WEBIOT scheduler semantics and proof

April 13, 2023

1 Scheduled WEBI + Theorem

In order to be able to analyse such a complex model, we should be able to constrain non-determinism in webi and being able, if they exist, to collect "all" the possible observations from an initial configuration conf, a provider of events wo(World Oracle), and a set of events $\mathcal{E}$ that may happen in the world.

A step in this direction is to define a scheduled version of WEBIOT. What do we want is that the scheduled version of WEBIOTcan emulate an execution of the classical WEBIOT. We can do this by providing to the scheduler an $\mathcal{L}$ generated by an execution of WEBIOTon a certain initial conf. The scheduled execution of should make us generating an $\mathcal{L}'$ that has the same observations of the ones in $\mathcal{L}$.

1.1 Assumptions for the Scheduler

To tame the non-determinism, we must put constraints on the and tiers' semantics. The following hypotheses allow us to prove the equivalence between the scheduler and the non-deterministic semantics.

Concerning the server and client language, we assume that they rely on a deterministic semantics. Moreover, the services are concurrent, modifying the host memory of the server they are executing on. We assume that a running service never touches the host memory. We present these constraints in Hypotheses 1, 2, and 3.

Hypothesis 1 *Given a web service configurations sc and a host memory μ_h, two different executions result in the same final configuration.*

$$\begin{aligned}\forall \langle sc, \mu_h\rangle : \\ (\langle sc, \mu_h\rangle \rightsquigarrow_S \langle sc', \mu'_h\rangle) \wedge \\ (\langle sc, \mu_h\rangle \rightsquigarrow_S \langle sc'', \mu''_h\rangle) \Rightarrow (sc' = sc'') \wedge (\mu_h = \mu'_h)\end{aligned}$$

Hypothesis 2 *The client semantics is deterministic.*

$$\forall cc, cc', cc'' : \ (cc \rightsquigarrow_C cc') \wedge (cc \rightsquigarrow_C cc'') \Rightarrow (cc' = cc'')$$

Hypothesis 3 *Given a web service configurations sc and a host memory μ_h, executing one step of the semantics does not change the host memory.*

$$\begin{aligned}\forall \langle sc, \mu_h\rangle : \\ (\langle sc, \mu_h\rangle \rightsquigarrow_S \langle sc', \mu'_h\rangle) \Rightarrow \\ (\mu_h = \mu'_h)\end{aligned}$$

To avoid ambiguity in choosing clients or services, we define two hypotheses concerning the uniqueness of the web services and web clients in a configuration. Every client and service running in a configuration is unique. We present these properties of the running model in Hypotheses 4 and 5.

Hypothesis 4 *Every running web service in a configuration is uniquely identified.*

$$\begin{aligned}\forall \langle sc\rangle^{((h,j),i)}, \langle sc'\rangle^{((h',k),l)} \in SS : \\ h = h' \wedge j = k \wedge i = l \Rightarrow \\ sc = sc'\end{aligned}$$

Hypothesis 5 *Every web client in a configuration is uniquely identified.*

$$\forall cc^{(j,u)}, cc'^{(j',u')} \in CC : \\ j = j' \wedge u = u' \Rightarrow \\ cc = cc'$$

1.1.1 Scheduler Configuration

We first introduce a set of labels to represent rules:

- $\mathbb{SS}$ is the label representing the `ServiceStep`. The label can hold indices (j, i), written as $\mathbb{SS}_{(j,i)}$, for making step the service identified by (j, i).
- $\mathbb{RS}$ is the label representing either the `RetService` or `RetServiceBoot`. The label can hold indices (j, i), written as $\mathbb{RS}_{(j,i)}$, for making return the service identified by (j, i).
- $\mathbb{CS}$ is the label representing the `ClientStep`. The label can hold an index j, written as $\mathbb{CS}_j$, for making step the client identified by j.
- $\mathbb{IN}$ is the label representing the `ServiceInit`. The label does not hold indices.
- $\mathbb{CC}_j$ is the label representing the `ClientCall`. The label can hold an index j, written as $\mathbb{CC}_j$, for making call the client identified by j.
- $\mathbb{RN}$ is the label representing the `Run`. The label can hold indices (j, i), written as $\mathbb{RN}_{(j,i)}$, to run the thunk i of the client identified by j.
- $\mathbb{DS}$ is the label representing the `DeviceSensor`. The label does not hold indices.
- $\mathbb{DA}$ is the label representing the `DeviceActuator`. The label can hold indices (j, i), written as $\mathbb{DA}_{(j,i)}$, for making a service (j, i) issuing an actuation.
- $\mathbb{DR}$ is the label representing the `DeviceReading`. The label can hold indices (j, i), written as $\mathbb{DR}_{(j,i)}$, for making a service (j, i) issuing a reading.
- $\mathbb{DV}$ is a generic device label. $\mathbb{DV} \in \{\mathbb{DS}, \mathbb{DA}, \mathbb{DR}\}$.

We add a special label `end` to signal that the 's scheduler has finished computing. We let $\mathbb{X}$ range over labels, *i.e.* $\mathbb{X} \in \{\mathbb{SS}, \mathbb{RS}, \mathbb{CS}, \mathbb{IN}, \mathbb{CC}, \mathbb{RN}, \mathbb{DS}, \mathbb{DA}, \mathbb{DR}, \mathbb{DV}, \texttt{end}\}$.

We define a scheduler transition as a relation between the scheduler's configurations. A scheduler configuration is an extended configuration enclosed in a scheduler's context $[\![]\!]$.

The configuration defined in Section **??** is extended with Π, written $\langle \mathcal{L}, \mathcal{I}, HM, SS, CC, IC, \Pi \rangle$, where Π is a finite list of labels. Intuitively, the sequence Π represents a execution trace and it will be useful for proving equivalence. We define in Definition 1 how an initial configuration is formed.

Definition 1 *A well-formed initial configuration, is a configuration $\langle \mathcal{L}, \mathcal{I}, HM, SS, CC, IC, \Pi \rangle$, where:*

- *Lists $\mathcal{L}$ and Π are empty lists.*
- *Sets SS and CC are empty sets.*
- *Set $\mathcal{I}$ is a non-empty set of clients that are booting.*
- *Mapping HM is a pre-instantiated map of host-memories `hostname` $\rightarrow$ `hostmemory`.*
- *Set IC is a set of intial device configurations.*

Filtering Functions for Generating $\mathfrak{L}$

$$\pi_{\mathfrak{L}} = \lambda\Pi.\ \texttt{lfilter}\ \Pi\ (\lambda x.x\ \in\ \{\mathbb{DS}, \mathbb{DA}, \mathbb{DR}\})$$

Filtering Functions for Generating $\mathfrak{T}$

$$\pi_{\mathfrak{T}} = \lambda\Pi.\ \texttt{lfilter}\ \Pi\ (\lambda x.x\ \in\ \{\mathbb{RN}\})$$

Figure 1: Filtering functions on Π, where `lfilter` is the classic `filter` function on lists in functional programming.

The scheduler context holds three parameters, a label representing a rule name—meaning the current scheduling step executing-and two filtering of the trace Π—one containing device rule applications and the other having the order of the `Run` rule applications. The two filterings ensure the execution of the scheduler semantics to reproduce a final configuration equivalent to the one produced by .

We can write a scheduling context configuration as $[\![\mathsf{conf}]\!]_{\mathbb{X}}$, with the $\mathbb{X}$ denoting the generic scheduling step. For example, to write the scheduling step of the `ServiceStep` we write $[\![\mathsf{conf}]\!]_{\mathbb{SS}}$. Moreover, we use these labels to annotate the non-deterministic transition relation. Intuitively, $\rightsquigarrow_{\mathbb{X}}$ keeps the abstraction of the model, and $\rightsquigarrow_{\mathbb{SS}_{(j,i)}}$ states the application of a service step on a web client signed by (j, i).

Device and Thunk Filterings. We introduce two parameters which are lists of labels: $\mathfrak{L}$ for the device rule applications and $\mathfrak{T}$ for the `Run` rule applications. The purpose is to drive the scheduler's execution and obtain a trace equivalent to the one produced by the non-deterministic execution of the model. Figure 1 presents the algorithms for generating the two execution guides, $\mathfrak{L}$ and $\mathfrak{T}$. We call the two filtering functions $\pi_{\mathfrak{L}}$ and $\pi_{\mathfrak{T}}$, which return a sub-trace given a trace Π.

Scheduler Transition Relation. We write the scheduler relation as $\rightsquigarrow_{\mathfrak{S}}$. Thus, the relation between scheduler's configurations has the following form:

$$\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}]\!]_{\mathbb{X}}^{(\mathfrak{L},\mathfrak{T})} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}']\!]_{\mathbb{X}'}^{(\mathfrak{L}',\mathfrak{T}')}$$

1.2 Annotated WEBiotSemantics

We extend a WEBIOTconfiguration conf with Π. The latter is the trace of a WEBIOTexecution. Now $conf = \langle \mathcal{L}, \mathcal{I}, HM, SS, CC, IC, \Pi \rangle$. Every time a rule is applied successfully, the rule logs itself in Π.

$\rightsquigarrow_{\mathbb{SS}}^{j,i}$

SERVERSTEP

$$\frac{SS = \{\langle sc \rangle^{((h,j),i)}\} \uplus SS' \qquad \langle sc, HM(h) \rangle \rightsquigarrow_S \langle sc', \mu'_h \rangle \qquad HM' = HM[h \mapsto \mu'_h] \qquad SS'' = SS' \cup \{\langle sc' \rangle^{((h,j),i)}\} \qquad \Pi' = ((j,i), \mathbb{SS}) :: \Pi}{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{SS, HM, \Pi\} \rightsquigarrow_{\mathbb{SS}}^{j,i} \mathsf{conf}\{SS'', HM', \Pi'\}}$$

$\rightsquigarrow_{\mathbb{RS}}^{j,i}$

RETSERVICE

$$\frac{SS = SS' \uplus \{\langle v_s \rangle^{((h,j),i)}\} \qquad CC = CC' \uplus \{\langle cc, B \cup \{b^i\}, T \rangle^{(j,u)}\} \qquad \mathsf{gencc}(v_s) = v_c \qquad CC'' = CC' \cup \{\langle cc, B, T \cup \{(b, v_c)^i\} \rangle^{(j,u)}\} \qquad \Pi' = ((j,i), \mathbb{RS}) :: \Pi}{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{SS, CC, \Pi\} \rightsquigarrow_{\mathbb{RS}}^{j,i} \mathsf{conf}\{SS', CC'', \Pi'\}}$$

RetServiceBoot

$$\frac{SS = SS' \uplus \{\langle v_s \rangle^{((h,j),0)}\} \quad CC = CC' \uplus \{\langle \mathsf{boot} \rangle^{(j,u)}\} \quad \mathsf{gencc}(v_s) = cc \quad CC'' = CC' \cup \{\langle cc, \emptyset, \emptyset \rangle\}^{(j,u)}\} \quad \Pi' = ((j,0), \mathbb{RS}) :: \Pi}{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{SS, CC, \Pi\} \rightsquigarrow^{j,0}_{\mathbb{RS}} \mathsf{conf}\{SS', CC'', \Pi'\}}$$

$\rightsquigarrow^{j}_{\mathbb{CS}}$

ClientStep

$$\frac{CC = \{\langle cc, B, T \rangle^{(j,u)}\} \uplus CC' \quad cc \rightsquigarrow_C cc' \quad CC'' = CC' \cup \{\langle cc', B, T \rangle^{(j,u)}\} \quad \Pi' = (j, \mathbb{CS}) :: \Pi}{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{CC, \Pi\} \rightsquigarrow^{j}_{\mathbb{CS}} \mathsf{conf}\{CC'', \Pi'\}}$$

$\rightsquigarrow_{\mathbb{IN}}$

ServiceInit

$$\frac{\mathcal{I} = ((j,u), v_a) \uplus \mathcal{I}' \quad \mathtt{WebServices}(u) = (\mathsf{serviceinit}, h) \quad sc = \mathsf{serviceinit}(\mathsf{v_a}) \quad CC' = CC \cup \{\langle \mathsf{boot} \rangle^{(j,u)}\} \quad SS' = SS \cup \{sc^{((h,j),0)}\} \quad \Pi' = \mathbb{IN} :: \Pi}{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{\mathcal{I}, CC, SS, \Pi\} \rightsquigarrow_{\mathbb{IN}} \mathsf{conf}\{\mathcal{I}', CC', SS', \Pi'\}}$$

$\rightsquigarrow^{j}_{\mathbb{CC}}$ A subtle thing might be the origin of the i identifier. We should assume that it is provided by the client call transition, and it is supposed to be fresh for the calling client j. This means that i is a client-side token passed as a parameter of each call. The client semantics handles the way the client keeps track of the number of the call. A Naïve implementation might be the one that increases by one the value of i each time a client calls.

