\bullet t}$ be such that $q \neq p$. Let $p_I \in Init(\boldsymbol{G})$ be the initial place of $\boldsymbol{G}$. By Lemma E.2, $(p_I, q) \in F^*_{-t}$. Then let σ be a simple path from p_I to q where t does not occur. We are going to show that p does not occur in σ either. By Proposition 2.1, $|{\bullet t}| > 1$ and $p \in {\bullet t}$ imply $p{\bullet} = \{t\}$. Then $p \in \sigma$ implies $t \in \sigma$ as well. But $t \notin \sigma$, then $p \notin \sigma$. Then t is an entry node for the cycle tpt. By Definition 2.3(C1), $t \in P$. Then we have our contradiction. □

The following result says that if an entry node of a cycle is contained in a diamond, then the ending transition of the diamond is not contained in the cycle. As a technical remark, note that the diamond is not required to be straight. However, the path considered, going from the initial node to the start node of the diamond, is required to be simple.

Lemma E.4. *Let $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ be a well-formed SG with initial place p_I. Let σ be a simple path from p_I to $t \in T$, let $\langle \sigma_1, \sigma_2\rangle \in \mathsf{diamond}(t, t')$ be a diamond from t to $t' \in T$ and let $\varphi \in \Phi$ such that $\sigma \cap \varphi = \emptyset$. Then $t' \notin \varphi$.*

Proof. We assume $t' \in \varphi$ and show that this leads to a contradiction. Let σ' be the prefix of σ which ends right before t, i.e. $\sigma = \sigma' t$. We do the proof by cases.

Case 1: $\varphi \cap (\sigma_1 \cup \sigma_2) = \{t'\}$. The path $\sigma'\sigma_1$ goes from p_I to t'. Moreover, it only intersects φ in t'. Therefore, t' is an entry node for φ. By Definition 2.3(R1), $t' \in P$. On the other hand, Definition 2.3(D1) implies $t' \in T$, since $t \in T$. Contradiction.

Case 2: $\{t'\} \subsetneq \varphi \cap \sigma_1$. Let $x \in X$ be the first node occurring in σ_1 which also occurs in φ. Note that $x \neq t$, since $t \in \sigma$ and $\sigma \cap \varphi = \emptyset$. Let σ_x be the prefix of σ_1 ending at x. Then $\mathsf{simple}(\sigma'\sigma_x)$ is a simple path from p_I to x which only intersects φ in x. Then x is an entry node for φ. By Definition 2.3(R1), $x \in P$.

Now let $y \in X$ be the first node occurring in σ_2 which also occurs in φ (if $\sigma_2 \cap \varphi = \{t'\}$, then $y = t'$). Let σ_y be the prefix of σ_2 ending at y and let σ_{yx} be the sub-path of φ starting from the successor of y and ending at x. Note that $\sigma_y\sigma_{yx}$ is a simple path. Note also that $\sigma_x \cap \sigma_y = \sigma_x \cap \sigma_2 = \{t\}$ and $\sigma_x \cap \sigma_{yx} = \sigma_x \cap \varphi = \{x\}$. Then $\langle \sigma_x, \sigma_y\sigma_{yx}\rangle \in \mathsf{diamond}(t, x)$. By Definition 2.3(D1), $t \in T$ implies $x \in T$. Contradiction.

Case 3: $\{t'\} \subsetneq \varphi \cap \sigma_2$. This case is symmetric to the previous case. □

The following result complements the previous one by saying that if an exit node of a cycle is contained in a diamond, then the starting transition of the diamond is not contained in the cycle. The proof is dual and we omit it.

Lemma E.5. *Let $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ be a well-formed SG and $p_T \in Term(\boldsymbol{G})$. Let σ be a simple path from $t \in T$ to p_T, let $\langle \sigma_1, \sigma_2\rangle \in \mathsf{diamond}(t', t)$ be a diamond from $t' \in T$ to t and let $\varphi \in \Phi$ such that $\sigma \cap \varphi = \emptyset$. Then $t' \notin \varphi$.*

The following result says that any simple path from the initial place is the prefix of an S-path that ends at a terminal place.

Lemma E.6. *Let $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ be a well-formed SG with initial place p_I. For any simple path σ from p_I, there are a terminal place p_T and a simple path σ' ending at p_T such that $\sigma\sigma'$ is an S-path.*

Proof. Let $x \in X$ be the last node occurring in σ, where $X = P \cup T$ is the set of nodes. By Definition 2.3(R3), there is $p_T \in Term(\boldsymbol{G})$ such that $(x, p_T) \in F^*$. Let $x\sigma'$ be a path from x to p_T which, by Lemma C.1, we can assume to be simple. The case of $x = p_T$ is trivial, then let $x \neq p_T$. If $\sigma \cap \sigma' = \emptyset$ then $\sigma\sigma'$ is simple and, therefore, it is also an S-path.

Suppose that $\sigma \cap \sigma' \neq \emptyset$. Let y be the first node of σ' which also appears in σ. Let also σ_{yx} be the suffix of σ which starts from y and σ_y be the prefix of σ' which ends at y. Since both σ_{yx} and σ_y are simple, $\sigma_{yx}\sigma_y$ is a cycle and y is an entry node for it. By Definition 2.3(C1), $y \in P$.

Since $\sigma \cap \sigma_y = \{y\} \subseteq P$ and both σ and σ_y are simple, $\sigma\sigma_y$ is an S-path. Now let z be the last node of σ' which also appears in σ_{yx}. Then z is an exit node and by Definition 2.3(C2), $z \in P$. Let $y\sigma_z$ be the prefix of σ_{yx} which ends at z. Note that $y\sigma_z$ is a subpath of σ, then since $\sigma\sigma_y$ is an S-path, $\sigma\sigma_y\sigma_z$ is an S-path as well.

Let $z\sigma''$ be the suffix of σ' which starts from z. Since z is the last node of σ' which also appears in σ_{yx} and since σ' is simple, we have $\sigma'' \cap \sigma_{yx}\sigma_y = \emptyset$. Then we can apply all the previous reasoning to the prefix of σ ending at z (which is simple like σ) and σ'' (which is simple like σ' but shorter). The path obtained extends $\sigma\sigma_y\sigma_z$, while preserving the properties of S-paths. By applying this reasoning recursively, we eventually obtain an extension of σ which ends in p_T. If the suffix used for the extension is not simple, apply $\mathsf{simple}(-)$ to it and show that if $\sigma_1\sigma_2$ is an S-path, then $\sigma_1\mathsf{simple}(\sigma_2)$ is an S-path. □

The following result shows that any path from the initial place to a transition with multiple incoming arcs has a suffix, which is one of the two sides of a cross-free diamond. First, we show the case where said transition occurs only once in the path.

Lemma E.7. *Let $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ be a well-formed SG with initial place $p_I \in Init_{\boldsymbol{G}}(P)$. Let σ be a path from p_I to $t \in T$ such that t occurs only once in σ, and let $q \in {\bullet}t$ such that $q \notin \sigma$. Then there are a suffix σ' of σ and a simple path σ'' such that $\langle \sigma',\ \sigma''\rangle$ is a cross-free diamond, and σ'' ends with (q, t).*

Proof. By Lemma E.2, there is a simple path σ_q from p_I that ends with the arc (q, t). By Lemma C.1, $\mathsf{simple}(\sigma)$ is also a simple path from p_I to t but does not contain q, by Corollary C.1. Without loss of generality, let $\sigma_q = \sigma_0 x \sigma_1 t$ and $\mathsf{simple}(\sigma) = \sigma_0 x \sigma_2 t$, for some paths σ_0, σ_1 and σ_2, and node x such that $S_{2,1} = S_{2,1}^{\sigma_0 \cup \{x,t\}} = \emptyset$, where we use $S_{i,j}$ and $\bar{S}_{i,j}$ as defined in (1a) and (1b), respectively (i.e. among all the paths from p_I ending with (q, t), σ_q is the one that shares the longest prefix with $\mathsf{simple}(\sigma)$). Note that σ_0, σ_1 and σ_2 are simple and that $\langle x\sigma_1 t, x\sigma_2 t\rangle$ is a cross-free diamond with entry path σ_0.

By Lemma C.3, there are σ'_0 and σ_3 such that $\sigma = \sigma'_0 x \sigma_3 t$, $\mathsf{simple}(\sigma'_0) = \sigma_0$ and $\mathsf{simple}(x\sigma_3 t) = x\sigma_2 t$. We are only interested in the suffix of $x\sigma_3 t$ that starts from the last occurrence of x, hence we assume $x \notin \sigma_3$, without loss of generality.

By contradiction, suppose that $S_{3,1} \neq \emptyset$ (i.e. there is a node of σ_1 which can be reached from some node of σ_3). From $\mathsf{simple}(x\sigma_3 t) = x\sigma_2 t$ and $x \notin \sigma_3$, we infer $\mathsf{simple}(\sigma_3 t) = \sigma_2 t$. Then $\sigma_3 t$ and $\sigma_2 t$ start from the same node, by Lemma C.1. This implies $\bar{S}_{2,3} = \sigma_3$ (i.e. every node in σ_3 is reachable from some node in σ_2). From $\bar{S}_{2,3} = \sigma_3$ and $S_{3,1} \neq \emptyset$, we infer $S_{2,1} \neq \emptyset$. This contradicts a previous statement. Then $S_{3,1} = \emptyset$. Then $\langle x\sigma_1 t, x\sigma_3 t\rangle$ is a cross-free diamond from x to t with entry path σ_0. □

Now we show the rest of the result.

Lemma E.8. *Let $\boldsymbol{G} = \langle P, T, F, f, g\rangle$ be a well-formed SG with initial place $p_I \in Init_{\boldsymbol{G}}(P)$. Let σ be a path from p_I with last arc $(p,t) \in F$ and let $q \in {}^\bullet t$ such that $q \neq p$. Then there are a suffix σ' of σ and a simple path σ'' such that $\langle \sigma',\ \sigma''\rangle$ is a cross-free diamond, and σ'' ends with (q,t).*

Proof. The case where t appears only once in σ is given by Lemma E.7. In this proof, we assume that t appears more than once in σ. Then there is a suffix of σ of the form $t\sigma_t t$, for some path σ_t such that $t \notin \sigma_t$. By Lemma E.2, there is a simple path σ_q from p_I to q such that $t \notin \sigma_q$. We consider two separate cases: in the first case, we assume $\sigma_t \cap \sigma_q \neq \emptyset$; in the second case, we show that the assumption $\sigma_t \cap \sigma_q = \emptyset$ leads to a contradiction.

case: $\sigma_t \cap \sigma_q \neq \emptyset$. Let $t'\sigma'_t$ be the suffix of σ_t such that $t' \in \sigma_q$ and $\sigma'_t \cap \sigma_q = \emptyset$ (i.e. t' is the last node of σ_t which is also a node of σ_q). Then $\sigma_q = \sigma_0 t' \sigma'_q$, for some σ_0 and σ'_q. Since σ_q is simple, $t' \notin \sigma'_q$. And $\sigma'_t \cap \sigma'_q = \emptyset$, since $\sigma'_t \cap \sigma_q = \emptyset$. Then $\langle t'\sigma'_t t, t'\sigma'_q t\rangle$ is a diamond. Moreover, since σ_q is simple, we have $\sigma'_q \cap \sigma_0 = \emptyset$. And since $\sigma'_t \cap \sigma_q = \emptyset$, we also have $\sigma'_t \cap \sigma_0 = \emptyset$. Then $\langle t'\sigma'_t t, t'\sigma'_q t\rangle$ is a straight diamond with entry path σ_0. Then we can apply Lemma C.7 and we are done.

case: $\sigma_t \cap \sigma_q = \emptyset$. By Lemma E.2, there is a simple path σ_p from p_I to p such that $t \notin \sigma_p$. Let $t'\sigma'_p$ be a suffix of σ_p such that $t' \in \sigma_q$ and $\sigma'_p \cap \sigma_q = \emptyset$. Then σ_q is of the form $\sigma_q = \sigma_0 t' \sigma'_q$, for some simple paths σ_0 and σ'_q. From $\sigma'_p \cap \sigma_q = \emptyset$ we infer both $\sigma'_p \cap \sigma'_q = \emptyset$. Then $\delta = \langle t'\sigma'_p t, t'\sigma'_q t\rangle$ is a diamond. Let $\varphi_t = t \cdot \mathsf{simple}(\sigma_t t)$. By Lemma C.1, φ_t is a cycle. From $\sigma_t \cap \sigma_q = \emptyset$ and $t \notin \sigma_q$, we infer $\varphi_t \cap \sigma_0 t' = \emptyset$. But this contradicts Lemma E.4, since $t \in \varphi_t$. □

The following result says that any simple path from the end node of a cross-free diamond to a terminal place does not intersect any other node of the diamond.