ClientCall

$$\frac{CC = \{\langle cc, B, T \rangle^{(j,u)}\} \uplus CC' \quad cc \rightsquigarrow_C^{u'?v_a, b^i} cc' \quad \mathtt{WebServices}(u') = (\mathsf{serviceinit}, h) \quad sc = \mathsf{serviceinit}(\mathsf{v_a}) \quad B' = B \cup \{b^i\} \quad CC'' = CC' \cup \{\langle cc', B', T \rangle^{(j,u)}\} \quad SS'' = SS \cup \{sc^{((h,j),i)}\} \quad \Pi' = (j, \mathbb{CC}) :: \Pi}{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{CC, SS, \Pi\} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}\{CC'', SS'', \Pi'\}}$$

$\rightsquigarrow^{j,i}_{\mathbb{RN}}$

Run

$$\frac{CC = \{\langle \langle v_c', \mu \rangle, B, T \uplus \{(\lambda x.P, v_c)^i\} \rangle^{(j,u)}\} \cup CC' \quad CC'' = \{\langle \langle (P\{x \mapsto v_c\}, \mu \rangle, B, T \rangle^{(j,u)}\} \cup CC' \quad \Pi' = ((j,i), \mathbb{RN}) :: \Pi}{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{CC, \Pi\} \rightsquigarrow^{j,i}_{\mathbb{RN}} \mathsf{conf}\{CC'', \Pi'\}}$$

$\rightsquigarrow_{\mathbb{DS}}$

DeviceSensor

$$\frac{\mathtt{last\text{-}time}(\mathcal{L}) \leq t' \quad \mathsf{wo}(\mathcal{L}, \mathcal{E}, t') = pe \quad IC \rightsquigarrow_D^{\mathsf{sens}(pe)} IC' \quad \mathcal{L}' = (pe, t') :: \mathcal{L} \quad \Pi' = \mathbb{DS} :: \Pi}{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{\mathcal{L}, IC, \Pi\} \rightsquigarrow_{\mathbb{DS}} \mathsf{conf}\{\mathcal{L}', IC', \Pi'\}}$$

$\rightsquigarrow^{j,i}_{\mathbb{DA}}$

DeviceActuator

$$\frac{SS = \{\langle sc \rangle^{((h,j),i)}\} \uplus SS' \quad \langle sc, HM(h) \rangle \rightsquigarrow_S^{\mathsf{act}(a,d,p)} \langle sc', \mu_h' \rangle \quad HM' = HM[h \mapsto \mu_h'] \quad IC \rightsquigarrow_D^{\mathsf{act}(a,d,p)} IC' \quad SS'' = \{\langle sc' \rangle^{((h,j),i)}\} \cup SS' \quad \mathtt{last\text{-}time}(\mathcal{L}) \leq t' \quad \mathcal{L}' = (a,d,t') :: \mathcal{L} \quad \Pi' = ((j,i), \mathbb{DA}) :: \Pi}{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{\mathcal{L}, SS, HM, IC, \Pi\} \rightsquigarrow_{\mathbb{DA}_{j,i}}^{\mathsf{conf}\{\mathcal{L}', SS'', HM', IC', \Pi'\}}}$$

Step 1

$$\textsc{Init}\quad \frac{\mathcal{I} \neq \emptyset \qquad \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{\mathcal{I}\} \rightsquigarrow_{\mathbb{IN}} \mathsf{conf}'}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{\mathcal{I}\}]\!]^{(\mathfrak{L},\mathfrak{T})}_{\mathbb{IN}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}']\!]^{(\mathfrak{L},\mathfrak{T})}_{\mathbb{IN}}}$$

$$\textsc{InitDone}\quad \frac{\mathcal{I} = \emptyset}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{\mathcal{I}\}]\!]^{(\mathfrak{L},\mathfrak{T})}_{\mathbb{IN}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{\mathcal{I}\}]\!]^{(\mathfrak{L},\mathfrak{T})}_{\mathbb{SS}}}$$

Figure 2: Semantic rules for Step 1

Step 2

$$\textsc{ServiceChosen}\quad \frac{SS = \{SS_r, SS_a, SS_g, SS_e\} \quad SS_r \neq \emptyset \qquad \langle sc\rangle^{((h,j),i)} \in SS_r}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{SS}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{SS}_{(j,i)}}}$$

$$\textsc{MultiServiceSteps}\quad \frac{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{SS\} \rightsquigarrow_{\mathbb{SS}_{(j,i)}} \mathsf{conf}'\{SS'\} \quad SS' = \{SS_r, SS_a, SS_g, SS_e\} \qquad \langle sc\rangle^{(((h,j),i)} \in SS_r}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{SS}_{(j,i)}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}'\{SS'\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{SS}_{(j,i)}}}$$

$$\textsc{OneServiceStep}\quad \frac{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{SS\} \rightsquigarrow_{\mathbb{SS}_{(j,i)}} \mathsf{conf}'\{SS'\} \quad SS' = \{SS_r, SS_a, SS_g, SS_e\} \qquad \langle sc\rangle^{((h,j),i)} \notin SS_r}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{SS}_{(j,i)}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}'\{SS'\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{SS}}}$$

$$\textsc{ServicesDone}\quad \frac{SS = \{SS_r, SS_a, SS_g, SS_e\} \qquad SS_r = \emptyset}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{SS}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{RS}}}$$

Figure 3: Semantic rules for Step 2

$\rightsquigarrow^{j,i}_{\mathbb{DR}}$

$$\textsc{DeviceReading}\quad \frac{SS = \{\langle sc\rangle^{((h,j),i)}\} \uplus SS' \qquad \langle sc, HM(h)\rangle \rightsquigarrow^{\mathsf{get}(d,p,v_a)}_{S} \langle sc', \mu'_h\rangle \quad IC \rightsquigarrow^{\mathsf{get}(d,p,v_a)}_{D} IC' \qquad HM' = HM[h \mapsto \mu'_h] \qquad SS'' = \{\langle sc'\rangle^{((h,j),i)}\} \cup SS' \qquad \Pi' = ((j,i), \mathbb{DR}) :: \Pi}{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{SS, HM, IC, \Pi\} \rightsquigarrow^{\mathsf{conf}\{SS'', HM', IC', \Pi'\}}_{\mathbb{DR}_{j,i}}}$$

1.2.1 Small-Step Semantics

We present the scheduler's semantics by presenting each step in a paragraph. Given an initial configuration conf, an initial configuration for the scheduling is $[\![\mathsf{conf}]\!]^{(\mathfrak{L},\mathfrak{T})}_{\mathbb{IN}}$, for some $\mathfrak{L}$ and $\mathfrak{T}$.

Step 1, $\mathbb{IN}$. The first iteration of the scheduler starts evaluating all the initializers. This approach introduces all the booting clients and the related booting services. We show the semantic rules in Figure 2.

The rule `Init` performs a `ServiceInit` transition—Figure **??**-in case the set $\mathcal{I}$ is not empty. Note that the left-hand side of the conclusions of the inference rule keeps the label $\mathbb{IN}$, stating that the scheduler step does not change.

The rule `InitDone` makes the scheduler transit to the next step in case the set of initializers is empty. As we deal with a finite number initializing of clients, once the $\mathcal{I}$ set is emptied, it will not have new elements to initialize the next iteration. Step 1 of the scheduler is performed only at the first iteration. Note that this rule makes the scheduler transit to the second scheduler step by changing the label $\mathbb{IN}$ to $\mathbb{SS}$ in the context.

Step 2, $\mathbb{SS}$. The second step of the scheduling policy consumes all the possible `ServiceSteps`. This means that each time the rule is applied, an element of SS_r performs an execution step. The step finishes as soon as no more services can step. We present the semantics in Figure 3.

The `ServiceChosen` rule picks a service that can step and updates the scheduler. The rule label is set to context with $\mathbb{SS}_{(j,i)}$, meaning that the service (j,i) is the one chosen to step.

Step 3

$$\textsc{RetServiceChosen}\ \frac{SS = \{SS_r, SS_a, SS_g, SS_e\} \quad SS_e \neq \emptyset \quad \langle sc \rangle^{((h,j),i)} \in SS_e \quad \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{SS\} \rightsquigarrow_{\mathbb{RS}_{(j,i)}} \mathsf{conf}'\{SS'\}}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{RS}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}'\{SS'\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{RS}}}$$

$$\textsc{RetServicesDone}\ \frac{SS = \{SS_r, SS_a, SS_g, SS_e\} \quad SS_e = \emptyset}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{RS}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{DV}}}$$

Figure 4: Semantic rules for Step 3

The `MultiServiceSteps` rule makes step the web service (j, i). If the service can do one more step, the label is left unchanged, updating only the configuration.

The `OneServiceStep` acts closely to the `MultiServiceStep` rule. The service performs one small evaluation step, and the rule label is set to $\mathbb{SS}$. This means that the service (j, i) cannot step again. Another service has to be chosen or, in case, transit to the next scheduling step.

The `ServicesDone` rule makes the scheduler transit to the next scheduling step. Hence, the SS_r partition set is empty, and the rule label changes to $\mathbb{RS}$.

Generally, the running service executing on a host are concurrent, as they share the host memory. In this preliminary study of , we decided to put constraints, saying that the running services never touch the host memory. This permits the `MultiServiceStep` to be correct. Moreover, we constrain the server language and consider only deterministic languages. These constraints are formally presented in Hypotheses 3 and 1.

Step 3, $\mathbb{RS}$. The third step consumes all the elements of SS_e, and only `RetService` and `RetServiceBoot` apply.
We present the semantics in Figure 4.

The `RetServiceChosen` rule picks a service in SS_e. This service is related to a booting or a running client. Hence, one of the two applies. The configuration is updated accordingly.

The `RetServiceDone` rule makes the scheduler transit to the next step because of $SS_g = \emptyset$. The rule label changes to $\mathbb{DV}$. The configuration conf, $\mathfrak{L}$, and $\mathfrak{T}$ are immuted.

Step 4, $\mathbb{DV}$. The fourth step of the scheduler consumes all the services interacting with devices. We are considering the $\mathbb{DS}$, $\mathbb{DA}$, and $\mathbb{DR}$ rules, and trace $\mathfrak{L}$ guides the application of these rules. We present the semantics in Figure 5.

The `DevSens` rule applies a `DeviceSensor` in case the head of the device-trace $\mathfrak{L}$ is $\mathbb{DS}$. The configuration conf is updated accordingly. Regarding the scheduler context, the new context contains the list $\mathfrak{L}$ without its head, and the rule label stays unchanged.

The `DevAct` rule, if the head of the $\mathfrak{L}$ is an actuator, applies the `DeviceActuation` rule to the configuration. The service requesting the act is (j, i), and must be an element of SS_a. The configuration and the $\mathfrak{L}$ are updated, and the rule label stays unchanged.

Similarly, the `DevRead` performs a `DeviceReading` requested by the service (j, i) in case the service belongs to SS_g. The head of the $\mathfrak{L}$ is removed, the configuration is updated, and the rule label does not change.

The `DevNoAct` and `DevNoRead` apply when respectively:

- The head of $\mathfrak{L}$ is an actuator request done by a service (j, i), but the service is not an element of SS_a.
- The head of $\mathfrak{L}$ is a read request done by a service (j, i), but the service is not an element of SS_g.

In this case, for keeping the device-rules application order, the rule makes the scheduler context transit to the next step to the scheduler. The rule label is set to $\mathbb{CS}$.

Finally, The `DevStepDone` applies when the list $\mathfrak{L}$ is empty. We know that if this trace is empty, then both the partition sets SS_a and SS_g are empty.
This rule application only changes the rule label to $\mathbb{CS}$.

Step 4

$$\textsc{DevSens}\ \frac{\mathfrak{L} = \mathbb{DS} :: \mathfrak{L}' \qquad \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{DS}} \mathsf{conf}'}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{DV}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}']\!]^{\mathfrak{L}',\mathfrak{T}}_{\mathbb{DV}}}$$

$$\textsc{DevAct}\ \frac{\mathfrak{L} = \mathbb{DA}_{(j,i)} :: \mathfrak{L}' \qquad SS = \{SS_r, SS_a, SS_g, SS_e\} \qquad \langle sc \rangle^{((h,j),i)} \in SS_a \qquad \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{SS\} \rightsquigarrow^{\mathsf{conf}'\{SS'\}}_{\mathbb{DA}} (j,i)}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{DV}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}'\{SS'\}]\!]^{\mathfrak{L}',\mathfrak{T}}_{\mathbb{DV}}}$$

$$\textsc{DevRead}\ \frac{\mathfrak{L} = \mathbb{DR}_{(j,i)} :: \mathfrak{L}' \qquad SS = \{SS_r, SS_a, SS_g, SS_e\} \qquad \langle sc \rangle^{((h,j),i)} \in SS_g \qquad \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{SS\} \rightsquigarrow^{\mathsf{conf}'\{SS'\}}_{\mathbb{DR}} (j,i)}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{\mathrm{SS}\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{DV}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}'\{SS'\}]\!]^{\mathfrak{L}',\mathfrak{T}}_{\mathbb{DV}}}$$

$$\textsc{DevNoAct}\ \frac{\mathfrak{L} = \mathbb{DA}_{(j,i)} :: \mathfrak{L}' \qquad SS = \{SS_r, SS_a, SS_g, SS_e\} \qquad \langle sc \rangle^{((h,j),i)} \notin SS_a}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{DV}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CS}}}$$

$$\textsc{DevNoRead}\ \frac{\mathfrak{L} = \mathbb{DR}_{(j,i)} :: \mathfrak{L}' \qquad SS = \{SS_r, SS_a, SS_g, SS_e\} \qquad \langle sc \rangle^{((h,j),i)} \notin SS_g}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{DV}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CS}}}$$

$$\textsc{DevStepDone}\ \frac{SS = \{SS_r, SS_a, SS_g, SS_e\} \qquad \mathfrak{L} = [] \Rightarrow (SS_a = \emptyset \wedge SS_g = \emptyset)}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{DV}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{SS\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CS}}}$$

Figure 5: Semantic rules for Step 4

Step 5, $\mathbb{CS}$ The fifth step handles the `ClientStep` application. We present the small-step semantics in Figure 6.

The `ClientChosen` rule picks a client to step from the partition set CC_r. If there is a client (j, u), the rule label is set to $\mathbb{CS}_j$.

The `MultiClientSteps` rule performs an execution step on the web client signed by (j, u). The client, still a member of CC_r in conf', is set again for evaluation. Hence, the rule label does not change, and the configuration is updated.

The `OneClientStep` rule performs an execution step on the web client (j, u). The client is no more in CC_r of conf'. Hence, the rule label is set back to $\mathbb{CC}$. The label change permits choosing another client to make an execution step, to go to the next scheduling step, or to terminate , according to Definition **??**, page 128. The configuration is updated.

The rule `ClientsDone` applies if there are no more clients in the partition set CC_r. In this case, the rule label of the scheduler execution context is set to $\mathbb{CC}$. Note that the last premise is $|CC_e| \neq |CC|$, meaning the cardinality of the partition and the whole set CC differ. There are still clients that might call or have a thunk to run.

If the last premise is $|CC_e| = |CC|$, then the last rule applies, namely `WebiDone`. The rule label is set to `end`, meaning that the scheduler must stop executing, and conf becomes the final configuration.

Step 6, $\mathbb{CC}$ This scheduling step is concerned with applying, wherever possible, the ClientCall rule. We provide the small-step rules in Figure 7. The `CallChoosen` rule picks a client j from the partition set CC_c. The client performs a `ClientCall` step, returning an updated configuration. The scheduler context is updated with the new configuration conf'.

Otherwise, the `CallsDone` applies when the partition set CC_c is empty, meaning there are no more calls to evaluate this iteration. Hence, this rule updates the rule label to $\mathbb{RN}$ for going to the next step.

Step 5

$$\textsc{ClientChosen}\ \frac{CC = \{CC_r, CC_c, CC_t, CC_e\} \qquad CC_r \neq \emptyset \qquad cc^{(j,u)} \in CC_r}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CS}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CS}_j}}$$

$$\textsc{MultiClientSteps}\ \frac{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{CC\} \rightsquigarrow_{\mathbb{CS}_j} \mathsf{conf}'\{CC'\} \qquad CC' = \{CC_r, CC_c, CC_t, CC_e\} \qquad cc^{(j,u)} \in CC'}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CS}_j} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}'\{CC'\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CS}_j}}$$

$$\textsc{OneClientStep}\ \frac{\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{CC\} \rightsquigarrow_{\mathbb{CS}_j} \mathsf{conf}'\{CC'\} \qquad CC' = \{CC_r, CC_c, CC_t, CC_e\} \qquad cc^{(j,u)} \notin CC_r}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CS}_j} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}']\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CC}}}$$

$$\textsc{ClientsDone}\ \frac{CC = \{CC_r, CC_c, CC_t, CC_e\} \qquad CC_r = \emptyset \qquad |CC_e| \neq |CC|}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CS}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CC}}}$$

$$\textsc{WebiDone}\ \frac{CC = \{CC_r, CC_c, CC_t, CC_e\} \qquad CC_r = \emptyset \qquad |CC_e| = |CC|}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{CC\}]\!]^{[],[]}_{\mathbb{CS}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{CC\}]\!]^{[],[]}_{\mathtt{end}}}$$

Figure 6: Semantic rules for Step 5

Step 6

$$\textsc{CallChosen}\ \frac{CC = \{CC_r, CC_c, CC_t, CC_e\} \qquad CC_c \neq \emptyset \qquad cc^{(j,u)} \in CC_c \qquad \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{CC\} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}'\{CC'\}}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CC}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}'\{CC'\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CC}}}$$

$$\textsc{CallsDone}\ \frac{CC = \{CC_r, CC_c, CC_t, CC_e\} \qquad CC_c = \emptyset}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{CC}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{RN}}}$$

Figure 7: Semantic rules for Step 6

Step 7

$$\textsc{RunChosen}\ \frac{\mathfrak{T} = (j,i) :: \mathfrak{T}' \qquad CC = \{CC_r, CC_c, CC_t, CC_e\} \qquad \langle cc, B, T\rangle^{(j,u)} \in CC_t \qquad t^i \in T \qquad \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}\{CC\} \rightsquigarrow_{\mathbb{RN}_{(j,i)}} \mathsf{conf}'\{CC'\}}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{RN}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}'\{CC'\}]\!]^{\mathfrak{L},\mathfrak{T}'}_{\mathbb{RN}}}$$

$$\textsc{RunNotFound}\ \frac{\mathfrak{T} = (j,i) :: \mathfrak{T}' \qquad CC = \{CC_r, CC_c, CC_t, CC_e\} \qquad \langle cc, B, T\rangle^{(j,u)} \in CC_t \qquad t^i \notin T}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{RN}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{SS}}}$$

$$\textsc{RunDone}\ \frac{CC = \{CC_r, CC_c, CC_t, CC_e\} \qquad \mathfrak{T} = [] \wedge CC_t = \emptyset}{\mathsf{wo}, \mathcal{E} \vdash [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{RN}} \rightsquigarrow_{\mathfrak{S}} [\![\mathsf{conf}\{CC\}]\!]^{\mathfrak{L},\mathfrak{T}}_{\mathbb{SS}}}$$

Figure 8: Semantic rules for Step 7

Step 7, $\mathbb{RN}$ In the last step of the scheduler, we present rules for handling the `Run` rule application. Similarly to the device's rules, the order in which thunks execute matters. For example, consider two thunks of a client,

one that wants to call a service that opens the window and one that calls the one that closes it—the observations made on the final $\mathcal{L}$ of the final configuration change depending on the order of the thunk choice. Hence, this step is guided by $\mathfrak{T}$, the filtering of the non-deterministic trace describing the order of the $\mathbb{RN}$ rule application.

The rule `RunChosen` applies the $\mathbb{RN}$ transition, executing the thunk (j, i), which is the head of the list $\mathfrak{T}$. A constraint is that the thunk i of the client j must exist. The rule returns the new configuration and updates the scheduler context with the $\mathfrak{T}$ deprived of its head.

Otherwise, the rule `RunNotFound` applies if the thunk does not belong to the client's set T. It means that the thunk is still a callback, or the client has not yet called the service (j, i). In this case, the rule label of the scheduler context changes to $\mathbb{SS}$. We keep the order of the $\mathbb{RN}$ applications. We remark that step 1 of the scheduler is executed only in the first iteration. Afterward, the scheduler does not consider further client initializations.

Finally, if the list $\mathfrak{T}$ is empty, no other client's thunk will ever be executed. The scheduler directly passes to the second step.

1.3 Theorem

Remark 1 *A well formed initial WEBIOTconfiguration, is a WEBIOTconfiguration* $\langle \mathcal{L}, \mathcal{I}, HM, SS, CC, IC, \Pi \rangle$, *where* $\mathcal{L}, SS, CC$ *are empty sets, and* $\mathcal{I}$ *is a non-empty multiset of clients that are booting.*

Remark 2 *The sets* SS *and* CC *can be functionally partitioned in subsets. For the first, we can partition the set by considering services that either are evaluating their code, asking to a device to act, to read the device state, or that have finished to compute, respectively* SS_r, SS_a, SS_g, *and* SS_e. *Similarly, we can distinguish partition-subsets also for the web-clients. Indeed we call* CC_r, CC_c, *and* CC_e *web-clients that are respectively either run their code, perform a call, or have converged to a final result.*

Notice that every client or service that exist in a WEBIOTconfiguration can be placed in only one of the subsets.