Lemma E.9. *Let $\langle t\sigma_0 t',\ t\sigma_1 t'\rangle$ be a cross-free diamond in a well-formed SG $\boldsymbol{G} = \langle P, T, F, f, g\rangle$. Let σ_T be a simple path from t' to some terminal place $p_T \in Term_{\boldsymbol{G}}(P)$. Then $\sigma_T \cap (\sigma_0 \cup \sigma_1 \cup \{t\}) = \emptyset$.*

Proof. By contradiction, suppose that $\sigma_T \cap (\sigma_0 \cup \sigma_1 \cup \{t\}) \neq \emptyset$. Then $\sigma_T = \sigma'_{T1} x_1 \sigma_{T1}$, for some x_1, σ_{T1} and σ'_{T1} such that $x_1 \in (\sigma_0 \cup \sigma_1 \cup \{t\})$ and $\sigma_{T1} \cap$

$(\sigma_0 \cup \sigma_1 \cup \{t\}) = \emptyset$. Without loss of generality, let us assume that $x_1 \in t\sigma_1$ and let $x_1\sigma_{x1}$ be a suffix of $t\sigma_1$. Then x_1 is an exit place for the cycle $\varphi_1 = \mathsf{simple}(x_1\sigma_{x1}\sigma'_{T1}) \cdot x_1$. By Definition 2.3(C2), $x_1 \in P$. This implies $x_1 \neq t$ and $x_1 \in \sigma_1$. By Definition 2.3(D3), there is σ_2 such that $\langle t\sigma_0 t',\ t\sigma_2 t' \rangle$ is a cross-free diamond and $\sigma_{T1} \cap \sigma_2 \neq \emptyset$.

Then there is a suffix of σ_{T1} of the form $x_2\sigma_{T2}$, for some x_2 and σ_{T2} such that $x_2 \in \sigma_2$ and $\sigma_{T2} \cap \sigma_2 = \emptyset$. By applying the above reasoning again, we infer that x_2 is the exit place of a cycle φ_2 such that $\varphi_2 \cap \sigma_{T2} = \emptyset$, and that $x_2 \in P$. But since σ_{T2} is strictly shorter than σ_{T1}, this reasoning can only be applied a finite number of times. Eventually, we obtain a suffix σ_{Tn} of σ_T and a path σ_n such that $\langle t\sigma_0 t',\ t\sigma_n t' \rangle$ is a cross-free diamond and the last node of σ_{Tn} appears also in σ_n. Since σ_{Tn} is a suffix of σ_T, its last node is p_T. Since p_T is a terminal place, $p_T\bullet = \emptyset$. But $p_T \in \sigma_n$ implies $p_T\bullet \neq \emptyset$. Contradiction. □

Note that the proof of Lemma E.9 above does not include any reference to any entry path. In other words, the definition of cross-free diamond is never invoked explicitly. Instead, Definition 2.3(D3) is used for the conditions that a cross-free diamond must obey.

E.2 Decidability of Well-formedness

This section shows that well-formedness is decidable (Proposition 2.2). We start by providing an outline, then we add all the proof details.

The definition of well-formed SG (Definition 2.3) relies on two main structures: cycles (for conditions (C1) and (C2)) and diamonds (for conditions (D1), (D2) and (D3)). While a cycle is defined as a simple path that goes back to the start node, a diamond is defined as a pair of (non-necessarily simple) paths with the same start and end nodes. In general, there may be an infinite number of paths between two nodes: for instance, if there is a path which contains a cycle, then there are infinitely many other paths that only differ from the first one in the number of repetitions of the cycle. Therefore, the key for decidability is to show that the diamond conditions in the definition of well-formed SG (Definition 2.3(D1), (D2) and (D3)) can be verified by checking simple paths only, which are finitely enumerable. The hardest condition is Definition 2.3(D3), which requires checking a property on every single node inside every diamond.

The following result reduces Definition 2.3(D3) to the verification of the existence of *some* diamond and of an additional condition, which boils down to verifying the existence of some simple paths. The proof is given at the end of this section.

Lemma E.10. *Let $\boldsymbol{G} = \langle P, T, F, f, g \rangle$ be an SG. Then $\boldsymbol{G}$ satisfies Definition 2.3(D3) if and only if it satisfies the following condition. Let $\delta = \langle t\sigma_1 t',\ t\sigma_2 t' \rangle$ be a cross-free diamond from a transition t to a transition t', where σ is a simple entry path of δ, and q is the last node of σ_1. Then for any place p such that:*

$$(t, p) \in F^*_{-\sigma \cup \sigma_2 t'} \ \wedge \ (p, q) \in F^*_{-\sigma t \sigma_2 t'}$$

and for any transition $t'' \neq t'$ such that $(p, t'') \in F$, the following holds:

$$(t, t'') \in F^*_{-\sigma \cup \sigma_2 t'} \ \wedge \ (t'', q) \in F^*_{-\sigma t \sigma_2 t'}$$

The main point about Lemma E.10 is that it allows us to only check for simple paths (e.g. a condition of the form $(p, q) \in F^*_{-\sigma}$) and for the *existence* of diamonds and cross-free diamonds (as we do for Definition 2.3(D1) and Definition 2.3(D2)). We do not have to check a property like Definition 2.3(D3), for any place that is contained in *any* diamond.

Such an existential verification is further simplified by applying Lemma C.6, which shows that: if a (cross-free) diamond between two nodes exists, then a *simple* (cross-free) diamond between the same nodes also exists, where a diamond is simple just when its two paths are both simple. Note also that the "only if" direction follows from the fact that every simple diamond is also a diamond. Similarly, Lemma C.4 shows that if a diamond has some entry path (which is the case for cross-free diamonds) then it has also a simple entry path.

We have thus shown that, all we need to verify Definition 2.3, is to check for the existence of a relatively small set of simple paths. Then well-formedness is decidable, since the length of any simple path is bounded by the number of nodes in the graph.

Proof of Proposition 2.2. It follows from Lemmata C.4, C.6 and E.10. □

Decidability: a technical remark. In the rest of this section we show the proof of Lemma E.10. The results that we introduce for this proof are not used in the proofs of progress and safety and can thus be skipped. Nonetheless, they do provide further insights on Definition 2.3(D3). We start by introducing some technical notions that are used in the proof.

Let us recall the definition of cross-free diamond. Let δ be a diamond with start node x. Then δ is cross-free if it is pre-cross-free in the graph obtained by removing a path from an initial node to each $y \in \bullet x$. By Definition 2.3(R1), there is a unique initial place p_I. If x is a transition, $|x\bullet| > 1$ by definition of diamond. Then we can apply Lemma D.2 and infer that there is a unique place $p \in \bullet x$. For the rest of this section, we are treating Definition 2.3(D3), that is only concerned with cross-free diamonds which do start from a transition. Then, *in this particular case*, the definition of cross-free diamond given above can be rephrased as follows. The diamond δ given above is cross-free if it is pre-cross-free in the graph obtained by removing a single entry path σ, from p_I to p.

Decidability: proof details. We strengthen Definition 2.3(D3) progressively, by imposing additional conditions on the derived diamond with respect to the original diamond. The next result shows that the replacement path in the new diamond ends at the same place as the path it replaces.

Lemma E.11. *Let $\delta = \langle t\sigma_1 y, t\sigma_2 y \rangle$ be a cross-free diamond in an SG $\boldsymbol{G} = \langle P, T, F, f, g \rangle$, where $t \in T$ is a transition. Then, for any place $p \in \sigma_1$ and*

transition $t' \in p\bullet$, there is a path σ'_1 such that $t' \in \sigma'_1 y$ and $\langle t\sigma'_1 y, t\sigma_2 y\rangle$ is a cross-free diamond. Moreover, if σ_1 ends at $q \in P$, then so does σ'_1.

Proof. Let $p \in \sigma_1$ be a place and $t' \in p\bullet$ a transition. Let us assume that $t' \notin \sigma_1 y$, as the other case is immediate. A consequence of this is that $|p\bullet| > 1$. By Definition 2.3(D3), there is a path σ_3 such that $t' \in \sigma_3 y$ and $\delta' = \langle t\sigma_3 y, t\sigma_2 y\rangle$ is a cross-free diamond. By Proposition 2.1, $|p\bullet| > 1$ implies $\bullet t' = \{p\}$. Then $p \in \sigma_3$ as well.

By contradiction, suppose that p is the maximal element of $S_{3,1}$, where $S_{3,1} = S_{3,1}^{\sigma,t,y}$ is defined as in (1a) and σ is some entry path of δ. Let: $p\sigma_{1p}$ be the suffix of σ_1 such that $p \notin \sigma_{1p}$ (i.e. the suffix following the last occurrence of p); $p\sigma_{3p}$ be the suffix of σ_3 such that $p \notin \sigma_{3p}$; and $\sigma'_{3p}p$ be the prefix of σ_3 such that $p \notin \sigma'_{3p}$. Then $\langle p\sigma_{1p}t', p\sigma_{3p}t'\rangle$ is a cross-free diamond with entry path $\mathsf{simple}(\sigma t\sigma'_{3p})$. By Definition 2.3(D2), $t' \in T$ implies $p \in T$. Contradiction.

Then p is not the maximal element of $S_{3,1}$. Then there is σ'_3 containing t', ending at q and such that $\delta' = \langle t\sigma'_3 y, t\sigma_2 y\rangle$ is a cross-free diamond. □

Lemma C.4 shows that if a diamond has some entry path then it has also a simple entry path. The next result shows that any two cross-free diamonds between the same two nodes have the same simple entry paths.

Lemma E.12. *Let $\delta = \langle t\sigma_1 y, t\sigma_2 y\rangle$ be a cross-free diamond with a simple entry path σ in an SG $\boldsymbol{G} = \langle P, T, F, f, g\rangle$, where $t \in T$. Then σ is an entry path of any cross-free diamond δ' of the form $\delta' = \langle t\sigma'_1 y, t\sigma_2 y\rangle$, for some σ'_1.*

Proof. Let σ'_1 be such that $\delta' = \langle t\sigma'_1 y, t\sigma_2 y\rangle$ is a cross-free diamond. We assume that $\sigma'_1 \cap \sigma \neq \emptyset$, and show that this leads to a contradiction.

Let $\sigma_{tx}x$ be the unique prefix of σ'_1 such that $x \in \sigma$ and $\sigma_{tx} \cap \sigma = \emptyset$ (i.e. x is the first node of σ'_1 which is also contained in σ). Then σ has the form $\sigma = \sigma_x x \sigma_{xt}$, for some σ_x and σ_{xt}. Then $\varphi_x = x\sigma_{xt}t \cdot \mathsf{simple}(\sigma_{tx}) \cdot x$ is a cycle with entry path $\sigma_x x$. By Definition 2.3(C1), $x \in P$.

By Definition 2.3(R3), there is a terminal place $p_T \in Term(\boldsymbol{G})$ such that $(y, p_T) \in F^*$. Then let σ_T be a simple path from y to p_T, and consider the path $t\sigma_2 y\sigma_T$, from t to p_T. Since $t \in \varphi_x$, if $\sigma_2 y\sigma_T \cap \varphi_x = \emptyset$ then t is an exit node for φ_x, which in turn implies $t \in P$, by Definition 2.3(C2). But that is a contradiction. Then $\sigma_2 y\sigma_T \cap \varphi_x \neq \emptyset$. But $\sigma \cap \sigma_2 y = \emptyset$, since δ is a cross-free diamond with entry path σ; and $t\sigma'_1 \cap \sigma_2 y = \emptyset$, since δ' is a diamond. Then $\varphi_t \cap \sigma_2 y = \emptyset$, which implies $\varphi_t \cap \sigma_T \neq \emptyset$. But $\sigma'_1 \cap \sigma_T = \emptyset$, by Lemma E.9. Then $\sigma_{tx}x \cap \sigma_T = \emptyset$, which implies $\sigma_{xt} \cap \sigma_T \neq \emptyset$. Then $\sigma \cap \sigma_T \neq \emptyset$, since σ_{xt} is a suffix of σ.

Let $\sigma_{yx'}x'$ be the unique prefix of σ_T such that $x' \in \sigma$ and $\sigma_{yx'} \cap \sigma = \emptyset$ (i.e. x' is the first node of σ_T which is also contained in σ). We can apply to x' the same reasoning we applied to x above. Thus we show that $x' \in P$. Now, since $x' \in \sigma$ and σ is a simple path, either $x' \in \sigma_x x$ or $x' \in \sigma_{xt}$. In the former case, let $\sigma_{x'x}$ be the subpath of σ that starts from x' and ends at x. Then we have a diamond $\delta_{tx} = \langle t\sigma_{tx}x, t\sigma_2 y\sigma_{yx'}\sigma_{x'x}\rangle$. In the latter case, let $\sigma_{xx'}$ be the subpath of σ that starts from x and ends at x'. Then we have a diamond $\delta_{tx'} = \langle t\sigma_{tx}\sigma_{xx'}, t\sigma_2 y\sigma_{yx'}x'\rangle$. In both cases we can apply Definition 2.3(D1), and obtain $x \in T$ or $x' \in T$, respectively. Either way, we have a contradiction. □

As a consequence of the above result, Definition 2.3(D3) is further strengthened, as we can now require that the entry path for the derived diamond be the same as the entry path for the original diamond, as follows.

Corollary E.1. *Let $\delta = \langle t\sigma_1 y, t\sigma_2 y\rangle$ be a cross-free diamond with a simple entry path σ in an SG $\boldsymbol{G} = \langle P, T, F, f, g\rangle$, where $t \in T$ is a transition. Then, for any place $p \in \sigma_1$ and transition $t' \in p\bullet$, there is a path σ'_1 such that $t' \in \sigma'_1 y$ and $\langle t\sigma'_1 y, t\sigma_2 y\rangle$ is a cross-free diamond with entry path σ.*

By combining Corollary E.1 with Lemma E.11, we obtain the following result.

Corollary E.2. *Let $\delta = \langle t\sigma_1 y, t\sigma_2 y\rangle$ be a cross-free diamond with a simple entry path σ in an SG $\boldsymbol{G} = \langle P, T, F, f, g\rangle$, where $t \in T$ is a transition and σ_1 ends at a place q. Then, for any place $p \in \sigma_1$ and transition $t' \in p\bullet$, there is a path σ'_1 ending at q such that $t' \in \sigma'_1 y$ and $\langle t\sigma'_1 y, t\sigma_2 y\rangle$ is a cross-free diamond with entry path σ.*

We are now ready to prove Lemma E.10.