Lemma 1 (Filter $\mathbb{IN}$) *Given a well formed initial WEBIOTconfiguration* conf, *a function* $\mathtt{filterInits} : \Pi \to (\Pi \times \Pi)$, *that given a WEBIOTtrace* Π', *returns two traces* $(\Pi^{\mathbb{IN}}, \Pi^{\neg\mathbb{IN}})$, *where* $|\Pi^{\mathbb{IN}}| + |\Pi^{\neg\mathbb{IN}}| = |\Pi|$ *and* $\Pi^{\mathbb{IN}}$ *is a trace containing all the* $\mathbb{IN}$, *and* $\Pi^{\neg\mathbb{IN}}$ *the trace withouth any of the inits, then*

$$\begin{gathered}
\forall \mathsf{wo}, \mathcal{E}, \mathsf{conf}, \mathcal{I}, HM, IC, n, \mathsf{conf}_n, CC_e\mathcal{L}, IC', \Pi_n. \\
\mathcal{I} \neq \emptyset \quad \wedge \quad \mathsf{conf} = \langle [], \mathcal{I}, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,\emptyset\}, IC, [] \rangle \quad \wedge \quad \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^n \mathsf{conf}_n \quad \wedge \\
\mathsf{conf}_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset, CC_e\}, IC', \Pi_n \rangle \quad \wedge \quad |\mathcal{I}| = |CC_e| \\
\Rightarrow \\
\exists \mathsf{conf}_{k1}, \mathsf{conf}'_n, k1, k2, \Pi^{\mathbb{IN}}_{k1}, \Pi^{\neg\mathbb{IN}}_{k2}. \\
|\Pi^{\mathbb{IN}}_{k1}| + |\Pi^{\neg\mathbb{IN}}_{k2}| = |\Pi_n| \quad \wedge \quad |\Pi^{\neg\mathbb{IN}}_{k2}| < |\Pi_n| \quad \wedge \quad \mathsf{conf}_{k1} = \langle [], \emptyset, HM, \{SS_r,\emptyset,\emptyset,\emptyset\}, \{CC_r,\emptyset,\emptyset,\emptyset\}, IC, \Pi^{\mathbb{IN}}_{k1} \rangle \\
\wedge \quad SS_r \neq \emptyset \quad \wedge \quad CC_r \neq \emptyset \quad \wedge \quad \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k1}_{\Pi^{\mathbb{IN}}_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathbb{IN}}_{k2}} \mathsf{conf}'_n \quad \wedge \\
\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}}_{k1} :: \Pi^{\neg\mathbb{IN}}_{k2} \rangle
\end{gathered}$$

Lemma 2 ($\mathbb{SS}$ interval) *The generic form of the previous lemma*

$$\forall wo, \mathcal{E}, k1, k2, n, \mathsf{conf}, \mathsf{conf}_{k1}, \mathsf{conf}_n, \mathcal{I}, HM, SS_r, CC_r, CC_e, CC'_e, IC, IC', IC'', \mathcal{L}, \mathcal{L}', \Pi^{\mathbb{IN}}, \Pi^{\mathfrak{S}}, \Pi^{\neg\mathfrak{S}}_{k2}.$$
$$|\Pi^{\mathbb{IN}}| + |\Pi^{\mathfrak{S}}| = k1 \quad \wedge \quad k1 + |\Pi^{\neg\mathfrak{S}}_{k2}| = n \quad \wedge \quad \mathsf{conf} = \langle [], \mathcal{I}, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,\emptyset\}, IC, [] \rangle \quad \wedge$$
$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}}_{k2}} \mathsf{conf}_n \quad \wedge$$
$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1} \rangle$$
$$\wedge \quad SS_r \neq \emptyset \quad \wedge \quad \mathsf{conf}_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\neg\mathfrak{S}}_{k2} \rangle$$
$$\wedge \quad |\mathcal{I}| = |CC_e| \quad \wedge \quad |CC'_e| < |\mathcal{I}|$$
$$\Rightarrow$$
$$\exists k1', k2', \mathsf{conf}_{(k1+k1')}, \mathsf{conf}'_n, SS_a, SS_g, SS_e, \Pi^{\mathbb{SS}}_{k1'}, \Pi^{\neg\mathfrak{S}'}_{k2'}.$$
$$|\Pi^{\neg\mathfrak{S}}_{k2}| = |\Pi^{\mathbb{SS}}_{k1'}| + |\Pi^{\neg\mathfrak{S}'}_{k2'}| \quad \wedge \quad |\Pi^{\neg\mathfrak{S}'}_{k2'}| < |\Pi^{\neg\mathfrak{S}}_{k2}| \quad \wedge \quad \mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k1'}_{\Pi^{\mathbb{SS}}_{k1'}} \mathsf{conf}_{(k1+k1')} \rightsquigarrow^{k2'}_{\Pi^{\neg\mathfrak{S}'}_{k2'}} \mathsf{conf}'_n$$
$$\wedge \quad \mathsf{conf}_{(k1+k1')} = \langle \mathcal{L}', \emptyset, HM, \{\emptyset, SS_a, SS_g, SS_e\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC'', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\mathbb{SS}}_{k1'} \rangle \quad \wedge$$
$$(SS_a \neq \emptyset \vee SS_g \neq \emptyset \vee SS_e \neq \emptyset) \quad \wedge \quad \mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\mathbb{SS}}_{k1'} :: \Pi^{\neg\mathfrak{S}'}_{k2'} \rangle$$

Lemma 3 ($\mathbb{RS}$ interval) *The generic form of the previous lemma*

$$\forall wo, \mathcal{E}, k1, k2, n, \mathsf{conf}, \mathsf{conf}_{k1}, \mathsf{conf}_n, \mathcal{I}, HM, SS_a, SS_g, SS_e, CC_r, CC_e, CC'_e, IC, IC', IC'', \mathcal{L}, \mathcal{L}', \Pi^{\mathbb{IN}}, \Pi^{\mathfrak{S}}, \Pi^{\neg\mathfrak{S}}_{k2}.$$
$$|\Pi^{\mathbb{IN}}| + |\Pi^{\mathfrak{S}}| = k1 \quad \wedge \quad k1 + |\Pi^{\neg\mathfrak{S}}_{k2}| = n \quad \wedge \quad \mathsf{conf} = \langle [], \mathcal{I}, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,\emptyset\}, IC, [] \rangle \quad \wedge$$
$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}}_{k2}} \mathsf{conf}_n \quad \wedge$$
$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{\emptyset, SS_a, SS_g, SS_e\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1} \rangle$$
$$\wedge \quad SS_e \neq \emptyset \quad \wedge \quad \mathsf{conf}_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\neg\mathfrak{S}}_{k2} \rangle$$
$$\wedge \quad |\mathcal{I}| = |CC_e| \quad \wedge \quad |CC'_e| < |\mathcal{I}|$$
$$\Rightarrow$$
$$\exists k1', k2', \mathsf{conf}_{(k1+k1')}, \mathsf{conf}'_n, CC'_r, \Pi^{\mathbb{RS}}_{k1'}, \Pi^{\neg\mathfrak{S}'}_{k2'}.$$
$$|\Pi^{\neg\mathfrak{S}}_{k2}| = |\Pi^{\mathbb{RS}}_{k1'}| + |\Pi^{\neg\mathfrak{S}'}_{k2'}| \quad \wedge \quad |\Pi^{\neg\mathfrak{S}'}_{k2'}| < |\Pi^{\neg\mathfrak{S}}_{k2}| \quad \wedge \quad \mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k1'}_{\Pi^{\mathbb{RS}}_{k1'}} \mathsf{conf}_{(k1+k1')} \rightsquigarrow^{k2'}_{\Pi^{\neg\mathfrak{S}'}_{k2'}} \mathsf{conf}'_n$$
$$\wedge \quad \mathsf{conf}_{(k1+k1')} = \langle \mathcal{L}', \emptyset, HM, \{\emptyset, SS_a, SS_g, \emptyset\}, \{CC'_r, \emptyset, \emptyset, CC'_e\}, IC'', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\mathbb{RS}}_{k1'} \rangle \quad \wedge$$
$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\mathbb{RS}}_{k1'} :: \Pi^{\neg\mathfrak{S}'}_{k2'} \rangle$$

Lemma 4 ($\mathbb{DV}$ Interval) *The generic form of the previous lemma*

$$\forall wo, \mathcal{E}, k1, k2, n, \mathsf{conf}, \mathsf{conf}_{k1}, \mathsf{conf}_n, \mathcal{I}, HM, SS_a, SS_g, CC_r, CC_e, CC'_e, IC, IC', IC'', \mathcal{L}, \mathcal{L}', \Pi^{\mathbb{IN}}, \Pi^{\mathfrak{S}}, \Pi^{\neg\mathfrak{S}}_{k2}.$$
$$|\Pi^{\mathbb{IN}}| + |\Pi^{\mathfrak{S}}| = k1 \quad \wedge \quad k1 + |\Pi^{\neg\mathfrak{S}}_{k2}| = n \quad \wedge \quad \mathsf{conf} = \langle [], \mathcal{I}, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,\emptyset\}, IC, [] \rangle \quad \wedge$$
$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}}_{k2}} \mathsf{conf}_n \quad \wedge$$
$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{\emptyset, SS_a, SS_g, \emptyset\}, \{CC_r, \emptyset, \emptyset, CC'_e, \}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1} \rangle$$
$$\wedge \quad \mathsf{conf}_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\neg\mathfrak{S}}_{k2} \rangle$$
$$\wedge \quad |\mathcal{I}| = |CC_e| \quad \wedge \quad |CC'_e| < |\mathcal{I}|$$
$$\Rightarrow$$
$$\exists k1', k2', \mathsf{conf}_{(k1+k1')}, \mathsf{conf}'_n, SS_r, IC''', \mathcal{L}'', \Pi^{(\mathbb{DV}+\mathbb{DS})}_{k1'}, \Pi^{\neg\mathfrak{S}'}_{k2'}.$$
$$|\Pi^{\neg\mathfrak{S}}_{k2}| = |\Pi^{(\mathbb{DV}+\mathbb{DS})}_{k1'}| + |\Pi^{\neg\mathfrak{S}'}_{k2'}| \quad \wedge \quad |\Pi^{\neg\mathfrak{S}'}_{k2'}| < |\Pi^{\neg\mathfrak{S}}_{k2}| \quad \wedge \quad \mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k1'}_{\Pi^{(\mathbb{DV}+\mathbb{DS})}_{k1'}} \mathsf{conf}_{(k1+k1')} \rightsquigarrow^{k2'}_{\Pi^{\neg\mathfrak{S}'}_{k2'}} \mathsf{conf}'_n$$
$$\wedge \quad \mathsf{conf}_{(k1+k1')} = \langle \mathcal{L}'', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC''', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\mathbb{DV}}_{k1'} \rangle \quad \wedge$$
$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{(\mathbb{DV}+\mathbb{DS})}_{k1'} :: \Pi^{\neg\mathfrak{S}'}_{k2'} \rangle$$

This lemma is proved by first applying the lemma on $\mathbb{DS}$ till it is not anymore applicable, and then applying the $\mathbb{DV}$ till it is not anymore possible. Once neither of the is anymore appliacable, then the interval for devices is ordered.

Lemma 5 ($\mathbb{CS}$ Interval) *The generic form of the previous lemma*

$$\forall wo, \mathcal{E}, k1, k2, n, \mathsf{conf}, \mathsf{conf}_{k1}, \mathsf{conf}_n, \mathcal{I}, HM, SS_r, CC_r, CC_e, CC'_e, IC, IC', IC'', \mathcal{L}, \mathcal{L}', \Pi^{\mathbb{IN}}, \Pi^{\mathfrak{S}}, \Pi^{\neg\mathfrak{S}}_{k2}.$$
$$|\Pi^{\mathbb{IN}}| + |\Pi^{\mathfrak{S}}| = k1 \quad \wedge \quad k1 + |\Pi^{\neg\mathfrak{S}}_{k2}| = n \quad \wedge \quad \mathsf{conf} = \langle [], \mathcal{I}, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,\emptyset\}, IC, [] \rangle \quad \wedge$$
$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}}_{k2}} \mathsf{conf}_n \quad \wedge$$
$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1} \rangle \quad \wedge \quad CC_r \neq \emptyset$$
$$\wedge \quad \mathsf{conf}_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\neg\mathfrak{S}}_{k2} \rangle \quad \wedge \quad |\mathcal{I}| = |CC_e| \quad \wedge \quad |CC'_e| < |\mathcal{I}|$$
$$\Rightarrow$$
$$\exists k1', k2', \mathsf{conf}_{(k1+k1')}, \mathsf{conf}'_n, SS_r, CC_c, CC_t, CC''_e, \Pi^{\mathbb{CS}}_{k1'}, \Pi^{\neg\mathfrak{S}'}_{k2'}.$$
$$|\Pi^{\neg\mathfrak{S}}_{k2}| = |\Pi^{\mathbb{CS}}_{k1'}| + |\Pi^{\neg\mathfrak{S}'}_{k2'}| \quad \wedge \quad |\Pi^{\neg\mathfrak{S}'}_{k2'}| \leq |\Pi^{\neg\mathfrak{S}}_{k2}| \quad \wedge \quad \mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k1'}_{\Pi^{\mathbb{CS}}_{k1'}} \mathsf{conf}_{(k1+k1')} \rightsquigarrow^{k2'}_{\Pi^{\neg\mathfrak{S}'}_{k2'}} \mathsf{conf}'_n$$
$$\wedge \quad \mathsf{conf}_{(k1+k1')} = \langle \mathcal{L}', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{\emptyset, CC_c, CC_t, CC''_e\}, IC'', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\mathbb{CS}}_{k1'} \rangle \quad \wedge$$
$$(|CC'_e| \leq |CC''_e| \leq |\mathcal{I}|) \quad \wedge \quad \mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\mathbb{CS}}_{k1'} :: \Pi^{\neg\mathfrak{S}'}_{k2'} \rangle$$

Lemma 6 ($\mathbb{CC}$ Interval) *The generic form of the previous lemma*

$$\forall \mathit{wo}, \mathcal{E}, k1, k2, n, \mathsf{conf}, \mathsf{conf}_{k1}, \mathsf{conf}_n, \mathcal{I}, HM, SS_r, CC_c, CC_t, CC_e, CC'_e, IC, IC', IC'', \mathcal{L}, \mathcal{L}', \Pi^{\mathbb{IN}}, \Pi^{\mathfrak{S}}, \Pi^{\neg\mathfrak{S}}_{k2}.$$
$$|\Pi^{\mathbb{IN}}| + |\Pi^{\mathfrak{S}}| = k1 \quad \wedge \quad k1 + |\Pi^{\neg\mathfrak{S}}_{k2}| = n \quad \wedge \quad \mathsf{conf} = \langle [], \mathcal{I}, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,\emptyset\}, IC, [] \rangle \quad \wedge$$
$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}}_{k2}} \mathsf{conf}_n \quad \wedge$$
$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{\emptyset, CC_c, CC_t, CC'_e\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1} \rangle \quad \wedge \quad CC_c \neq \emptyset \quad \wedge$$
$$\mathsf{conf}_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\neg\mathfrak{S}}_{k2} \rangle \quad \wedge \quad |\mathcal{I}| = |CC_e| \quad \wedge \quad |CC'_e| < |\mathcal{I}|$$
$$\Rightarrow$$
$$\exists k1', k2', \mathsf{conf}_{(k1+k1')}, \mathsf{conf}'_n, SS'_r, CC_r, CC'_t, CC''_e, \Pi^{\mathbb{CC}}_{k1'}, \Pi^{\neg\mathfrak{S}'}_{k2'}.$$
$$|\Pi^{\neg\mathfrak{S}}_{k2}| = |\Pi^{\mathbb{CC}}_{k1'}| + |\Pi^{\neg\mathfrak{S}'}_{k2'}| \quad \wedge \quad |\Pi^{\neg\mathfrak{S}'}_{k2'}| < |\Pi^{\neg\mathfrak{S}}_{k2}| \quad \wedge \quad \mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k1'}_{\Pi^{\mathbb{CC}}_{k1'}} \mathsf{conf}_{(k1+k1')} \rightsquigarrow^{k2'}_{\Pi^{\neg\mathfrak{S}'}_{k2'}} \mathsf{conf}'_n$$
$$\wedge \quad \mathsf{conf}_{(k1+k1')} = \langle \mathcal{L}', \emptyset, HM, \{SS'_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, CC'_t, CC''_e\}, IC'', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\mathbb{CC}}_{k1'} \rangle \quad \wedge$$
$$(|CC'_e| \leq |CC''_e| < |\mathcal{I}|) \quad \wedge \quad \mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\mathbb{CC}}_{k1'} :: \Pi^{\neg\mathfrak{S}'}_{k2'} \rangle$$

Lemma 7 ($\mathbb{RN}$ Interval) *The generic form of the previous lemma*

$$\forall \mathit{wo}, \mathcal{E}, k1, k2, n, \mathsf{conf}, \mathsf{conf}_{k1}, \mathsf{conf}_n, \mathcal{I}, HM, SS_r, CC_r, CC_t, CC_e, CC'_e, IC, IC', IC'', \mathcal{L}, \mathcal{L}', \Pi^{\mathbb{IN}}, \Pi^{\mathfrak{S}}, \Pi^{\neg\mathfrak{S}}_{k2}.$$
$$|\Pi^{\mathbb{IN}}| + |\Pi^{\mathfrak{S}}| = k1 \quad \wedge \quad k1 + |\Pi^{\neg\mathfrak{S}}_{k2}| = n \quad \wedge \quad \mathsf{conf} = \langle [], \mathcal{I}, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,\emptyset\}, IC, [] \rangle \quad \wedge$$
$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}}_{k2}} \mathsf{conf}_n \quad \wedge$$
$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, CC_t, CC'_e\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1} \rangle \quad \wedge \quad CC_t \neq \emptyset \quad \wedge$$
$$\mathsf{conf}_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\neg\mathfrak{S}}_{k2} \rangle \quad \wedge \quad |\mathcal{I}| = |CC_e| \quad \wedge \quad |CC'_e| < |\mathcal{I}|$$
$$\Rightarrow$$
$$\exists k1', k2', \mathsf{conf}_{(k1+k1')}, \mathsf{conf}'_n, SS'_r, CC'_r, CC''_e, \Pi^{\mathbb{RN}}_{k1'}, \Pi^{\neg\mathfrak{S}'}_{k2'}.$$
$$|\Pi^{\neg\mathfrak{S}}_{k2}| = |\Pi^{\mathbb{RN}}_{k1'}| + |\Pi^{\neg\mathfrak{S}'}_{k2'}| \quad \wedge \quad |\Pi^{\neg\mathfrak{S}'}_{k2'}| < |\Pi^{\neg\mathfrak{S}}_{k2}| \quad \wedge \quad \mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k1'}_{\Pi^{\mathbb{RN}}_{k1'}} \mathsf{conf}_{(k1+k1')} \rightsquigarrow^{k2'}_{\Pi^{\neg\mathfrak{S}'}_{k2'}} \mathsf{conf}'_n$$
$$\wedge \quad \mathsf{conf}_{(k1+k1')} = \langle \mathcal{L}', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{CC'_r, \emptyset, \emptyset, CC''_e\}, IC'', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\mathbb{RN}}_{k1'} \rangle \quad \wedge$$
$$(|CC'_e| \leq |CC''_e| < |\mathcal{I}|) \quad \wedge \quad \mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\mathbb{RN}}_{k1'} :: \Pi^{\neg\mathfrak{S}'}_{k2'} \rangle$$

Lemma 8 (Termination for finite traces) *If there are no active clients in* WEBIOT*, then no services are running.*

$$\forall k, n, \mathsf{conf}, \mathsf{conf}_n, SS_r, SS_a, SS_g, SS_e, CC_e, \mathcal{I}, , HM, IC, IC', \mathcal{L}, \Pi_n, \Pi_k, \Pi_{(n-k)}.$$
$$\mathsf{conf} \rightsquigarrow^{k}_{\Pi_k} \mathsf{conf}_k \rightsquigarrow^{(n-k)}_{\Pi_{(n-k)}} \mathsf{conf}_n \quad \wedge \quad \mathsf{conf} = \langle [], \mathcal{I}, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,\emptyset\}, IC, [] \rangle \quad \wedge$$
$$\mathsf{conf}_k = \langle \mathcal{L}, \emptyset, HM, \{SS_r, SS_a, SS_g, SS_e\}, \{CC_r, CC_c, CC_t, CC_e\}, IC', \Pi_k \rangle \quad \wedge \quad |CC_e| = |\mathcal{I}|$$
$$\Rightarrow$$
$$\mathsf{conf}_k = \langle \mathcal{L}, \emptyset, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,CC_e\}, IC', \Pi_k \rangle$$
$$\wedge \quad k = n \quad \wedge \quad \Pi_{(n-k)} = []$$

Lemma 9 (Trace commutativity($\mathbb{IN}$)) *Given* conf*, a well-formed WEBIOTconfiguration, if*

$$\mathsf{conf}\{\mathcal{I}\},\ \forall\ (j,u),(j',u') :\ ((j,u)\in\mathcal{I})\wedge((j',u')\in\mathcal{I})\wedge(j=j')\Rightarrow u=u' \tag{H1$_{\text{L9}}$}$$

$$\mathbb{X}\in\{\mathbb{IN},\mathbb{SS},\mathbb{RS},\mathbb{CS},\mathbb{CC},\mathbb{RN},\mathbb{DS}\} \tag{H2$_{\text{L9}}$}$$

$$\mathit{wo},\mathcal{E}\vdash\mathsf{conf}\rightsquigarrow_{\mathbb{X}}\mathsf{conf}''\rightsquigarrow_{\mathbb{IN}}\mathsf{conf}' \tag{H3$_{\text{L9}}$}$$

then

$$\exists\mathsf{conf}'''\,:\mathsf{wo},\mathcal{E}\vdash\mathsf{conf}\rightsquigarrow_{\mathbb{IN}}\mathsf{conf}'''\rightsquigarrow_{\mathbb{X}}\mathsf{conf}'$$