Proof of Lemma E.10. We prove the two directions of the result separately.

Right-to-left direction. We assume that the stated conditions hold and we show that they imply Definition 2.3(D3). Let $\delta = \langle t\sigma'_1 p\sigma''_1 t', t\sigma_2 t'\rangle$ be a cross-free diamond from t to t', with entry path σ. Let q be the last node of σ''_1. Then:

$$(t,p) \in F^*_{-\sigma\cup\sigma_2 t'} \ \wedge \ (p,q) \in F^*_{-\sigma t\sigma_2 t'}$$

by definition of cross-free diamond. By Lemma C.4, $\mathsf{simple}(\sigma)$ is also an entry path of δ. Then we can assume, without loss of generality, that σ is a simple path. Let also $t'' \in p\bullet$ such that $t'' \neq t'$ (the case of $t'' = t'$ is immediate). Our assumptions imply that:

$$(t,t'') \in F^*_{-\sigma\cup\sigma_2 t'} \ \wedge \ (t'',q) \in F^*_{-\sigma t\sigma_2 t'}$$

Then let $\sigma_1 = \sigma'_1 p\sigma''_1$ and $\sigma_3 = \sigma'_1 pt''\sigma'_3$, for some σ'_3 ending at q and such that $\sigma'_3 \cap \sigma t\sigma_2 t' = \emptyset$. Let also $S_{i,j} = S^{\sigma,t,t'}_{i,j}$ and $\bar{S}_{i,j} = \bar{S}^{\sigma,t,t'}_{i,j}$ be defined as in (1a) and (1b), for $i,j \in \{1,2,3\}$. Note that $S_{3,1} = \bar{S}_{1,3} = \sigma_3$ (i.e. they both contain all the nodes of σ_3) since both σ_1 and σ_3 end at q. Note also that either $S_{1,2} = \emptyset$ or $\bar{S}_{2,1} = \emptyset$, since δ is a cross-free diamond. In the former case, $S_{3,2} = \emptyset$ since $\bar{S}_{1,3} = \sigma_3$. In the latter case, $\bar{S}_{2,3} = \emptyset$ since $S_{3,1} = \sigma_3$. Then $\langle t\sigma_3 t', t\sigma_2 t'\rangle$ is a cross-free diamond from t to t' (with entry path σ).

Left-to-Right direction: Contradiction hypothesis and its implications. Now we assume that Definition 2.3(D3) holds and show that it implies the stated conditions. Let $\delta = \langle t\sigma_1 t',\ t\sigma_2 t'\rangle$ be a cross-free diamond with a simple entry path σ, where q is the last node of σ_1. Let $p \in P$ be such that:

$$(t,p) \in F^*_{-\sigma\cup\sigma_2 t'} \ \wedge \ (p,q) \in F^*_{-\sigma t\sigma_2 t'}$$

and let $t'' \neq t'$ be a transition such that $(p,t'') \in F$.

The above implies that there is σ_3 of the form $\sigma_3 = \sigma'_3 p \sigma''_3$, where $t\sigma'_3 p$ is a simple path from t to p, $p\sigma''_3$ is a simple path from p to q and $\sigma_3 \cap \sigma t \sigma_2 t' = \emptyset$.

By contradiction, suppose that:

$$\neg((t, t'') \in F^*_{-\sigma \cup \sigma_2 t'} \ \wedge \ (t'', q) \in F^*_{-\sigma t \sigma_2 t'})$$

But $(t, p) \in F^*_{-\sigma \cup \sigma_2 t'}$ and $(p, t'') \in F$ imply $(t, t'') \in F^*_{-\sigma \cup \sigma_2 t'}$. Then we have:

$$(t'', q) \notin F^*_{-\sigma t \sigma_2 t'}$$

The contradiction hypothesis above implies that there is no σ_4 ending at q, containing t'' and such that $\langle t\sigma_4 t', t\sigma_2 t' \rangle$ is a cross-free diamond with entry path σ. By Corollary E.2, we infer that there is no σ_4 ending at q, containing p and such that $\langle t\sigma_4 t', t\sigma_2 t' \rangle$ is a cross-free diamond with entry path σ. In particular, the diamond $\delta' = \langle t\sigma_3 t', t\sigma_2 t' \rangle$ is not cross-free. Then σ_1 is different from σ_3 and $p \notin \sigma_1$, since σ_1 is a path of δ and δ is cross-free.

As done for the other direction, let $S_{i,j} = S^{\sigma,t,t'}_{i,j}$ and $\bar{S}_{i,j} = \bar{S}^{\sigma,t,t'}_{i,j}$ be defined as in (1a) and (1b), for $i, j \in \{1, 2, 3\}$. Since δ' is not a cross-free diamond, both $S_{2,3}$ and $S_{3,2}$ are non-empty. Since both σ_1 and σ_3 end at place q, $S_{3,1}$ contains all the nodes that occur in σ_3 and $S_{1,3}$ contains all the nodes that occur in σ_1 (i.e. q is the maximal element of both). Then $S_{2,1}$ is also non-empty. Then $S_{1,2}$ must be empty, since δ is a cross-free diamond. Note that $\bar{S}_{1,3} \neq \emptyset$, since $S_{1,3} \neq \emptyset$.

Let y be the minimal element of $\bar{S}_{1,3}$. Then there is a cross-free diamond $\delta' = \langle t\sigma_{3y} y, t\sigma_{1y} y \rangle$ with entry path σ, where $\sigma_{3y} y$ is a prefix of σ_3 and σ_{1y} has some non-empty prefix which is also a prefix of σ_1. By Definition 2.3(D1), $t \in T$ implies $y \in T$. Let also σ'_{3y} be the remaining suffix of σ_3, i.e. $\sigma_3 = \sigma_{3y} y \sigma'_{3y}$. Note that $S_{1,2} = \emptyset$ implies that $\langle t\sigma_{1y} y \sigma'_{3y} t', t\sigma_2 t' \rangle$ is a cross-free diamond. As we showed above, there is no σ_4 ending at q, containing p and such that $\langle t\sigma_4 t', t\sigma_2 t' \rangle$ is a cross-free diamond. Then $p \notin y\sigma'_{3y}$, which implies $p \in \sigma_{3y}$.

Left-to-Right direction: Reaching a contradiction. Since δ' is a cross-free diamond and $p \in \sigma_{3y}$, we can apply Corollary E.1. Then there is σ_{4y} containing t'' and such that $\langle t\sigma_{4y} y, t\sigma_{1y} y \rangle$ is a cross-free diamond with entry path σ. By using Lemma E.4, it is easy to show that $t' \notin \sigma_{4y}$. Then:

$$(t'', q) \in F^*_{-\sigma \cup \{t, t'\}}$$

This implies that there is q' such that q' is the first node to appear in σ_3 that satisfies $(t'', t_{q'}) \in F^*_{-\sigma t \sigma_3 t'}$, for some $t_{q'} \in \bullet q'$. We first show that q' is a place, then we contradict that statement.

The choice of t'' and $t_{q'}$ implies that there is a simple path $\sigma_{t'' t_{q'}}$ from t'' to $t_{q'}$ such that $\sigma_{t'' t_{q'}} \cap \sigma t \sigma_3 t' = \emptyset$. If $q' = p$ or the first occurrence of q' appears in σ_3 before the first occurrence of p, there is a cycle $\varphi_{q'} = \mathsf{simple}(\sigma_{3q'p}) \cdot \sigma_{t'' t_{q'}} q'$, where $\sigma_{3q'p}$ is a subpath of σ_3 that starts from q' and ends at p. Note that q' is an entry node of $\varphi_{q'}$, having entry path $\sigma t \sigma_{3q'}$, where $\sigma_{3q'}$ is the prefix of σ_3 that ends at the first occurrence of q'. Then q' is a place, by Definition 2.3(C1). On the other hand, If the first occurrence of q' appears in σ_3 after the first occurrence

of p, then there is a cross-free diamond $\delta_{pq'}$ of the form $\delta_{pq'} = \langle \sigma_{3pq'}, p\sigma_{t''t_{q'}}q' \rangle$, where $\sigma_{3pq'}$ is a subpath of σ_3 that starts from p and ends at q'. In particular, the entry path for such diamond is of the form $\sigma t \sigma_{3p}$, for some prefix σ_{3p} of σ_3. By Definition 2.3(D2), $p \in P$ implies $q' \in P$.

Now we are going to show that q' is in fact a transition. Note that $(t'', q') \in F^*_{-\sigma \cup \{t,t'\}}$, while $(t'', q') \notin F^*_{-\sigma t \sigma_2 t'}$. Then $\sigma_{t''t_{q'}} \cap \sigma_2 \neq \emptyset$, which implies that there is a diamond $\delta_{q'} = \langle t\sigma_{2q'}q', t\sigma_{3q'}q' \rangle$, where $\sigma_{3q'}$ is a prefix of σ_3 and $\sigma_{2q'}$ is a path which has a prefix in common with σ_2. By Definition 2.3(D1), $t \in T$ implies $q' \in T$.

By the above contradiction, we have thus showed that:

$$(t'', q) \in F^*_{-\sigma t \sigma_2 t'}$$

which concludes the proof. □

F Appendix: Session Nets

In this section, we prove a property of session nets that is used for both safety and progress.

We introduce the notion of firing path. Let $\langle \boldsymbol{P}, M_0 \rangle$ be a Petri net and let $\phi : M_0 \xrightarrow{t_1} M_1 \ldots \xrightarrow{t_n} M_n$ be a firing sequence. Then $\sigma = p_0 t_{h(1)} p_1 t_{h(2)} \ldots p_m t$ is a *firing path* (with respect to ϕ) when: p_0 is the initial place and $h : N \to N$ is a monotonic function such that $h(0) = 0$ and $h(m+1) = n+1$; for all $i \in \{0, \ldots, m\}$ and $j \in \{h(i), \ldots, h(i+1)\}$, $M_j(p_i) > 0$. Intuitively, each place in a firing path contains a token until the next transition fires, then the token is removed and a new token is added to the next place. We say that h is the function *associated to* σ.

Lemma F.1. *Let $\langle \boldsymbol{G}, M_0 \rangle$ be an initial session net. For any $\phi : M_0 \xrightarrow{t_0} M_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{n-1}} M$ such that $M \xrightarrow{t}$, there is a firing path σ from the initial place to t.*

Proof. We prove the claim by induction on the length of ϕ.

Base case. Let ϕ be the empty sequence of firings. Hence, $M = M_0$. The path $\sigma = p_I t$ is such that $M(\sigma) = M(p_I) = 1 > 0$.

Inductive case. Let the claim hold for firing sequences of length less than n, and let $\phi' : M_0 \xrightarrow{t_0} M_1 \xrightarrow{t_1} M_1 \ldots \xrightarrow{t_{n-2}} M_{n-1}$. First, suppose that $M_{n-1} \xrightarrow{t}$. By induction hypothesis there is a firing path σ of the form $\sigma = p_I \ldots pt$. Note that $p \in {\bullet}t$. Since t is still enabled in M, we have $M(p) > 0$. Let h' be the monotonic function associated to σ. Then, given the unique $m \geq 0$ such that $h'(m) = n$, let h be a monotonic function such that $h(m) = n + 1$ and $h(i) = h'(i)$, for all $i \in \{0, \ldots, m-1\}$. Then h and σ satisfy the claim for ϕ.

Now suppose that t is not enabled in M_{n-1}. Since t is enabled in M_n, it must be the case that the firing $M_{n-1} \xrightarrow{t_{n-1}} M_n$ adds a token to some previously empty place $p \in {\bullet}t$. Hence, $M_{n-1}(p) = 0$ and $M_n(p) = 1$ By induction

hypothesis, there is a firing path σ' from $p_I = p_0$ to t_{n-1} with associated monotonic function $h' : N \to N$, such that $h'(0) = 0$ and $h'(m) = n$. Then $\sigma = p_0 t_{f'(1)} p_1 t_{f'(2)} \dots p_{m-1} t_{n-1} p t$ is a firing path from $p_I = p_0$ to t, whose associated function $h : N \to N$ is such that $h(m+1) = n+1$ and $h(i) = h'(i)$, for all $i \neq m+1$. □

The following result is immediate, but useful for proving a progress invariant.

Lemma F.2. *Let $\langle \boldsymbol{G}, M \rangle$ be a session net, where $\boldsymbol{G} = \langle P, T, F, f, g \rangle$ and let σ be a path in $\boldsymbol{G}$ such that $M(\sigma) > 0$. Then σ is of the form $\sigma = \sigma' p \sigma''$, for σ', σ'' and $p \in P$, such that $M(p\sigma'') = M(p) > 0$.*

Proof. Let p be the last non-empty place in σ. Then there are σ' and σ'' such that $\sigma = \sigma' p \sigma''$ and $M(p\sigma'') = M(p) > 0$. □

G Appendix: Session Net Safety

In this section, we prove that session nets enjoy the safety property. The core of the proof is to show the following invariant.

Lemma G.1. *Any S-path σ that starts from the initial place and ends at a terminal place in a session net $\boldsymbol{N} = \langle \boldsymbol{G}, M \rangle$ is such that $M(\sigma) \leq 1$.*

Proof. Let $M = M_n$ be reachable from the marking of an initial session net M_0 through a sequence of firings $\phi : M_0 \xrightarrow{t_0} M_1 \xrightarrow{t_1} M_1 \dots \xrightarrow{t_{n-1}} M_n$. We do the proof by induction on n.