Proof 1

Lemma 10 (Trace commutativity($\mathbb{SS}$)) *Given* conf*, a well-formed WEBIOTconfiguration:*

$$\forall\ \langle sc\rangle^{((h,j),i)}\in SS :\ (\langle cc,B,T\rangle\in CC\wedge\ b^i\in B)\vee(\langle \texttt{boot}\rangle\in CC)$$
$$\wedge$$
$$\forall\ \mathsf{conf},j,i:\mathsf{wo},\mathcal{E}\vdash\mathsf{conf}\{SS\}\rightsquigarrow^{i,j}_{\mathbb{RS}}\mathsf{conf}'\{SS'\}\Rightarrow\langle sc\rangle^{((h,j),i)}\notin SS'$$
$$\wedge$$
$$\forall\ \mathsf{conf},j,i,h:(i\neq 0)\wedge(\mathsf{wo},\mathcal{E}\vdash\mathsf{conf}\{SS\}\rightsquigarrow^{j,0}_{\mathbb{RS}}\mathsf{conf}'\{SS'\})\Rightarrow\langle sc\rangle^{((h,j),i)}\notin SS'$$
$$\wedge$$
$$\mathbb{X}\in\{\mathbb{SS},\mathbb{RS},\mathbb{CS},\mathbb{CC},\mathbb{RN},\mathbb{DS}\}$$
$$\wedge$$
$$\exists\mathsf{conf}\{CC,SS\},cc',\forall j,i,(\{\langle cc,B,T\rangle^{(j,u)}\}\in CC\wedge cc\rightsquigarrow_{\overline{C}}^{?-,-^{\ i}}cc')\rightarrow(ss^{((\_,j),i)}\notin SS)$$
$$\wedge$$
$$\exists\ j,i:\ \forall\ \mathsf{conf}:\{\_,\{b^i\}\uplus B,\_\}^{(j,\_)}\in\mathsf{conf}.CC\Rightarrow ss^{((\_,j),i)}\notin SS$$
$$\wedge$$
$$\mathit{wo},\mathcal{E}\vdash\mathsf{conf}\rightsquigarrow_{\mathbb{X}}\mathsf{conf}_{i1}\rightsquigarrow^{j,i}_{\mathbb{SS}}\mathsf{conf}_1$$
$$\Rightarrow$$
$$\exists\mathsf{conf}_{i2}:\mathsf{wo},\mathcal{E}\vdash\mathsf{conf}\rightsquigarrow^{j,i}_{\mathbb{SS}}\mathsf{conf}_{i2}\rightsquigarrow_{\mathbb{X}}\mathsf{conf}_2\wedge(\mathsf{conf}_1=\mathsf{conf}_2)$$

.

Proof 2 *We first introduce the hypothesys of the lemma in the following.*

$$\forall\ \langle sc\rangle^{((h,j),i)}\in SS :\ (\langle cc,B,T\rangle\in CC\wedge\ b^i\in B)\vee(\langle \texttt{boot}\rangle\in CC) \tag{H1$_{\text{L10}}$}$$

$$\forall\ \mathsf{conf},j,i:\mathsf{wo},\mathcal{E}\vdash\mathsf{conf}\{SS\}\rightsquigarrow^{j,i}_{\mathbb{RS}}\mathsf{conf}'\{SS'\}\Rightarrow\langle sc\rangle^{((h,j),i)}\notin SS' \tag{H2$_{\text{L10}}$}$$

$$\forall\ \mathsf{conf},j,i,h:(\mathsf{wo},\mathcal{E}\vdash\mathsf{conf}\{SS\}\rightsquigarrow^{j,0}_{\mathbb{RS}}\mathsf{conf}'\{SS'\})\wedge(i\neq 0)\Rightarrow\langle sc\rangle^{((h,j),i)}\notin SS' \tag{H3$_{\text{L10}}$}$$

$$\mathbb{X}\in\{\mathbb{SS},\mathbb{RS},\mathbb{CS},\mathbb{CC},\mathbb{RN},\mathbb{DS}\} \tag{H4$_{\text{L10}}$}$$

$$\exists\mathsf{conf}\{CC,SS\},cc',\forall j,i,(\{\langle cc,B,T\rangle^{(j,u)}\}\in CC\wedge cc\rightsquigarrow_{\overline{C}}^{?-,-^{\ i}}cc')\Rightarrow(ss^{((\_,j),i)}\notin SS) \tag{H5$_{\text{L10}}$}$$

$$\exists\ j,i:\ \forall\ \mathsf{conf}:\{\_,\{b^i\}\uplus B,\_\}^{(j,\_)}\in\mathsf{conf}.CC\Rightarrow ss^{((\_,j),i)}\notin SS \tag{H6$_{\text{L10}}$}$$

$$\mathit{wo},\mathcal{E}\vdash\mathsf{conf}\rightsquigarrow_{\mathbb{X}}\mathsf{conf}_{i1}\rightsquigarrow^{j,i}_{\mathbb{SS}}\mathsf{conf}_1 \tag{H7$_{\text{L10}}$}$$

We proceed by case analysis on hypothesis H4$_{L10}$.

$\mathbb{X} = \mathbb{SS}$ *In this case we consider two consecutive* **ServerSteps**. *We rewrite hypothesys* $H7_{L10}$ *as*

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{SS}}^{k,l} \mathsf{conf}_{i1} \rightsquigarrow_{\mathbb{SS}}^{j,i} \mathsf{conf}_1 \tag{H8$_{\text{L10}}$}$$

We now analyze the possible values that k *and* l *may assume.*

$k = j \wedge l = i$ *In this case, we can consider hypothesis* $H8_{L10}$ *to be*

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{SS}}^{j,i} \mathsf{conf}_{i1} \tag{H9$_{\text{L10}}$}$$
$$wo, \mathcal{E} \vdash \mathsf{conf}_{i1} \rightsquigarrow_{\mathbb{SS}}^{j,i} \mathsf{conf}_1 \tag{H10$_{\text{L10}}$}$$

We do apply $\rightsquigarrow_{\mathbb{SS}}^{j,i}$ *on* conf, *obtaining the following knowledge.*

$$SS = \{\langle sc \rangle^{((h,j),i)}\} \uplus SS' \tag{H11$_{\text{L10}}$}$$
$$\langle sc, HM(h) \rangle \rightsquigarrow_S \langle sc', \mu'_h \rangle \tag{H12$_{\text{L10}}$}$$
$$HM' = HM[h \mapsto \mu'_h] \tag{H13$_{\text{L10}}$}$$
$$conf_{i1} = \langle \mathcal{L}, \mathcal{I}, HM', SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC, IC \rangle \tag{H14$_{\text{L10}}$}$$

Now, as we know what $conf_{i1}$ *is, we can apply again the* **ServerStep** *rule, this time on* conf_{i1}, *deriving the following.*

$$SS' \cup \{\langle sc' \rangle^{((h,j),i)}\} = \{\langle sc'' \rangle^{((h',j),i)}\} \uplus SS'' \tag{H15$_{\text{L10}}$}$$
$$\langle sc'', HM'(h') \rangle \rightsquigarrow_S \langle sc''', \mu'_{h'} \rangle \tag{H16$_{\text{L10}}$}$$
$$conf_1 = \langle \mathcal{L}, \mathcal{I}, HM'[h' \mapsto \mu'_{h'}], SS'' \cup \{\langle sc''' \rangle^{((h',j),i)}\}, CC, IC \rangle \tag{H17$_{\text{L10}}$}$$

As we know that a service is identified univoquely, we can apply $H4$ *and* $H3$ *to* $H15_{L10}$ *to derive:*

$$h = h' \tag{H18$_{\text{L10}}$}$$
$$sc' = sc'' \tag{H19$_{\text{L10}}$}$$
$$SS' = SS'' \tag{H20$_{\text{L10}}$}$$

We can the apply the new $H18_{L10}$, $H19_{L10}$, *and* $H20_{L10}$ *to* $H15_{L10}$, $H16_{L10}$, *and* $H17_{L10}$ *obtaining*

$$SS' \cup \{\langle sc' \rangle^{((h,j),i)}\} = \{\langle sc' \rangle^{((h,j),i)}\} \uplus SS' \tag{H21$_{\text{L10}}$}$$
$$\langle sc', HM'(h) \rangle \rightsquigarrow_S \langle sc'', \mu''_h \rangle \tag{H22$_{\text{L10}}$}$$
$$conf_1 = \langle \mathcal{L}, \mathcal{I}, HM'[h \mapsto \mu''_h], SS' \cup \{\langle sc'' \rangle^{((h,j),i)}\}, CC, IC \rangle \tag{H23$_{\text{L10}}$}$$

Now lets rewrite the goal of the lemma, for this subcase.

$$\exists \mathsf{conf}_{i2} : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{SS}}^{j,i} \mathsf{conf}_{i2} \rightsquigarrow_{\mathbb{SS}}^{j,i} \mathsf{conf}_2 \wedge (\mathsf{conf}_1 = \mathsf{conf}_2) \tag{SG1$_{\text{L10}}$}$$

Knowing that the semantics of the server language is deterministic $H1$, *and that the client is unique in* SS *and univoquely identified, respectively* $H4$ *and* $H3$, *we can say that it exists only a* conf_{i2} *such that* $(\mathsf{conf}_1 = \mathsf{conf}_2)$, *which is* $conf_{i1}$ *(*$H14_{L10}$*). is not necessary in this case, because it is just a sequential application of the ServerStep rule on the same running service, which its interpretation is deterministic. This Hypothesis is necessary in other cases, if not it becomes really complicated.*

$\neg(k = j \wedge l = i)$ *As* j *and* i *are joint identifiers for a running service, we can safely say that if at least one of them changes, then we are considering another service. We are trying to prove commutativity between* $\rightsquigarrow_{\mathbb{SS}}^{k,l}$ *and* $\rightsquigarrow_{\mathbb{SS}}^{j,i}$, *where* k *and* i *are one of those cases.*

- $k = j \wedge l \neq i$

- $k \neq j \wedge l = i$
- $k \neq j \wedge l \neq i$

We present the latter case, $k \neq j \wedge l \neq i$. Lets consider the hypothesis $H8_{L10}$ for this subcase.

$$\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k,l}_{\mathbb{SS}} \mathsf{conf}_{i1} \tag{H24$_{\text{L10}}$}$$

$$\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}_{i1} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_{1} \tag{H25$_{\text{L10}}$}$$

We first apply the `ServerStep` *rule to $H24_{L10}$ and then to $H25_{L10}$ deriving the following.*

$$SS = \{\langle sc_{kl}\rangle^{((h_{kl},k),l)}, \langle sc_{ji}\rangle^{((h_{ji},j),i)}\} \uplus SS' \tag{H26$_{\text{L10}}$}$$

$$\langle sc_{kl}, HM(h_{kl})\rangle \rightsquigarrow_S \langle sc'_{kl}, \mu'_{h_{kl}}\rangle \tag{H27$_{\text{L10}}$}$$

$$HM_{i1} = HM[h_{kl} \mapsto \mu'_{h_{kl}}] \tag{H28$_{\text{L10}}$}$$

$$\mathsf{conf}_{i1} = \langle \mathcal{L}, \mathcal{I}, HM_{i1}, SS' \cup \{\langle sc'_{kl}\rangle^{((h,k),l)}, \langle sc_{ji}\rangle^{((h_{ji},j),i)}\}, CC, IC\rangle \tag{H29$_{\text{L10}}$}$$

$$\langle sc_{ji}, HM_{i1}(h_{ji})\rangle \rightsquigarrow_S \langle sc'_{ji}, \mu'_{h_{ji}}\rangle \tag{H30$_{\text{L10}}$}$$

$$\mathsf{conf}_{1} = \langle \mathcal{L}, \mathcal{I}, HM_{i1}[h_{ji} \mapsto \mu'_{h_{ji}}], SS' \cup \{\langle sc'_{kl}\rangle^{((h_{kl},k),l)}, \langle sc'_{ji}\rangle^{((h_{ji},j),i)}\}, CC, IC\rangle \tag{H31$_{\text{L10}}$}$$

What we can say here is that, for each server language step, the host memory will never change. This is because of the hypothesis H2. This hold no matter what we do. We can rewrite the latter hypotheses as:

$$SS = \{\langle sc_{kl}\rangle^{((h_{kl},k),l)}, \langle sc_{ji}\rangle^{((h_{ji},j),i)}\} \uplus SS' \tag{H32$_{\text{L10}}$}$$

$$\langle sc_{kl}, HM(h_{kl})\rangle \rightsquigarrow_S \langle sc'_{kl}, \mu'_{h_{kl}}\rangle \tag{H33$_{\text{L10}}$}$$

$$\mu'_{h_{kl}} = HM(h_{kl}) \tag{H34$_{\text{L10}}$}$$

$$HM[h_{kl} \mapsto \mu'_{h_{kl}}] = HM \tag{H35$_{\text{L10}}$}$$

$$\mathsf{conf}_{i1} = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc'_{kl}\rangle^{((h,k),l)}, \langle sc_{ji}\rangle^{((h_{ji},j),i)}\}, CC, IC\rangle \tag{H36$_{\text{L10}}$}$$

$$\langle sc_{ji}, HM(h_{ji})\rangle \rightsquigarrow_S \langle sc'_{ji}, \mu'_{h_{ji}}\rangle \tag{H37$_{\text{L10}}$}$$

$$\mu'_{h_{ji}} = HM(h_{ji}) \tag{H38$_{\text{L10}}$}$$

$$HM[h_{ji} \mapsto \mu'_{h_{ji}}] = HM \tag{H39$_{\text{L10}}$}$$

$$\mathsf{conf}_{1} = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc'_{kl}\rangle^{((h_{kl},k),l)}, \langle sc'_{ji}\rangle^{((h_{ji},j),i)}\}, CC, IC\rangle \tag{H40$_{\text{L10}}$}$$

.

Now lets consider the goal for this subcase.

$$\exists \mathsf{conf}_{i2} : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k,l}_{\mathbb{SS}} \mathsf{conf}_{i2} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_{2} \wedge (\mathsf{conf}_{1} = \mathsf{conf}_{2})$$

We can split the goal in three subgoals:

$$\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_{i2} \tag{SG2$_{\text{L10}}$}$$

$$\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}_{i2} \rightsquigarrow^{k,l}_{\mathbb{SS}} \mathsf{conf}_{2} \tag{SG3$_{\text{L10}}$}$$

$$\mathsf{conf}_{1} = \mathsf{conf}_{2} \tag{SG4$_{\text{L10}}$}$$

We apply again the `ServerStep` *rule to $SG2_{L10}$ and then to $SG3_{L10}$ deriving the following.*

$$SS = \{\langle sc_{kl}\rangle^{((h_{kl},k),l)}, \langle sc_{ji}\rangle^{((h_{ji},j),i)}\} \uplus SS' \tag{SG5$_{\text{L10}}$}$$

$$\langle sc_{ji}, HM(h_{ji})\rangle \rightsquigarrow_S \langle sc'_{ji}, \mu'_{h_{ji}}\rangle \tag{SG6$_{\text{L10}}$}$$

$$HM_{i2} = HM[h_{ji} \mapsto \mu'_{h_{ji}}] \tag{SG7$_{\text{L10}}$}$$

$$\mathsf{conf}_{i2} = \langle \mathcal{L}, \mathcal{I}, HM_{i2}, SS' \cup \{\langle sc_{kl}\rangle^{((h,k),l)}, \langle sc'_{ji}\rangle^{((h_{ji},j),i)}\}, CC, IC\rangle \tag{SG8$_{\text{L10}}$}$$

$$\langle sc_{kl}, HM_{i2}(h_{kl})\rangle \rightsquigarrow_S \langle sc'_{kl}, \mu'_{h_{kl}}\rangle \tag{SG9$_{\text{L10}}$}$$

$$\mathsf{conf}_{2} = \langle \mathcal{L}, \mathcal{I}, HM_{i2}[h_{kl} \mapsto \mu'_{h_{kl}}], SS' \cup \{\langle sc'_{kl}\rangle^{((h_{kl},k),l)}, \langle sc'_{ji}\rangle^{((h_{ji},j),i)}\}, CC, IC\rangle \tag{SG10$_{\text{L10}}$}$$

- *By H4, H3 and $H32_{L10}$ we solve $SG5_{L10}$*
- *By H1, H2 and $H37_{L10}$ we solve $SG6_{L10}$*
- *By $H38_{L10}$ and $H39_{L10}$ we solve $SG7_{L10}$.*

$$HM_{i2} = HM \tag{H41$_{L10}$}$$

- *By H1, H2, $H33_{L10}$ and $H41_{L10}$ we solve $SG9_{L10}$*
- *By $H34_{L10}$, $H35_{L10}$ and $H41_{L10}$ we rewrite $SG10_{L10}$ as*

$$\mathsf{conf}_2 = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc'_{kl}\rangle^{((h_{kl},k),l)}, \langle sc'_{ji}\rangle^{((h_{ji},j),i)}\}, CC, IC\rangle$$

By substituting $H40_{L10}$ we obtain $\mathsf{conf}_2 = \mathsf{conf}_1 = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc'_{kl}\rangle^{((h_{kl},k),l)}, \langle sc'_{ji}\rangle^{((h_{ji},j),i)}\}, CC, IC\rangle$. We can apply $H40_{L10}$ deriving

$$\mathsf{conf}_2 \tag{H42$_{L10}$}$$
$$\mathsf{conf}_1 = \mathsf{conf}_2 \tag{H43$_{L10}$}$$

- *By applying $H43_{L10}$ we solve $SG4_{L10}$*
- *The intermediate configuration that make hold the equivalence is $SG8_{L10}$. We conclude by saying that $\mathbb{SS}$ can commute with any $\mathbb{SS}$.*

$\mathbb{X} = \mathbb{RS}$ *We rewrite the hypothesis $H7_{L10}$ as*

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k,l}_{\mathbb{RS}} \mathsf{conf}_{i1} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_1 \tag{H44$_{L10}$}$$

Now we reason on the possible assignments of k and l.

$k = j \wedge l = i$ *By hypothesis $H2_{L10}$ we know that the relation*

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,i}_{\mathbb{RS}} \mathsf{conf}_{i1} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_1$$

cannot commute to

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_{i1} \rightsquigarrow^{j,i}_{\mathbb{RS}} \mathsf{conf}_1$$

as the service does not run anymore once it has returned a value to the client.

$\neg(k = j \wedge l = i)$ *As we mentioned before, j and i identifies univoquely a running service. We can factorize the three cases in one. This time we prove the case $k = j \wedge l \neq i$. We first rewrite the hypothesis $H45_{L10}$ as*

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,l}_{\mathbb{RS}} \mathsf{conf}_{i1} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_1 \tag{H45$_{L10}$}$$

Now we apply the `RetService` *rule. As defines two of them, one for the service called by a booting client and one by a running client, we define two cases.*

$l = 0 \wedge l \neq i$ *In this case we know that the service has a booting client j that has a callback signed by 0 waiting for a result. We know this because of Hypothesis $H1_{L10}$. We apply* `RetServiceBoot` *to* conf*, deriving the following Hypotheses.*

$$\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, \{\langle v_s\rangle^{((h,j),0)}\} \uplus SS', \{\langle \mathsf{boot}\rangle^{(j,u)}\} \uplus CC', IC\rangle \tag{H46$_{L10}$}$$
$$\mathsf{gencc}(v_s) = cc \tag{H47$_{L10}$}$$
$$\mathsf{conf}_{i1} = \langle \mathcal{L}, \mathcal{I}, HM, SS', CC' \cup \{\langle cc, \emptyset, \emptyset\}\rangle^{(j,u)}\}, IC\rangle \tag{H48$_{L10}$}$$

We can directly solve this proof subtree as we know, because of Hypothesis $H3_{L10}$, that if a client is booting, it does not have any callbacks except for the one signed with 0. This means that it is not possible to immediately apply `ServerStep` *on conf_{i1}.*

$l \neq 0 \wedge l \neq i$ *In this case the running service has a running client callback waiting for the result of the service computation. We apply* `RetService` *to* conf*, deriving the following Hypotheses.*

$$\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, \{\langle v_s \rangle^{((h,j),l)}\} \uplus SS', \{\langle cc, B \cup \{b^l\}, T \rangle^{(j,u)}\} \uplus CC', IC \rangle \quad (\mathrm{H49}_{\mathrm{L10}})$$

$$\mathsf{gencc}(v_s) = v_c \quad (\mathrm{H50}_{\mathrm{L10}})$$

$$\mathsf{conf}_{i1} = \langle \mathcal{L}, \mathcal{I}, HM, SS', CC' \cup \{\langle cc, B, T \cup \{(b, v_c)^l\} \rangle^{(j,u)}\}, IC \rangle \quad (\mathrm{H51}_{\mathrm{L10}})$$

Now we can apply `ServerStep` *on* $conf_{i1}$ *obtaining the following.*

$$SS' = \{\langle sc \rangle^{((h,j),i)}\} \uplus SS'' \quad (\mathrm{H52}_{\mathrm{L10}})$$

$$\langle sc, HM(h) \rangle \rightsquigarrow_S \langle sc', \mu'_h \rangle \quad (\mathrm{H53}_{\mathrm{L10}})$$

$$\mathsf{conf}_1 = \langle \mathcal{L}, \mathcal{I}, HM[h \mapsto \mu'_h], SS'' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle cc, B, T \cup \{(b, v_c)^l\} \rangle^{(j,u)}\}, IC \rangle \quad (\mathrm{H54}_{\mathrm{L10}})$$

We can perform some transformations on te hypothesis. We know that the host memory never changes because of Hypotheses H1 and H2. We can also perform some rewriting to have a more clear vision on how the Set SS is modified.

$$\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, \{\langle v_s \rangle^{((h,j),l)}, \langle sc \rangle^{((h,j),i)}\} \uplus SS', \{\langle cc, B \cup \{b^l\}, T \rangle^{(j,u)}\} \uplus CC', IC \rangle \quad (\mathrm{H55}_{\mathrm{L10}})$$

$$\mathsf{gencc}(v_s) = v_c \quad (\mathrm{H56}_{\mathrm{L10}})$$

$$\mathsf{conf}_{i1} = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc \rangle^{((h,j),i)}\}, CC' \cup \{\langle cc, B, T \cup \{(b, v_c)^l\} \rangle^{(j,u)}\}, IC \rangle \quad (\mathrm{H57}_{\mathrm{L10}})$$

$$\langle sc, HM(h) \rangle \rightsquigarrow_S \langle sc', \mu'_h \rangle \quad (\mathrm{H58}_{\mathrm{L10}})$$

$$HM(h) = \mu'_h \quad (\mathrm{H59}_{\mathrm{L10}})$$

$$HM[h \mapsto \mu'_h] = HM \quad (\mathrm{H60}_{\mathrm{L10}})$$

$$\mathsf{conf}_1 = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle cc, B, T \cup \{(b, v_c)^l\} \rangle^{(j,u)}\}, IC \rangle \quad (\mathrm{H61}_{\mathrm{L10}})$$