Base case. Let $n = 0$. By definition of initial session net (Definition 2.4), $M_0(p_I) = 1$ and $M_0(p) = 0$, for any $p \neq p_I$. Since $p_I \in \sigma$, $M_0(\sigma) = 1$.

Inductive case: induction hypothesis and its implications. Suppose that, for all $i \in \{0, \dots, n-1\}$, $M_i(\sigma) \leq 1$. The firing of $t = t_{n-1}$ only increases the number of tokens of the places in $t\bullet$. Then let $p \in t\bullet \cap \sigma$. By Lemma D.3(1), $t\bullet \cap \sigma = \{p\}$.

Suppose that there is $q \in \bullet t \cap \sigma$. Since t is enabled in M_{n-1}, M_{n-1} assigns 1 to every place in $\bullet t$. Conversely, $M_{n-1}(\sigma) = 1$ implies that M_{n-1} assigns 1 to exactly one place in σ and 0 to all the others. Then we infer that $\bullet t \cap \sigma$ is a singleton set. To resume, we have $\bullet t \cap \sigma = \{q\}$ and $t\bullet \cap \sigma = \{p\}$. Since the firing of t removes a token from q adds one to p, it preserves the invariant $M(\sigma) = M_n(\sigma) = 1$.

Given the above observations, we will henceforth assume the following:

$$\bullet t \cap \sigma = \emptyset \qquad \wedge \qquad t \notin \sigma \qquad \wedge \qquad t\bullet \cap \sigma = \{p\}$$

where $t \notin \sigma$ is implied by $\bullet t \cap \sigma = \emptyset$.

Inductive case: firing path σ'. By Lemma F.1, there is a firing path σ' from p_I to t. Since both σ and σ' start from p_I, $\sigma \cap \sigma' \neq \emptyset$. Let x be the last node occurring in σ which also occurs in σ'. We want to show that $x \in P$. Since $p \in P$, let us

assume $x \neq p$. Let σ_x be the suffix of σ starting from the last occurrence of x and σ_p the prefix of σ ending at the last occurrence of p.

First, suppose that the last occurrence of x in σ appears before the last occurrence of p. Then there is a diamond $\langle \mathsf{simple}(\sigma_{xp}), \mathsf{simple}(\sigma'_{xp})\rangle \in \mathsf{diamond}(x,p)$, where: σ_{xp} is the unique subpath which is both prefix of σ_x and suffix of σ_p; and σ'_{xp} consists of the suffix of σ' which starts from the last occurrence of x, extended by an occurrence of p. By Definition 2.3(D1), $p \in P$ implies $x \in P$.

Now suppose that the last occurrence of x in σ appears after the last occurrence of p. Then there is a cycle $\varphi = \mathsf{simple}(\sigma_{xp})\mathsf{simple}(\sigma_{px})$, where σ_{xp} is the suffix of σ' which starts from the last occurrence of x, and σ_{px} is the subpath of σ which starts from the last occurrence of p and ends at the last occurrence of x. Note that, if $\sigma_x \cap \varphi = \{x\}$ then x is an exit place for φ and, by Definition 2.3(C2), $x \in P$. Then suppose that $\{x\}$ is strictly contained in $\sigma_x \cap \varphi$.

Since x is the last node to occur in σ which also occurs in σ' and σ_x is the suffix of σ which starts from the last occurrence of x, $\sigma_x \cap \sigma' = \{x\}$. Since σ_{xp} is contained in σ', $\sigma_x \cap \sigma_{xp} = \{x\}$. Since $\varphi = \mathsf{simple}(\sigma_{xp})\mathsf{simple}(\sigma_{px})$ and $\{x\}$ is strictly contained in $\sigma_x \cap \varphi$, there must be $y \in \sigma_x \cap \sigma_{px}$ such that $y \neq x$.

Without loss of generality, we can assume that y is the first node to appear in σ_{px} which occurs also in σ_x. Since both σ_{px} and σ_x are subpaths of σ, $|\bullet y \cap \sigma| > 1$. Then since σ is an S-path, $y \in P$. Now, given the diamond $\langle \mathsf{simple}(\sigma_{xy}), \mathsf{simple}(\sigma_{xp})\mathsf{simple}(\sigma_{py})\rangle \in \mathsf{diamond}(x,y)$, where σ_{xy} is the prefix of σ_x ending at the first occurrence of y and σ_{py} is the prefix of σ_{px} ending at the first occurrence of y, we apply Definition 2.3(D1) and we conclude that $x \in P$.

Note that $t \notin x\bullet$, since $x \in \sigma$ and $\bullet t \cap \sigma = \emptyset$. Since σ' is a firing path, there is $i \in \{0, \ldots, n-1\}$ such that: $M_i(x) = 1$ and $M_{i+1}(x) = 0$; and t_i is the successor of x in σ'. By Lemma D.3(2) and since the place that follows t_i in σ' is not in σ, $t_i\bullet \cap \sigma = \emptyset$. Then $M_i(\sigma) = M_i(x) = 1$ implies $M_{i+1}(\sigma) = 0$. If no token is added in σ between M_{i+1} and M_{n-1}, i.e. $M_{n-1}(\sigma) = 0$, we conclude that $M(\sigma) = 1$.

Inductive case: contradiction argument. By contradiction, we assume the following:

$$\exists_{j \in \{i+1,\ldots,n-2\}} \quad M_j(\sigma) = 0 \wedge M_{j+1}(\sigma) = 1$$

Equivalently, we can say that the firing $M_j \xrightarrow{t_j} M_{j+1}$ adds a token to some place q in σ. Since $M_j(\sigma) = 0$ and $M_j \xrightarrow{t_j}$, we infer $\sigma \cap \bullet t_j = \emptyset$. Then $t_j \notin \sigma$.

By Lemma F.1; there is a firing path σ'' from p_I to t_j. The reasoning previously applied to σ' can be applied to σ'': there is $k \in \{0, \ldots, j-1\}$ such that t_k removes a token from a place $y \in P$, where y is the last node of σ to also occur in σ''.

Note that k may not be greater than i, since $M_k(y) > 0$. Suppose $k = i$. Then $t_k \in \sigma' \cap \sigma''$. The paths σ' and σ'' have different endings, therefore they must split at some node. But they may not split at a place, since they are both firing paths w.r.t. the same firing sequence ϕ. Then let $t' \in T$ be the last node of σ' which is also in σ''. If q appears in σ before p, we have a diamond from t' to p consisting of: the suffix of σ'' starting at t', concatenated with the sub-path of σ that goes from q to p; and the suffix of σ' following t'. By Definition 2.3(D1), $t' \in T$ implies $p \in T$, which is false. On the other hand, if q appears in σ after

p, then the diamond is between t' and q, which leads to a contradiction as well. Then $k < i$. This also implies that no pair of transitions occurring one in σ' and the other in σ'' are associated to the same firing in ϕ after M_i. Formally, let h' and h'' be the monotonic functions associated to the firing paths σ' and σ'', respectively. Then, for all $l \in \{i, \ldots, n-1\}$, if l is in the range of h' it is not in the range of h'' and viceversa.

Since $M_{k+1}(\sigma) = 0$ and $M_i(\sigma) = 1$, there must be a firing which adds a token in σ between M_{k+1} and M_i. Since σ' is a firing path, there must be a firing of a transition t' which adds a token in σ between M_{k+1} and M_i, such that t' occurs in σ' before x. Then we can apply the same reasoning again. As shown above, we apply it in alternation between σ' and σ''. By the finiteness of ϕ, we infer that the invariant stated above applies to all its transitions. Formally, for all $l \in \{1, \ldots, n-1\}$, if l is in the range of h' it is not in the range of h'' and viceversa. But since σ' and σ'' are firing paths both starting from p_I, we necessarily have $h'(1) = h''(1) = 1$. Contradiction. □

With the above invariant, the proof of safety is straightforward.

Proof of Theorem 2.1. Let $\langle \boldsymbol{G}, M \rangle$ be a session net, where $\boldsymbol{G} = \langle P, T, F, f, g \rangle$ and p_I is the initial place of $\boldsymbol{G}$. Let $p \in P$ be an arbitrary place. It follows from Definition 2.4 (of session net) that every marking reachable from M is also the marking of a session net. Then we just have to show that $M(p) \leq 1$.

By Definition 2.3(R3), we have $(p_I, p) \in F^*$. Then there is a simple path σ from p_I to p. By Lemma E.6, there are $p_T \in Term(\boldsymbol{G})$ and σ' such that $\sigma\sigma'$ is an S-path from p_I to p_T. Since it contains p, we have $M(p) \leq M(\sigma\sigma')$. By Lemma G.1, $M(\sigma\sigma') \leq 1$. Then $M(p) \leq 1$. □

H Appendix: Session Net Progress

In this section, we prove the progress property through a series of incremental results. Almost as a corollary of the proof, the no dead transition property of workflow nets follows (Theorem H.1).

The following invariant allows us to show that, whenever a transition has a non-empty input place, there is a token "on its way" towards each other input place of the transition.

Lemma H.1. *Let $\langle \boldsymbol{G}, M_0 \rangle$ be an initial session net, where $\boldsymbol{G} = \langle P, T, F, f, g \rangle$ and $p_I \in Init_{\boldsymbol{G}}(P)$ is the initial place. Let $\phi : M_0 \xrightarrow{t_1} M_1 \ldots \xrightarrow{t_n} M_n = M$ and let $t \in T$ be a transition such that there is a firing path σ from p_I to $t \in T$ with respect to ϕ and with associated function h. Let $q \in {\bullet}t$ be such that $M(q) = 0$. Then for some i in the domain of h, some suffix σ' of σ and for all $j \in \{h(i), \ldots, n\}$, there is $\sigma_j = \sigma'_j q_j \sigma''_j$ such that: $\delta_j = \langle \sigma_j,\ \sigma' \rangle$ is a cross-free diamond from $t_{h(i)}$ to t; $q_j\sigma''_j$ is a simple path with (q, t) as last arc; and $M_j(q_j\sigma''_j) = M_j(q_j) > 0$.*

Proof. By Lemma E.8, there is a cross-free diamond $\delta = \langle \sigma_?, \sigma' \rangle$ from some node $x \in \sigma$ to t such that $\sigma_?$ is a simple path with (q, t) as last arc and σ' is a suffix

of σ. By Definition 2.3(D2), $x \in T$. Then $x = t_{h(i)}$, for some i in the domain of h. Given the frequent usage of $x = t_{h(i)}$, we use the two names interchangeably. Let also $\sigma_{h(i)} = \sigma_?$ and $\delta_{h(i)} = \delta$. We do the proof by induction on $j \in \{h(i), \ldots, n\}$.

Base case. Let $j = h(i)$. Note that $\sigma_j = \sigma_{h(i)}$ is of the form $\sigma_j = xq_j\sigma''_j$, for some $q_j \in x\bullet$ and some simple path σ''_j. The firing $M_{j-1} \xrightarrow{t_j} M_j$ adds a token to q_j. Then $M_j(q_j) > 0$. Let σ'' be an entry path of δ_j, which we can assume to be simple by Lemma C.4. Then the path $\sigma''\sigma_j$ is simple, hence it is also an S-path. By Lemma G.1, $M_j(\sigma'\sigma_j) \leq 1$, which implies $M(q_j\sigma''_j) = M(q_j)$.

Inductive case. Let $j \in \{h(i)+1, \ldots, n\}$. As inductive hypothesis, we assume there is $\sigma_{j-1} = \sigma'_{j-1}q_{j-1}\sigma''_{j-1}$ with (q, t) as last arc and such that $\delta_{j-1} = \langle \sigma_{j-1},\ \sigma' \rangle$ is a cross-free diamond from x to t, $q_{j-1}\sigma''_{j-1}$ is a simple path and $M_{j-1}(q_{j-1}\sigma''_{j-1}) = M_{j-1}(q_{j-1}) > 0$. We are going to show that the same holds for index j.

In the case of $M_j(q_{j-1}) > 0$, we set $\sigma_j = \sigma_{j-1}$. Hence, $M(\sigma_j) > 0$. The existence of q_j, σ'_j and σ''_j that satisfy the conditions follows from Lemma F.2.

Below, we assume that $M_j(q_{j-1}) = 0$. Note that $M_{j-1}(q_{j-1}) = 1$ implies $t_j \in q_{j-1}\bullet$ (recall that t_j is the transition firing at M_{j-1}, leading to M_j). There are two main cases to consider: first we do the proof for the case of $q_{j-1} \neq q$; then we show that $q_{j-1} = q$ leads to a contradiction.

Inductive case - first sub-case: $q_{j-1} \neq q$. Lemma E.11 implies that there is σ_j with last arc (q, t) such that t_j occurs in σ_j and $\delta_j = \langle \sigma_j,\ \sigma' \rangle$ is a cross-free diamond from x to t. Then $t_j\bullet \cap \sigma_j \neq \emptyset$, which in turn implies $M_j(\sigma_j) > 0$. The existence of q_j, σ'_j and σ''_j that satisfy the conditions follows from Lemma F.2.

Inductive case - second sub-case: $q_{j-1} = q$. We are going to show that $q_{j-1} = q$ leads to a contradiction. By Proposition 2.1, $|\bullet t| > 1$ implies $q\bullet = p\bullet = \{t\}$. Then $t = t_j$. By Lemma D.2, $|\bullet t| > 1$ implies $t\bullet = \{p_t\}$, for some $p_t \in P$. By Lemma E.3, $|\bullet t| > 1$ implies $p_t \notin \bullet t$, then $p_t \neq p$.

Recall that $\sigma' = t_{h(i)}p_i \ldots t_{h(m)}p_m t_{h(m+1)}pt$, where $t = t_j$ only occurs once in σ'. Then $j \neq h(k)$, for all $k \in \{i, \ldots, m+1\}$ (i.e. the firing $M_{j-1} \xrightarrow{t_j} M_j$ does not correspond to any transition occurrence in σ'). The firing $M_{j-1} \xrightarrow{t_j} M_j$ removes a token from p and only adds a token to p_t. Then $M_{j-1}(p) > 0$, which implies that $M_{j-1}(p) = 1$ (Corollary 2.1) and in turn that $M_j(p) = 0$, since $p \neq p_t$. Then $j < n$, since $M_n(p) > 0$. Together with the inductive assumption, $j > h(i)$, this implies that there is $l \in \{i, \ldots, m\}$ such that $h(l) < j \leq h(l+1)$. Hence, the place $p_l \in \sigma'$ is such that $M_{j-1}(p_l) > 0$. Since t appears only once in σ' and $p\bullet = \{t\}$, p may not appear more than once either. Then p and p_l are distinct and such that both $M_{j-1}(p) > 0$ and $M_{j-1}(p_l) > 0$.

We are going to show that there is an S-path from the initial place p_I which contains both p_l and p. Let σ'' be an entry path for δ_{j-1}, which we can assume to be simple by Lemma C.4. By definition of entry path, $\sigma' \cap \sigma'' = \emptyset$. Then by Lemma C.1 and Corollary C.1, $\sigma'' \cdot \mathsf{simple}(\sigma')$ is a simple path from p_I to t (hence it is also an S-path). Since t only occurs as last node of σ' and since p is the place preceding such occurrence, Lemma C.3 and C.1 imply that (p, t) is the last arc of $\mathsf{simple}(\sigma')$.

If $p_l \in \mathsf{simple}(\sigma')$, we have found our S-path. Suppose that $p_l \notin \mathsf{simple}(\sigma')$. Then σ' is of the form $\sigma' = \sigma_{xy} y \sigma_y y \sigma_{yt}$, for some node y and paths σ_{xy}, σ_y and σ_{yt}, such that $\mathsf{simple}(\sigma') = \mathsf{simple}(\sigma_{xy}) \cdot y \cdot \mathsf{simple}(\sigma_{yt})$ and $p_l \in \sigma_y$. It follows from the definition of $\mathsf{simple}(-)$ that $\sigma_y \cap \sigma_{xy} = \emptyset$, while $\sigma' \cap \sigma'' = \emptyset$ implies $\sigma_y \cap \sigma'' = \emptyset$. Then y is an entry node for the cycle $\mathsf{simple}(y\sigma_y) \cdot y$. By Definition 2.3(C1), $y \in P$. Then the path $\sigma'' \sigma_{xy} \cdot \mathsf{simple}(y\sigma_y) \cdot y$ is an S-path.

Let z be the last node of $\mathsf{simple}(\sigma')$ which occurs also in $\mathsf{simple}(y\sigma_y)$ and σ_{yz} be the prefix of $\mathsf{simple}(y\sigma_y)$ ending at z. Let also $\sigma_{yt} = \sigma_{1yz}\sigma_{zt}$, where σ_{1yz} is the prefix that ends at z. We are going to show that z is a place, which implies that the path $\sigma'' \sigma_{xy} \cdot \mathsf{simple}(y\sigma_y) \cdot \sigma_{yz}\sigma_{zt}$ is an S-path.

Since z is the last node of $\mathsf{simple}(\sigma')$ which occurs also in $\mathsf{simple}(y\sigma_y)$, $\sigma_{zt} \cap \mathsf{simple}(y\sigma_y) = \emptyset$. By Definition 2.3(R3), there is $p_T \in Term_{\boldsymbol{G}}(P)$ such that $(t, p_T) \in F^*$. Since $t\bullet = \{p_t\}$, $(p_t, p_T) \in F^*$ as well. Then let σ_t be a simple path from p_t to p_T. By Lemma E.9, $\sigma_t \cap \sigma' = \emptyset$. Then $\sigma_t \cap \mathsf{simple}(y\sigma_y) = \emptyset$. This implies that z is an exit node for the cycle $\mathsf{simple}(y\sigma_y) \cdot y$. By Definition 2.3(C2), $z \in P$. Then the path $\sigma'' \sigma_{xy} \cdot \mathsf{simple}(y\sigma_y) \cdot \sigma_{yz}\sigma_{zt}$ is an S-path.

Since $p\bullet = \{t\}$ and $|z\bullet| > 1$, $p \neq z$. Since pt is a suffix of $\mathsf{simple}(\sigma')$, while σ_{zt} is the suffix of σ' that follows z, the former must be a suffix of the latter. This implies $p \in \sigma'' \sigma_{xy} \cdot \mathsf{simple}(y\sigma_y) \cdot \sigma_{yz}\sigma_{zt}$. It remains to see whether $p_l \in \sigma'' \sigma_{xy} \cdot \mathsf{simple}(y\sigma_y) \cdot \sigma_{yz}\sigma_{zt}$, that is whether $p_l \in \mathsf{simple}(y\sigma_y)$. If not, we can apply the same expansion technique recursively until we find a path that does contain p_l. Note that the number of recursive expansions is bounded by the length of σ_y. Then we eventually obtain an S-path σ''' from p_I to t such that $p \in \sigma'''$ and $p_l \in \sigma'''$. Since $M_{j-1}(p_l) > 0$, $M_{j-1}(p) > 0$ and $p_l \neq p$, we have $M_{j-1}(\sigma''') > 1$. But Lemma G.1 implies $M_{j-1}(\sigma''') \leq 1$. Contradiction. □

The following weakens the above invariant, extracting only what is useful for the progress proof.

Corollary H.1. *Let $\langle \boldsymbol{G}, M \rangle$ be a session net, where $\boldsymbol{G} = \langle P, T, F, f, g \rangle$. Let $t \in T$ and $p, q \in \bullet t$, such that $M(p) > 0$ and $M(q) = 0$. Then there are two paths σ_p and σ_q, ending respectively at p and at q, such that: $\langle \sigma_p t, \sigma_q t \rangle$ is a cross-free diamond; there is a suffix $q'\sigma'_q$ of σ_q, where $M(q'\sigma'_q) = M(q') > 0$.*

The following result shows that any transition with a non-empty input place is contained in a simple path from an enabled transition to a terminal place. Informally, this means that some token can move forward towards our target transition.

Lemma H.2. *Let $\langle \boldsymbol{G}, M \rangle$ be a session net, where $\boldsymbol{G} = \langle P, T, F, f, g \rangle$. Let $t \in T$ and $p, q \in \bullet t$, such that $M(p) > 0$ and $M(q) = 0$. Let also σ_T be a simple path from t to some terminal place p_T. Then there are a transition t' and a path σ' such that $M \xrightarrow{t'}$ and $\sigma' q \sigma_T$ is a simple path from t' to p_T.*

Proof. Let $t_0 = t$, $p_0 = p$, $q_0 = q$ and $\sigma_{t0} = \sigma_T$. By Corollary H.1, there are two paths σ_{p0} and σ_{q0}, and a suffix $p_1\sigma'_0$ of σ_{q0} such that: $\langle \sigma_{p0} p_0 t_0, \sigma_{q0} q_0 t_0 \rangle$ is a

cross-free diamond; and $M(p_1\sigma'_0) = M(p_1) > 0$. By Lemma E.9, $\sigma_{t0} \cap \sigma_{q0}q_0 = \emptyset$. Then $\sigma_{t1} = \sigma'_0 q \sigma_T$ is a simple path from some transition $t_1 \in p_1\bullet$ to p_T.

If there is $q_1 \in \bullet t_1$ such that $M(q_1) = 0$, we apply the same reasoning again. Suppose that we have applied the above reasoning $n-1$ times (for $n > 0$). Then we have a transition $t_n \in T$, a place $p_n \in \bullet t_n$ and a path σ'_n such that $\sigma_{tn} = \sigma'_n q \sigma_T$ is a simple path from t_n to p_T and $M(p_n) > 0$. Note also that, for all $i \in \{0, \ldots, n-1\}$, $\sigma_{t(i+1)}$ is longer than σ_{ti}.

Since σ_{tn} is a simple path, the number of times this reasoning can be applied is bounded by $|P \cup T|$, the number of nodes in $GStruct$. Therefore, we eventually find a transition $t' \in T$ and a path σ' such that $\sigma' q \sigma_T$ is a simple path from t' to p_T and $M \xrightarrow{t'}$. □

We now show that, whenever a transition has a non-empty input place, we can reach a marking where that transition is enabled. The proof is by induction on the length of the longest path of the kind shown in the previous result.

Lemma H.3. *Let $\langle \boldsymbol{G}, M\rangle$ be a session net and $\boldsymbol{G} = \langle P, T, F, f, g\rangle$. Let $t \in T$ and $p \in \bullet t$, such that $M(p) > 0$. Then some M' is reachable from M such that $M' \xrightarrow{t}$.*

Proof. By Definition 2.3(R3), there are a terminal place $p_T \in Term_{\boldsymbol{G}}(P)$ and a simple path σ_T from t to p_T. Let us show that there are also a transition $t' \in T$ and a path σ such that $M \xrightarrow{t'}$ and $\sigma\sigma_T$ is a path from t' to p_T. In the case of $M \xrightarrow{t}$, this is satisfied by $\sigma = \epsilon$. Otherwise, there is $q \in \bullet t$ such that $M(q) = 0$. Then the claim follows from Lemma H.2.

Then let S_M be the set of such paths (indexed by the marking M):

$$S_M = \{\sigma \mid \exists_{t' \in T}(M \xrightarrow{t'} \wedge \sigma\sigma_T \text{ is a simple path from } t' \text{ to } p_T)\}$$

Since they are all paths from some transition to some place, their lengths are always even. Then let $2n \geq 0$ be the length of the longest path in S, and let $S_{M,i}$ be the set of paths of S_M that have length $2i$, for any $i \in \{0, \ldots, n\}$. Note that $S_{M,0} \neq \emptyset$, if and only if $M \xrightarrow{t}$.

Now, suppose that there is $\sigma \in S_{M,i}$, for some $i > 0$, and let t' be the first transition of σ. By definition of $S_{M,i}$, t' is enabled at M. Then the firing $M \xrightarrow{t'} M'$ moves a token closer to t. Let us be more specific. Let $S_{M'}$ and $S_{M',j}$ be defined as S_M and $S_{M,j}$, respectively, for all $j \in \{0, \ldots, n\}$. Then $|S_{M',i}| = |S_{M,i}| - 1$, $|S_{M',i-1}| = |S_{M,i-1}| + 1$ and $|S_{M',j}| = |S_{M,j}|$, for all $j \in \{0, \ldots, i-2, i+1, \ldots\}$. Hence, $(|S_{M',n}|, \ldots, |S_{M',0}|) \leq_{lex} (|S_{M,n}|, \ldots, |S_{M,0}|)$, where $\leq_{lex}$ is the lexicographical order on tuples of natural numbers.

We have shown that, as long as the length of the longest path is greater than 0, a transition can fire that decreases the order. After a finite number of firings, we reach a marking M'' such that $S_{M'',0} \neq \emptyset$, which implies that $M'' \xrightarrow{t}$. □

The following result shows that if a terminal place is reachable from a non-empty non-terminal place, then that terminal place is empty.

Lemma H.4. *Let $\langle \boldsymbol{G}, M\rangle$ be a session net, where $\boldsymbol{G} = \langle P, T, F, f, g\rangle$. Let $p \in P \setminus Term(\boldsymbol{G})$ be a non-terminal place such that $M(p) > 0$. Then there is a terminal place $p_T \in Term(\boldsymbol{G})$ such that $(p, p_T) \in F^*$ and $M(p_T) = 0$.*

Proof. By Definition 2.3(R3), there exists $p_T \in Term_{\boldsymbol{G}}(P)$ such that $(p, p_T) \in F^*$. By contradiction, suppose that $M(p_T) > 0$. Since $(p, p_T) \in F^*$, there is a simple path $pt_1p_1 \dots t_np_T$ from p to p_T. By repeatedly applying Lemma H.3, we obtain a sequence of firings of the form $M \xrightarrow{s_1} M_1 \xrightarrow{t_1} M'_1 \dots M_{n-1} \xrightarrow{s_n} M_n \xrightarrow{t_n} M'_n$. By definition of terminal place, $p_T\bullet = \emptyset$. This implies the number of tokens in p_T never decreases. Then $M'_n(p_T) \geq M_n(p_T) \geq M(p_T) > 0$. Since the firing of t_n adds a token to p_T, we infer $M'_n(p_T) > M_n(p_T)$. Hence, $M'_n(p_T) > 1$. Now, let p_I be the initial place of $\boldsymbol{G}$. By Definition 2.3(R3), we have $(p_I, p_T) \in F^*$. Then there is a simple path σ from p_I to p_T. By Lemma E.6, σ is an S-path. Then, by Lemma G.1, $M'_n(\sigma) \leq 1$. But $M'_n(p_T) > 1$ implies $M'_n(\sigma) > 1$. Contradiction. □

And now the proof of progress.

Proof of Theorem 2.2. Let $\langle \boldsymbol{G}, M\rangle$ be a session net, where $\boldsymbol{G} = \langle P, T, F, f, g\rangle$. Let $EmptyT(\boldsymbol{G}, M) = \{p_T \in Term(\boldsymbol{G}) \mid M(p_T) = 0\}$ be the set of terminal places of $\boldsymbol{G}$ which are empty at M. Suppose that there is a non-terminal place $p \in P$ such that $M(p) > 0$. By Lemma H.4, there exists $p_T \in Term_{\boldsymbol{G}}(P)$ such that $(p, p_T) \in F^*$ and $M(p_T) = 0$, which in turn implies $EmptyT(\boldsymbol{G}, M) \neq \emptyset$.

Let $pt_1p_1 \dots t_np_T$ be a simple path from p to p_T. By repeatedly applying Lemma H.3, we obtain a sequence of firings of the form $M \xrightarrow{s_1}\xrightarrow{t_1} \dots \xrightarrow{s_n}\xrightarrow{t_n} M'$. Since the firing of t_n adds a token to p_T, we infer $M'(p_T) > 0$. Moreover, the number of tokens in any $p'_T \in Term(\boldsymbol{G})$ never decreases, since $p'_T\bullet = \emptyset$, by definition of terminal place. Hence, $EmptyT(\boldsymbol{G}, M')$ is strictly contained in $EmptyT(\boldsymbol{G}, M)$. Then, by applying the above reasoning a finite number of times (at most $|Term(\boldsymbol{G})|$), we reach a marking M'' such that $EmptyT(\boldsymbol{G}, M'') = \emptyset$. By Lemma H.4, this implies $M''(q) = 0$, for any $q \in P \setminus Term(\boldsymbol{G})$. Hence, M'' is a terminal marking. □

From the progress proof above and a basic property of well-formed SGs, it follows that no transition is dead from the initial marking.

Theorem H.1 (No dead transitions). *Let $\langle \boldsymbol{G}, M_0\rangle$ be an initial session net, where $\boldsymbol{G} = \langle P, T, F, f, g\rangle$. For every transition $t \in T$, there is a marking M reachable from M_0 such that $M \xrightarrow{t}$.*

Proof. Let p_I be the initial place of $\boldsymbol{G}$. By Definition 2.3(R3), $(p_I, t) \in F^*$. Then there is a simple path of the form $p_It_1p_2 \dots p_nt$. Since $M_0(p_I) > 0$, we repeatedly apply Lemma H.3 and obtain the following sequence of firings: $M_0 \xrightarrow{s_1}\xrightarrow{t_1} \dots \xrightarrow{s_n} M \xrightarrow{t}$. □

I Appendix: Conformance

Proof of Proposition 3.1 The syntax of endpoint types contains no parallel construct, then the set of derivatives of $T_{\mathtt{r}}$ is finite. The set of markings reachable

from the initial marking M_0 of $\boldsymbol{G}$ is also finite, by Corollary 2.1. The closure of $\{T_{\mathtt{r}}, \langle \boldsymbol{G}, M_0 \rangle\}$ under the rules of Definition 3.1 is a subset of the cartesian product of those two sets, hence it is also finite. Then we can find such a closure (or a violation of the rules) in finite time by playing out each possible derivation that conformance allows and then backtracking to the closest unexplored branch. □

The following result establishes an invariant between a transition sequence from an initial configuration and its weak simulation from an initial session net, where conformance is preserved at each step. The invariant associates a message in a configuration buffer to a token inside a related communication place.

Lemma I.1. *Let $\langle \boldsymbol{G}, M_0 \rangle$ be an initial session net and C_0 an initial configuration with set of roles R. Let $\langle \boldsymbol{G}, M_0 \rangle \xrightarrow{\tau^*} \xrightarrow{m_1} \ldots \xrightarrow{\tau^*} \xrightarrow{m_n} \langle \boldsymbol{G}, M_n = M \rangle$ and $C_0 \xrightarrow{m_1} \ldots \xrightarrow{m_n} C_n = C = (\vec{T}, \vec{w})$ such that $C_i = (\vec{T_i}, \vec{w_i})$ and $T_{i\mathtt{r}} \asymp \langle \boldsymbol{G}, M_i \rangle$, for all $i \in \{0, \ldots, n\}$ and $\mathtt{r} \in R$. Given $\mathtt{r}, \mathtt{r}' \in R$, $w_{\mathtt{r}'\mathtt{r}} = w' \cdot a \cdot w''$ if and only if:*

$$\exists_{p \in P \setminus \mathsf{dom}(f), t \in p\bullet} : \; M(p) = 1 \wedge g(t) = ?a \wedge \mathtt{local}(t) = \mathtt{r} \wedge \mathtt{remote}(t) = \mathtt{r}' \quad (2)$$

Proof. We prove it by induction on n, the length of the transition sequence from C_0 to C_n. Suppose that $n = 0$. The left-to-right direction is immediate, since $\vec{w}_0 = \vec{\epsilon}$. Let $p_I \in P$ be the initial place of $\boldsymbol{G}$. Hence, $M(p_I) = 1$ and $M(p) = 0$, for all $p \in P \setminus \{p_I\}$. Since $\bullet p_I = \emptyset$, p_I is not a communication place (i.e. $p_I \in \mathsf{dom}(f)$) by Definition 2.2. Then the right-to-left direction is also satisfied.

Suppose that $n > 0$ and that C_{n-1} and M_{n-1} satisfy (2). Let $m_n = \mathtt{r}!\mathtt{r}'\langle a \rangle$. Then $w_{n\mathtt{r}\mathtt{r}'} = w_{(n-1)\mathtt{r}\mathtt{r}'} \cdot a$ and $w_{n\mathtt{r}''\mathtt{r}'''} = w_{(n-1)\mathtt{r}''\mathtt{r}'''}$, for all $\mathtt{r}'', \mathtt{r}''' \in R$ such that $\mathtt{r}''\mathtt{r}''' \neq \mathtt{r}\mathtt{r}'$. We are now going to show that the correspondence of (2) is preserved by the transitions in the session net. $\langle \boldsymbol{G}, M_{n-1} \rangle \xrightarrow{\tau^*} \xrightarrow{\mathtt{r}!\mathtt{r}'\langle a \rangle} \langle \boldsymbol{G}, M_n \rangle$ implies that there is a sequence of firings $\langle \boldsymbol{G}, M_{n-1} \rangle \xrightarrow{st} \langle \boldsymbol{G}, M_n \rangle$, such that $g(t) = !a$, $\mathtt{local}(t) = \mathtt{r}$, $\mathtt{remote}(t) = \mathtt{r}'$ and $t' \notin \mathsf{dom}(g)$, for all $t' \in s$. By Definition 2.3(L2), there is $p \in P \setminus \mathsf{dom}(f)$ such that either $(p, t) \in F$ or $(t, p) \in F$. By Definition 2.2 and because t is an output transition, we discard $(p, t) \in F$, hence $p \in t\bullet$. Note that $p \in t\bullet$ is a communication place exactly as requested by (2). The firing of t adds a token to p. Then $M_n(p) > 0$. By Corollary 2.1, $M_n(p) = 1$. As we stated above, for all $t' \in s$, $t' \notin \mathsf{dom}(g)$. Then $M_n(q) = M_{n-1}(q)$, for every communication place $q \neq p$. Then (2) is satisfied.

The case of $m_n = \mathtt{r}?\mathtt{r}'\langle a \rangle$ follows a dual proof . □

The following result says that if we can weakly do some output in the projected LTS of a role $\mathtt{r}$, then we can do the same output after firing only internal transitions contained in a role structure for $\mathtt{r}$, while also preserving conformance.

Lemma I.2. *Let $T_{\mathtt{r}} \xrightarrow{\mathtt{r}!\mathtt{r}'\langle a \rangle} T'_{\mathtt{r}}$ and $\langle \boldsymbol{G}, M \rangle \Rightarrow \xrightarrow{\mathtt{r}!\mathtt{r}'\langle a \rangle} \langle \boldsymbol{G}, M' \rangle$ such that $T'_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M' \rangle$. Then there are transitions $t_1, \ldots, t_n, t$ and a marking M'' in $\boldsymbol{G}$ such that: $\langle \boldsymbol{G}, M \rangle \xrightarrow{t_1} \ldots \xrightarrow{t_n} \xrightarrow{t} \langle \boldsymbol{G}, M'' \rangle$ and $T'_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M'' \rangle$; $g(t) = !a$, $t_i \notin \mathsf{dom}(g)$ and $f(p_i) = \mathtt{r}$, for all $i \in \{1, \ldots, n\}$ and $p_i \in \bullet t_i \cup t_i\bullet$.*

Proof. We just roll back every transition in the original weak sequence that is not contained in a RS for $\mathtt{r}$. This can be done safely because such transitions do no affect the firing of t at all. □

Using the above results, we prove soundness of the conformance relation.

Proof of Theorem 3.1. We do the proof by induction on n. The base case is immediate. Let $n > 0$ and suppose that $\langle \boldsymbol{G}, M_0\rangle \xrightarrow{\tau^*} \xrightarrow{m_1} \langle \boldsymbol{G}, M_1\rangle \dots \xrightarrow{\tau^*} \xrightarrow{m_{n-1}} \langle \boldsymbol{G}, M_{n-1}\rangle$ and $T_{i\mathtt{r}} \asymp \langle \boldsymbol{G}, M_i\rangle$, for all $i \in \{0, \dots, n-1\}$ and $\mathtt{r} \in R$.

Let $m_n = \mathtt{r}!\mathtt{r}'\langle a\rangle$, for some roles $\mathtt{r}, \mathtt{r}' \in R$ and action a. Then $T_{(n-1)\mathtt{r}} \xrightarrow{m_n} T_{n\mathtt{r}}$. By Definition 3.1 (1a), $\langle \boldsymbol{G}, M_{n-1}\rangle \Rightarrow \xrightarrow{m_n} \langle \boldsymbol{G}, M_n\rangle$ and $T_{n\mathtt{r}} \asymp \langle \boldsymbol{G}, M_n\rangle$. By Lemma I.2, we can assume that the above transition sequence corresponds to some firing sequence $\langle \boldsymbol{G}, M_{n-1}\rangle \xrightarrow{t_1} \dots \xrightarrow{t_j} \xrightarrow{t} \langle \boldsymbol{G}, M_n\rangle$, where $g(t) = !a$, $t_i \notin \mathsf{dom}(g)$ and $f(p_i) = \mathtt{r}$, for all $i \in \{1, \dots, j\}$ and $p_i \in {\bullet t_i} \cup t_i{\bullet}$.

We are going to show that, for all $\mathtt{r}' \in R \setminus \{\mathtt{r}\}$, $T_{n\mathtt{r}'} = T_{(n-1)\mathtt{r}'} \asymp \langle \boldsymbol{G}, M_n\rangle$. Note that, for all m such that $\langle \boldsymbol{G}, M_n\rangle \Rightarrow \xrightarrow{m}$ in the projected LTS for $\mathtt{r}'$, $\langle \boldsymbol{G}, M_{n-1}\rangle \Rightarrow \xrightarrow{m}$. Then Definition 3.1 (2) still holds. We check Definition 3.1(1). Let $T_{n\mathtt{r}'} = T_{(n-1)\mathtt{r}'} \xrightarrow{\mathtt{r}'!\mathtt{r}''\langle b\rangle} T'_{\mathtt{r}'}$. By Definition 3.1(1a), $\langle \boldsymbol{G}, M_{n-1}\rangle \Rightarrow \xrightarrow{\mathtt{r}'!\mathtt{r}''\langle b\rangle} \langle \boldsymbol{G}, M'\rangle$ and $T'_{\mathtt{r}'} \asymp \langle \boldsymbol{G}, M'\rangle$. By applying again Lemma I.2 (as done above for $\mathtt{r}$), we can assume that the initial weak sequence only contains firings of transitions inside some RS for $\mathtt{r}'$. Then the same transitions can fire starting from M_n. Hence, $T_{n\mathtt{r}'} \asymp \langle \boldsymbol{G}, M_n\rangle$.

Let $T_{n\mathtt{r}'} = T_{(n-1)\mathtt{r}'}$ be an input. Suppose that $\langle \boldsymbol{G}, M_{n-1}\rangle \Rightarrow \xrightarrow{\mathtt{r}'?\mathtt{r}''\langle b\rangle}$ and that there are no role $\mathtt{r}'''$ and message c such that $\langle \boldsymbol{G}, M_n\rangle \Rightarrow \xrightarrow{\mathtt{r}'?\mathtt{r}'''\langle c\rangle}$. Let i be the highest index in $\{1, \dots, j\}$ such that $\langle \boldsymbol{G}, M_{n-1}\rangle \xrightarrow{t_1} \dots \xrightarrow{t_i} \langle \boldsymbol{G}, M'\rangle$ and $\langle \boldsymbol{G}, M'\rangle \Rightarrow \xrightarrow{\mathtt{r}'?\mathtt{r}'''\langle c\rangle}$, for some marking M', role $\mathtt{r}'''$ and message c. Let also M'' be the "next marking" towards M_n, i.e. $\langle \boldsymbol{G}, M'\rangle \xrightarrow{t_{i+1}} \langle \boldsymbol{G}, M''\rangle$. Our assumption implies that there are no role $\mathtt{r}'''$ and message c, such that $\langle \boldsymbol{G}, M''\rangle \Rightarrow \xrightarrow{\mathtt{r}'?\mathtt{r}'''\langle c\rangle}$. Since $\langle \boldsymbol{G}, M'\rangle \Rightarrow \xrightarrow{\mathtt{r}'?\mathtt{r}''\langle b\rangle}$, there are $p \in {\bullet t_{i+1}}$, $t'_{i+1} \in p{\bullet}$ and $p' \in P$ such that $(t'_{i+1}, p') \in F^*$ and $f(p') = \mathtt{r}'$. By Definition 2.3 (L1), there is $p'' \in P$ such that $(t_{i+1}, p'') \in F^*$ and $f(p'') = \mathtt{r}'$. The latter means that the place p'' is contained in a RS for $\mathtt{r}'$ then, in order to reach p'' from t_{i+1}, we must visit an input of $\mathtt{r}'$, i.e. there are a role $\mathtt{r}'''$ and an action c such that $\langle \boldsymbol{G}, M''\rangle \Rightarrow \xrightarrow{\mathtt{r}'?\mathtt{r}'''\langle c\rangle}$. This contradicts a previous statement. Hence, we conclude that $T_{n\mathtt{r}'} \asymp \langle \boldsymbol{G}, M_n\rangle$.

Now, let $m_n = \mathtt{r}?\mathtt{r}'\langle a\rangle$, for some roles $\mathtt{r}, \mathtt{r}' \in R$ and action a. This implies $w_{(n-1)\mathtt{r}'\mathtt{r}} = a \cdot w_{n\mathtt{r}'\mathtt{r}}$. By Lemma I.1, there is a communication place $p \notin \mathsf{dom}(f)$ with associated input transition $t \in p{\bullet}$, such that $M_{n-1}(p) = 1$, $g(t) = ?a$, $\mathtt{local}(t) = \mathtt{r}$ and $\mathtt{remote}(t) = \mathtt{r}'$. Since t is an input transition, ${\bullet t} = \{p\}$. Then t is enabled in M_{n-1} and its firing corresponds to $\langle \boldsymbol{G}, M_{n-1}\rangle \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a\rangle} \langle \boldsymbol{G}, M_n\rangle$. Thus we can apply Definition 3.1(2b) and since $T_{(n-1)\mathtt{r}}$ is an input such that $T_{(n-1)\mathtt{r}} \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a\rangle} T_{n\mathtt{r}}$, we infer $T_{n\mathtt{r}} \asymp \langle \boldsymbol{G}, M_n\rangle$.

We now show that, for all $\mathtt{r}' \in R \setminus \{\mathtt{r}\}$, $T_{n\mathtt{r}'} = T_{(n-1)\mathtt{r}'} \asymp \langle \boldsymbol{G}, M_n \rangle$. This is straightforward. The preservation of Definition 3.1(2) is the same as for the output case. Definition 3.1(1) follows from the fact that any $t' \neq t$ which is enabled in M_{n-1} is also enabled in M_n. □

The following is a key result for deadlock-freedom and reception error-freedom. It says that if a configuration contains an input, then: if all the buffers are empty, an output action may occur; if not, an input action may occur.

Lemma I.3. *Let $C = (\vec{T}, \vec{w})$ be a configuration with set of roles R and such that $C_0 \xrightarrow{\vec{m}} C$, where C_0 is an initial configuration. Let also $\boldsymbol{G}$ be a well-formed SG such that $C_0 \asymp \boldsymbol{G}$. Let $\mathtt{r} \in R$ such that $T_{\mathtt{r}}$ is an input. Then either:*

- *there are C', $\mathtt{r}' \in R$ and a such that $C \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a \rangle} C'$; or*
- *for all $\mathtt{r}' \in R$, $w_{\mathtt{r}'\mathtt{r}} = \epsilon$; and there exists $\mathtt{r}'' \in R$ such that $T_{\mathtt{r}''}$ is an output.*

Proof. Let M_0 be the initial marking of $\boldsymbol{G}$. By Theorem 3.1, $\langle \boldsymbol{G}, M_0 \rangle \xrightarrow{\tau^*} \xrightarrow{m_1} \ldots \xrightarrow{\tau^*} \xrightarrow{m_n} \langle \boldsymbol{G}, M \rangle$, where $m_1 \ldots m_n = \vec{m}$ and $T_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M \rangle$, for all $\mathtt{r} \in R$.

Suppose that there is $\mathtt{r}' \in R$ such that $w_{\mathtt{r}'\mathtt{r}} \neq \epsilon$ and let $w_{\mathtt{r}'\mathtt{r}} = a \cdot w'$, for some action a. By Lemma I.1, there is a communication place $p \notin \mathsf{dom}(f)$ with associated input transition $t \in p\bullet$, such that $M(p) = 1$, $g(t) = ?a$, $\mathtt{local}(t) = \mathtt{r}$ and $\mathtt{remote}(t) = \mathtt{r}'$. Since t is an input transition, $\bullet t = \{p\}$. Then t is enabled in M and its firing corresponds to $\langle \boldsymbol{G}, M \rangle \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a \rangle} \langle \boldsymbol{G}, M' \rangle$, for some marking M'. By Definition 3.1(2b) and because $T_{\mathtt{r}}$ is an input, we infer $T_{\mathtt{r}} \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a \rangle} T'_{\mathtt{r}}$. Since $w_{\mathtt{r}'\mathtt{r}} = a \cdot w'$, we infer $C \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a \rangle} C'$.

Now suppose that for all $\mathtt{r}' \in R$, $w_{\mathtt{r}'\mathtt{r}} = \epsilon$. By Definition 3.1(1b) and because $T_{\mathtt{r}}$ is an input, $\langle \boldsymbol{G}, M \rangle \Rightarrow \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a \rangle}$ for some role $\mathtt{r}' \in R$ and action a. By definition of the LTS for session nets, this implies that there is a sequence of firings $\phi : \langle \boldsymbol{G}, M \rangle \xrightarrow{s} \langle \boldsymbol{G}, M' \rangle \xrightarrow{t}$, for some $t \in T$ such that $g(t) = ?a$. By Definition 2.3(L2), there is $p \in P \setminus \mathsf{dom}(f)$ such that either $(p, t) \in F$ or $(t, p) \in F$. By Definition 2.2 and because t is an input transition, we discard $(t, p) \in F$, hence $p \in \bullet t$. Since t is enabled in M', $M'(p) > 0$. By Definition 2.2 and because p is a communication place, there is a unique output transition $t' \in T$ such that $\bullet p = \{t'\}$, and $g(t') = !a$. By Lemma I.1, and because $w_{\mathtt{r}'\mathtt{r}} = \epsilon$, we can infer $M(p) = 0$. Then ϕ contains a firing of t', such that $s = s't's''$. If there is no observable transition $t'' \in s'$ such that $\mathtt{local}(t'') = \mathtt{r}'$, then $\langle \boldsymbol{G}, M \rangle \Rightarrow \xrightarrow{\mathtt{r}'!\mathtt{r}\langle a \rangle}$. By Definition 3.1(2a), $T_{\mathtt{r}'}$ is an output. Otherwise, if there is an observable transition $t'' \in s'$ such that $\mathtt{local}(t'') = \mathtt{r}'$, there are two cases: if t'' is an input transition, we apply the same reasoning given previously for t; whereas if t'' is an output transition, we apply the reasoning given for t'. Since $t'' \in s'$, the transition sequence to consider is strictly a prefix of both s' and s. Then by applying the same reasoning repeatedly, we eventually find an output $T_{\mathtt{r}''}$, for some $\mathtt{r}'' \in R$. □

Now we can prove safety of conformance.

Proof of Theorem 3.2 Let $C_0 \xrightarrow{\vec{m}} C$, where $C = (\vec{T}, \vec{w})$. Let M_0 be the initial marking for $\boldsymbol{G}$. By Theorem 3.1, we have $\langle \boldsymbol{G}, M_0 \rangle \xrightarrow{\tau^*} \xrightarrow{m_1} \ldots \xrightarrow{\tau^*} \xrightarrow{m_n} \langle \boldsymbol{G}, M \rangle$, where $m_1 \ldots m_n = \vec{m}$ and and $T_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M \rangle$, for all $\mathtt{r} \in R$.

Deadlock-freedom. By contradiction, let C be a deadlock configuration. Then there is $\mathtt{r} \in R$ such that $T_{\mathtt{r}}$ is an input type. Lemma I.3 implies that there exists $\mathtt{r}' \in R$ such that either: $C \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a \rangle} C'$, for some C' and a; or $T_{\mathtt{r}'}$ is an output. Both cases contradict the assumption that C is a deadlock configuration.

Orphan message-freedom. By contradiction, let C be an orphan message configuration. Then $\vec{w} \neq \emptyset$. This implies that there are roles $\mathtt{r}, \mathtt{r}' \in R$ and an action a such that $w_{\mathtt{r}',\mathtt{r}} = w' \cdot a \cdot w''$, for some w' and w''. By Lemma I.1, there is a communication place $p \notin \mathsf{dom}(f)$ with associated input transition $t \in p\bullet$, such that $M(p) = 1$, $g(t) = ?a$, $\mathtt{local}(t) = \mathtt{r}$ and $\mathtt{remote}(t) = \mathtt{r}'$. Then $\langle \boldsymbol{G}, M \rangle \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a \rangle}$. By Definition 3.1(2b), there is an input $T'_{\mathtt{r}}$ such that $T_{\mathtt{r}} \xrightarrow{\vec{m}} T'_{\mathtt{r}}$ with a sequence of output actions $\vec{m}$. This is a contradiction since C is an orphan message configuration, which implies $T_{\mathtt{r}} = \mathsf{end}$.

Reception error-freedom. By contradiction, suppose that C is an unspecified reception configuration. Let $T_{\mathtt{r}}$ be an input, for some $\mathtt{r} \in R$. Hence $T_{\mathtt{r}} \xrightarrow{\mathtt{r}?\mathtt{r}'\langle a \rangle} T'_{\mathtt{r}}$, for some $T'_{\mathtt{r}}$, $\mathtt{r}' \in R$ and a. Lemma I.3 implies either: $w_{\mathtt{r}''\mathtt{r}} = \epsilon$, for all $\mathtt{r}'' \in R$; or $C \xrightarrow{\mathtt{r}?\mathtt{r}''\langle b \rangle} C'$, for some C', $\mathtt{r}'' \in R$ and b. The former case contradicts the assumption that C is an unspecified reception configuration. The latter case implies $T_{\mathtt{r}} \xrightarrow{\mathtt{r}?\mathtt{r}''\langle b \rangle} T''_{\mathtt{r}}$ and $w_{\mathtt{r}''\mathtt{r}} = b \cdot w'$. Then it also contradicts the initial assumption.

The progress property (2) in this theorem is a corollary of safety. □

J Appendix: Multiparty Asynchronous Session Calculus

This section lists the full definitions and proofs of § 3.

J.1 Reduction rules

We first define the full reduction rules.

$$\lfloor \textsc{Link} \rfloor \quad \overline{a}[\mathtt{r}_1, .., \mathtt{r}_n](x).P_1 \mid a[\mathtt{r}_2](x).P_2 \mid \cdots \mid a[\mathtt{r}_n](x).P_n \longrightarrow (\boldsymbol{\nu} s)(\Pi_{i \in \{1,..,n\}}(P_i[s[\mathtt{r}_i]/x] \mid \Pi_{j \in \{1,..,n\} \setminus i} s[\mathtt{r}_i, \mathtt{r}_j] : \epsilon)) \quad s \notin \mathsf{fn}(P_i)$$

$$\lfloor \textsc{Sel} \rfloor \quad s[\mathtt{r}]!\,\mathtt{r}' : l\langle v \rangle; P \mid s[\mathtt{r}, \mathtt{r}'] : h \longrightarrow P \mid s[\mathtt{r}, \mathtt{r}'] : h \cdot l\langle v \rangle$$

$$\lfloor \textsc{Bra} \rfloor \quad s[\mathtt{r}]?\{\mathtt{r}'_i : l_i(z_i).P_i\}_{i \in J} \mid s[\mathtt{r}'_j, \mathtt{r}] : l_j\langle v \rangle \cdot h \longrightarrow P_j[v/z_j] \mid s[\mathtt{r}'_j, \mathtt{r}] : h$$

$$\lfloor \textsc{Par} \rfloor \quad P \longrightarrow P' \;\Rightarrow\; P \mid Q \longrightarrow P' \mid Q$$

$$\lfloor \textsc{Res} \rfloor \quad P \longrightarrow P' \;\Rightarrow\; (\boldsymbol{\nu} a)P \longrightarrow (\boldsymbol{\nu} a)P'$$

$$\lfloor \textsc{Sres} \rfloor \quad P \longrightarrow P' \;\Rightarrow\; (\boldsymbol{\nu} s)P \longrightarrow (\boldsymbol{\nu} s)P'$$

$$\lfloor \textsc{Struct} \rfloor \quad P \longrightarrow P',\ P' \equiv Q',\ Q' \equiv Q \;\Rightarrow\; P \equiv Q$$

where the structure rules are defined as:

$$P \mid 0 \equiv P \qquad P \mid Q \equiv Q \mid P \qquad (P \mid Q) \mid R \equiv P \mid (Q \mid R)$$

$$(\boldsymbol{\nu} n)P \mid Q \equiv (\boldsymbol{\nu} n)(P \mid Q) \text{ if } n \notin \mathsf{fn}(Q) \quad (\boldsymbol{\nu} n)(\boldsymbol{\nu} n')P \equiv (\boldsymbol{\nu} n')(\boldsymbol{\nu} n)P \quad (\boldsymbol{\nu} n)0 \equiv 0$$

where $n ::= a \mid s$

$$(\boldsymbol{\nu} s)(s[\mathtt{r}_1, \mathtt{r}'_1] : \epsilon \mid .. \mid s[\mathtt{r}_n, \mathtt{r}'_n] : \epsilon) \equiv 0$$

$$\mu X.P \equiv P[\mu X.P/X]$$

J.2 Typing rules

This subsection lists the typing system for processes.

Typing system for programs This paragraph lists a typing system for programs, i.e. processes which do not contain free variables and session channels.

$$[\textsc{Id}] \quad \Gamma, u : \boldsymbol{G} \vdash u : \boldsymbol{G}$$

$$[\textsc{Req}] \frac{T_{\mathtt{r}_1} \asymp \boldsymbol{G} = \langle P, T, F, f, g \rangle \quad \Gamma \vdash u : \boldsymbol{G} \quad \mathtt{range}(f) = \{\mathtt{r}_1, .., \mathtt{r}_n\} \quad \Gamma \vdash Q \triangleright \Delta, x : T_{\mathtt{r}_1}}{\Gamma \vdash \overline{u}[\mathtt{r}_1, .., \mathtt{r}_n](x).Q \triangleright \Delta}$$

$$[\textsc{Acc}] \frac{T_{\mathtt{r}_i} \asymp \boldsymbol{G} \quad \Gamma \vdash u : \boldsymbol{G} \quad \Gamma \vdash Q \triangleright \Delta, x : T_{\mathtt{r}_i} i \neq 1}{\Gamma \vdash u[\mathtt{r}_i](x).Q \triangleright \Delta}$$

$$[\textsc{Sel}] \frac{j \in I \quad \Gamma \vdash P \triangleright \Delta, c : T_j \quad \Gamma \vdash u : \boldsymbol{G}_j}{\Gamma \vdash c! \, \mathtt{r}_j : l_j \langle u \rangle; P \triangleright \Delta, c : !\{\mathtt{r}_i \langle l_i \langle \boldsymbol{G}_i \rangle \rangle . T_i\}_{i \in I}}$$

$$[\textsc{Bra}] \frac{\forall i \in I \quad \Gamma, z_i : \boldsymbol{G}_i \vdash P_i \triangleright \Delta, c : T_i}{\Gamma \vdash c?\{\mathtt{r}_i : l_i(z_i).P_i\}_{i \in I} \triangleright \Delta, c : ?\{\mathtt{r}_i \langle l_i \langle \boldsymbol{G}_i \rangle \rangle . T_i\}_{i \in I}}$$

$$[\textsc{SSel}] \frac{j \in I \quad \Gamma \vdash P \triangleright \Delta, c : T_j}{\Gamma \vdash c! \, \mathtt{r}_j : l_j \langle c' \rangle; P \triangleright \Delta, c : !\{\mathtt{r}_i \langle l_i \langle T'_i \rangle \rangle . T_i\}_{i \in I}, c' : T'_j}$$

$$[\textsc{SBra}] \frac{\forall i \in I \quad \Gamma \vdash P_i \triangleright \Delta, c : T_i, z_i : T'_i}{\Gamma \vdash c?\{\mathtt{r}_i : l_i(z_i).P_i\}_{i \in I} \triangleright \Delta, c : ?\{\mathtt{r}_i \langle l_i \langle T'_i \rangle \rangle . T_i\}_{i \in I}}$$

$$[\textsc{Par}] \frac{\Gamma \vdash P \triangleright \Delta \quad \Gamma \vdash Q \triangleright \Delta'}{\Gamma \vdash P \mid Q \triangleright \Delta, \Delta'} \quad [\textsc{Inact}] \frac{\forall c \in \mathsf{dom}(\Delta).\ \Delta(c) = \mathsf{end}}{\Gamma \vdash 0 \triangleright \Delta} \quad [\textsc{NRes}] \frac{\Gamma, a : \boldsymbol{G} \vdash P \triangleright \Delta}{\Gamma \vdash (\boldsymbol{\nu} a)P \triangleright \Delta}$$

$$[\textsc{Var}] \quad \Gamma, X : \Delta \vdash X \triangleright \Delta \quad [\textsc{Rec}] \frac{\Gamma, X : \Delta \vdash P \triangleright \Delta}{\Gamma \vdash \mu X.P \triangleright \Delta}$$

[Req] types an initiation request conforming to $\boldsymbol{G}$ for role $\mathtt{r}_1$, assuming the set of index-ordered roles. [Acc] types an initiation accept dually. [Sel] and [Bra] type selection and branching with shared channel passing. Similarly, [SSel] and [SBra] type selection and branching with session delegation. Other rules are standard and identical with the long version of [2]:

> M. Coppo, M. Dezani-Ciancaglini, N. Yoshida and L. Padvani, Global Progress in Dynamically Interleaved Multiparty Sessions, *Mathematical Structures in Computer Science*, To appear. Available from `http://mrg.doc.ic.ac.uk/publications.html`

Typing systems for runtime processes The typing systems for runtime processes ($\Gamma \vdash_\Sigma P \triangleright \Delta$) are identical with Appendix A in the above long version where Σ denotes a set of names of queues in P, except that we do not require

the equivalent relation of message types and subtyping relations to type runime processes.

For typing runtime processes, we use the notations from Appendix A in the above long version, extending to the syntax of selection and branching types. The duality is defined to the extended types following the standard manner. We also use *consistency of session environments*, which assures that each pair of participants in a multiparty conversation performs their mutual communications in a consistent way. Consistency is defined using the notions of projection of endpoint types (see Definition A.1 in the above long version). We call a session environment Δ is *consistent for the session* s (notation $\mathsf{co}(\Delta, s)$) if $s[\mathtt{r}] : T \in \Delta$ and $T \upharpoonright \mathtt{r} \neq \mathsf{end}$ imply $s[\mathtt{r}'] : T' \in \Delta$ and $T \upharpoonright \mathtt{r}' \bowtie T' \upharpoonright \mathtt{r}$ for all $\mathtt{r}$ and $\mathtt{r}'$. A session environment Δ is *consistent* if it is consistent for all sessions which occur in it.

Lemma J.1 (Isomorphism and subtyping in conformance).

1. *Suppose* $T_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M \rangle$ *and* $T'_{\mathtt{r}} \leqslant T_{\mathtt{r}}$ *where* $\leqslant$ *is defined as in [8, Definition 8]. Then* $T'_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M \rangle$.
2. *Suppose* $T_{\mathtt{r}} \asymp \langle \boldsymbol{G}, M \rangle$ *for all* $\mathtt{r} \in R$. *Then there exits* $T'_{\mathtt{r}}$ *such that* $T_{\mathtt{r}} \leqslant T'_{\mathtt{r}}$ *which includes the permutation in Table 10 in the above long version and* $\{T'_{\mathtt{r}}\}_{\mathtt{r} \in R}$ *is consistent.*

Proof. **(1)** Suppose $T'_{\mathtt{r}} \leqslant T_{\mathtt{r}}$ by the output selection subtyping rule in [8, (3) in Definition 4]. Then $T'_{\mathtt{r}}$ satisfies 1(a) and 2(a) in Definition 3.1. Next suppose $T'_{\mathtt{r}} \leqslant T_{\mathtt{r}}$ by the input branching subtyping rule in [8, (2) in Definition 4]. Then $T'_{\mathtt{r}}$ satisfies 1(b) and 2(b) in Definition 3.1. The case of the end type is obvious.

(2) By (1) and Theorems 3.1 and 3.2. □

With this lemma, the typing system with the conformance is equivalent to the typing system with the projection (with merging) up to the equi-recursion.

Proofs of the subject reduction theorem (Theorem 4.1) First the subject congruence theorem is straightforward noting there is no change of $\equiv$ from (see Theorem A.9 in the above long version).

For the reduction relation, we define the reduction between session environments $\Delta \Rightarrow \Delta'$ as in § A.3 in in the above long version changing Π to a singleton. Note that $\Rightarrow$ simulates 1-bound execution of $(\vec{T}, \vec{w})$.

Then we prove the following general subject reduction theorem:

Theorem J.1 (Generalised Subject Reduction). *If* $\Gamma \vdash_{\Sigma} P \triangleright \Delta$ *with* Δ *consistent and* $P \longrightarrow^* P'$, *then* $\Gamma \vdash_{\Sigma} P' \triangleright \Delta'$ *for some* Δ' *such that* $\Delta \Rightarrow^* \Delta'$ *and* Δ' *is consistent.*

Proof. We first note that if Δ is consistent and $\Delta \Rightarrow \Delta'$, then Δ' is consistent. We also note that, by Subject Congruence Theorem, we only need to prove one step. We prove by induction of a derivation of $P \longrightarrow P'$. The only interesting case is [Init], which we prove below.

Case [Init]

$$\overline{a}[\mathtt{r}_1,..,\mathtt{r}_n](x).P_1 \mid a[\mathtt{r}_2](x).P_2 \mid \cdots \mid a[\mathtt{r}_n](x).P_n \longrightarrow (\boldsymbol{\nu} s)(\Pi_{i\in\{1,..,n\}}(P_i[s[\mathtt{r}_i]/x] \mid \Pi_{j\in\{1,..,n\}\setminus i} s[\mathtt{r}_i,\mathtt{r}_j] : \epsilon)).$$

By hypothesis

$$\Gamma \vdash_{\Sigma} \overline{a}[\mathtt{r}_1,..,\mathtt{r}_n](x).P_1 \mid a[\mathtt{r}_2](x).P_2 \mid \cdots \mid a[\mathtt{r}_n](x).P_n \rhd \Delta$$

Then the redex is an initial process, $\Sigma = \emptyset$ and on all the processes in parallel we have

$$\Gamma \vdash a[\mathtt{r}_i](x).P_i \rhd \Delta_i \quad (2 \leq i \leq n) \tag{3}$$

$$\Gamma \vdash \overline{a}[\mathtt{r}_1,..,\mathtt{r}_n](x).P_1 \rhd \Delta_1 \tag{4}$$

where $\Delta = \bigcup_{i=1}^{n} \Delta_i$. Then by [Acc], we have:

$$\frac{T_{\mathtt{r}_i} \asymp \boldsymbol{G} \quad \Gamma \vdash P_i \rhd \Delta_i, x : T_{\mathtt{r}_i} \ i \neq 1 \quad \Gamma \vdash a : \boldsymbol{G}}{} \tag{5}$$

Similarly by [Req], assuming $\mathtt{range}(f) = \{\mathtt{r}_1,..,\mathtt{r}_n\}$, we have:

$$\frac{T_{\mathtt{r}_1} \asymp \boldsymbol{G} \quad \Gamma \vdash a : \boldsymbol{G} \quad \Gamma \vdash P_1 \rhd \Delta_1, x : T_{\mathtt{r}_1}}{} \tag{6}$$

Applying the standard substitution lemma to (5) and (6), we have:

$$\Gamma \vdash P_i[s[\mathtt{r}_i]/x] \rhd \Gamma_i, s[\mathtt{r}_i] : T_{\mathtt{r}_i} \tag{7}$$

Then using (Qinit, GPar) in Table 11 and Table 12 in in the above long version on (7), we have:

$$\Gamma \vdash_{\{s\}} \Pi_{i\in\{1,..,n\}}(P_i[s[\mathtt{r}_i]/x] \mid \Pi_{j\in\{1,..,n\}\setminus i} s[\mathtt{r}_i,\mathtt{r}_j] : \epsilon) \rhd \Delta, s[\mathtt{r}_1] : T_{\mathtt{r}_1}, ..., s[\mathtt{r}_n] : T_{\mathtt{r}_n}$$

Then by Lemma J.1(2), we have consistent $\{T'_{\mathtt{r}_i}\}_{1\leq i\leq n}$ which conforms $\boldsymbol{G}$. Hence

$$\Gamma \vdash_{\{s\}} \Pi_{i\in\{1,..,n\}}(P_i[s[\mathtt{r}_i]/x] \mid \Pi_{j\in\{1,..,n\}\setminus i} s[\mathtt{r}_i,\mathtt{r}_j] : \epsilon) \rhd \Delta, s[\mathtt{r}_1] : T'_{\mathtt{r}_1}, ..., s[\mathtt{r}_n] : T'_{\mathtt{r}_n} \tag{8}$$

Thus by applying (GSRes) Table 12 in in the above long version on (8), we conclude:

$$\Gamma \vdash (\boldsymbol{\nu} s)(\Pi_{i\in\{1,..,n\}}(P_i[s[\mathtt{r}_i]/x] \mid \Pi_{j\in\{1,..,n\}\setminus i} s[\mathtt{r}_i,\mathtt{r}_j] : \epsilon)) \rhd \Delta \tag{9}$$

as required. Other cases are essentially similar with those in the proof of Theorem A.10 in in the above long version. □

Proof of Theorem 4.2 We now prove the progress property. By the same routine as [12, § 5] together with Theorem 4.1, Theorems 3.1 and 3.2. □

Remark on data types We note that since SG and endpoint type messages are injectively mapped to pairs of process labels $l_1, l_2, \ldots$ and $\boldsymbol{G}$ or T (see § 3), if we extend T to include constant types such as `bool` and `nat` as in the formalisms in [2, 12], the following types becomes not coherent in our definition:

$$s[1] : !2\langle a, \texttt{bool}\rangle.?2\langle b, \texttt{nat}\rangle.\texttt{end}, s[2] : ?1\langle a, \texttt{nat}\rangle.!1\langle b, \texttt{nat}\rangle.\texttt{end}$$

since the above does not map the labels to types injectively, i.e. a and b are mapped to the different types in role 1 and role 2. Hence by the conformance checking in § 3, and by the typing system in § 3, we can rule out processes which have the above non-coherent types.