Now we consider the following goal for this subcase.

$$\exists \mathsf{conf}_{i2} : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_{i2} \rightsquigarrow^{j,l}_{\mathbb{RS}} \mathsf{conf}_2 \wedge (\mathsf{conf}_1 = \mathsf{conf}_2)$$

We can split the goal in 3.

$$\mathit{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_{i2} \quad (\mathrm{SG11}_{\mathrm{L10}})$$

$$\mathit{wo}, \mathcal{E} \vdash \mathsf{conf}_{i2} \rightsquigarrow^{j,l}_{\mathbb{RS}} \mathsf{conf}_2 \quad (\mathrm{SG12}_{\mathrm{L10}})$$

$$\mathsf{conf}_1 = \mathsf{conf}_2 \quad (\mathrm{SG13}_{\mathrm{L10}})$$

We apply the `ServerStep` *and* `RetService` *on subgoals $SG11_{L10}$ and $SG12_{L10}$.*

$$\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, \{\langle sc \rangle^{((h,j),i)}, \langle v_s \rangle^{((h,j),l)}\} \uplus SS', \{\langle cc, B \cup \{b^l\}, T \rangle^{(j,u)}\} \uplus CC', IC \rangle \quad (\mathrm{SG14}_{\mathrm{L10}})$$

$$\langle sc, HM(h) \rangle \rightsquigarrow_S \langle sc', \mu'_h \rangle \quad (\mathrm{SG15}_{\mathrm{L10}})$$

$$\mathsf{conf}_{i2} = \langle \mathcal{L}, \mathcal{I}, HM[h \mapsto \mu'_h], SS' \cup \{\langle sc' \rangle^{((h,j),i)}, \langle v_s \rangle^{((h,j),l)}\}, CC, IC \rangle \quad (\mathrm{SG16}_{\mathrm{L10}})$$

$$\mathsf{gencc}(v_s) = v_c \quad (\mathrm{SG17}_{\mathrm{L10}})$$

$$conf_2 = \langle \mathcal{L}, \mathcal{I}, HM[h \mapsto \mu'_h], SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle cc, B, T \cup \{(b, v_c)^l\} \rangle^{(j,u)}\}, IC \rangle \quad (\mathrm{SG18}_{\mathrm{L10}})$$

The hypotheses matches the thesis. It follows that we can permute the two transitions withouth having any effect on the last derived configuration.

$k \neq j \ldots$ *Similar to the previous cases. Indipendent*

$\mathbb{X} = \mathbb{CS}$ *In this case we proof whether the client step and the service step commutes. We rewrite hypothesys $H7_{L10}$ as*

$$\mathit{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k}_{\mathbb{CS}} \mathsf{conf}_{i1} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_1 \quad (\mathrm{H62}_{\mathrm{L10}})$$

We now analyze the possible values that k may assume.

$k = j$ *The induction hypothesis is that k and j are equal. The Hypothesis $H7_{L10}$ can be rewritten as:*

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j}_{\mathbb{CS}} \mathsf{conf}_{i1} \tag{H63$_{\text{L10}}$}$$
$$wo, \mathcal{E} \vdash \mathsf{conf}_{i1} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_{1} \tag{H64$_{\text{L10}}$}$$

We apply the $\mathbb{CS}$ WEBIOTtransition to $H63_{L10}$, obtaining the following. We do split also SS in order to show that later on the $\mathbb{SS}$ can apply.

$$\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, \{\langle sc \rangle^{((h,j),i)}\} \uplus SS', \{\langle cc, B, T \rangle^{(j,u)}\} \uplus CC', IC \rangle \tag{H65$_{\text{L10}}$}$$
$$cc \rightsquigarrow_C cc' \tag{H66$_{\text{L10}}$}$$
$$\mathsf{conf}_{i1} = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc \rangle^{((h,j),i)}\}, \{\langle cc', B, T \rangle^{(j,u)}\} \uplus CC', IC \rangle \tag{H67$_{\text{L10}}$}$$

Now, knowing that it exists a service signed by j and i, we can apply the $\mathbb{SS}$ WEBIOTrule.

$$\langle sc, HM(h) \rangle \rightsquigarrow_S \langle sc', \mu'_h \rangle \tag{H68$_{\text{L10}}$}$$
$$\mathsf{conf}_1 = \langle \mathcal{L}, \mathcal{I}, HM[h \mapsto \mu'_h], SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle cc', B, T \rangle^{(j,u)}\}, IC \rangle \tag{H69$_{\text{L10}}$}$$

As we previously did, we have the global assumption that the host memory is never touched(H2). This means that we can introduce 2 new hypotheses, and reduce the $H72_{L10}$.

$$HM(h) = \mu'_h \tag{H70$_{\text{L10}}$}$$
$$HM[h \mapsto \mu'_h] = HM \tag{H71$_{\text{L10}}$}$$
$$\mathsf{conf}_1 = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle cc', B, T \rangle^{(j,u)}\}, IC \rangle \tag{H72$_{\text{L10}}$}$$

Now let's consider the goal. Can these relations permute? The goal for this subcase is the following.

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_{i2} \tag{SG19$_{\text{L10}}$}$$
$$wo, \mathcal{E} \vdash \mathsf{conf}_{i2} \rightsquigarrow^{j}_{\mathbb{CS}} \mathsf{conf}_{2} \tag{SG20$_{\text{L10}}$}$$
$$\mathsf{conf}_1 = \mathsf{conf}_2 \tag{SG21$_{\text{L10}}$}$$

Let's apply first the server step rule on $SG19_{L10}$. We split also on CC, to show that initially, $\langle cc, B, T \rangle^{(j,u)}$ exists.

$$\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, \{\langle sc \rangle^{((h,j),i)}\} \uplus SS', \{\langle cc, B, T \rangle^{(j,u)}\} \uplus CC', IC \rangle \tag{SG22$_{\text{L10}}$}$$
$$\langle sc, HM(h) \rangle \rightsquigarrow_S \langle sc', \mu'_h \rangle \tag{SG23$_{\text{L10}}$}$$
$$\mathsf{conf}_{i2} = \langle \mathcal{L}, \mathcal{I}, HM[h \mapsto \mu'_h], SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, \{\langle cc, B, T \rangle^{(j,u)}\} \uplus CC', IC \rangle \tag{SG24$_{\text{L10}}$}$$

Then we know that for H2, $SG24_{L10}$ can be rewritten.

$$HM(h) = \mu'_h \tag{SG25$_{\text{L10}}$}$$
$$HM[h \mapsto \mu'_h] = HM \tag{SG26$_{\text{L10}}$}$$
$$\mathsf{conf}_{i2} = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, \{\langle cc, B, T \rangle^{(j,u)}\} \uplus CC', IC \rangle \tag{SG27$_{\text{L10}}$}$$

Now we can apply the client step rule to $conf_{i2}$.

$$cc \rightsquigarrow_C cc' \tag{SG28$_{\text{L10}}$}$$
$$\mathsf{conf}_{i1} = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc \rangle^{((h,j),i)}\}, \{\langle cc', B, T \rangle^{(j,u)}\} \uplus CC', IC \rangle \tag{SG29$_{\text{L10}}$}$$

The hypotheses matches the sub-goals. proved.

$k \neq j$ *The issue might have been considering $k = j$. As, the other one is proved, this one follows.*

$\mathbb{X} = \mathbb{CC}$ *This case considers the commutativity between the* WEBIOTrule $\mathbb{CC}$ *and* $\mathbb{SS}$. *We rewrite hypothesys* $H7_{L10}$ *as*

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k}_{\mathbb{CC}} \mathsf{conf}_{i1} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_1 \tag{H73$_{\text{L10}}$}$$

We now analyze the possible values that k *may assume.*

$k = j$ *we apply* $\mathbb{CC}$ *on* conf.

$$\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, \{\langle sc \rangle^{((h,j),i)}\} \uplus SS', \{\langle cc, B, T \rangle^{(j,u)}\} \uplus CC', IC \rangle \tag{H74$_{\text{L10}}$}$$

$$cc \rightsquigarrow_C^{u'?v_a, b^{i'}} cc' \tag{H75$_{\text{L10}}$}$$

$$\texttt{WebServices}(u') = (\mathsf{serviceinit}, h') \tag{H76$_{\text{L10}}$}$$

$$sc_{v_a} = \mathsf{serviceinit}(\mathsf{v_a}) \tag{H77$_{\text{L10}}$}$$

$$\mathsf{conf}_{i1} = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc \rangle^{((h,j),i)}, sc_{v_a}^{((h',j),i')}\}, CC' \cup \{\langle cc', B \cup \{b^{i'}\}, T \rangle^{(j,u)}, IC \rangle \tag{H78$_{\text{L10}}$}$$

Then we can apply the server step rule to $conf_{i1}$.

$$\langle sc, HM(h) \rangle \rightsquigarrow_S \langle sc', \mu'_h \rangle \tag{H79$_{\text{L10}}$}$$

$$\mathsf{conf}_1 = \langle \mathcal{L}, \mathcal{I}, HM[h \mapsto \mu'_h], SS' \cup \{\langle sc' \rangle^{((h,j),i)}, sc_{v_a}^{((h',j),i')}\}, CC' \cup \{\langle cc', B \cup \{b^{i'}\}, T \rangle^{(j,u)}, IC \rangle \tag{H80$_{\text{L10}}$}$$

As we previously did, we have the global assumption that the host memory is never touched($H2$*). This means that we can introduce 2 new hypotheses, and reduce the* $H83_{L10}$.

$$HM(h) = \mu'_h \tag{H81$_{\text{L10}}$}$$

$$HM[h \mapsto \mu'_h] = HM \tag{H82$_{\text{L10}}$}$$

$$\mathsf{conf}_1 = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc' \rangle^{((h,j),i)}, sc_{v_a}^{((h',j),i')}\}, CC' \cup \{\langle cc', B \cup \{b^{i'}\}, T \rangle^{(j,u)}\}, IC \rangle \tag{H83$_{\text{L10}}$}$$

Now lets consider the goal.

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_{i2} \tag{SG30$_{\text{L10}}$}$$

$$wo, \mathcal{E} \vdash \mathsf{conf}_{i2} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_2 \tag{SG31$_{\text{L10}}$}$$

$$\mathsf{conf}_1 = \mathsf{conf}_2 \tag{SG32$_{\text{L10}}$}$$

We apply $\mathbb{SS}$ *to* conf.

$$\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, \{\langle sc \rangle^{((h,j),i)}\} \uplus SS', \{\langle cc, B, T \rangle^{(j,u)}\} \uplus CC', IC \rangle \tag{SG33$_{\text{L10}}$}$$

$$\langle sc, HM(h) \rangle \rightsquigarrow_S \langle sc', \mu'_h \rangle \tag{SG34$_{\text{L10}}$}$$

$$\mathsf{conf}_{i2} = \langle \mathcal{L}, \mathcal{I}, HM[h \mapsto \mu'_h], SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle cc, B, T \rangle^{(j,u)}\}, IC \rangle \tag{SG35$_{\text{L10}}$}$$

Because of $H2$, *we can rewrite the latter as.*

$$HM(h) = \mu'_h \tag{SG36$_{\text{L10}}$}$$

$$HM[h \mapsto \mu'_h] = HM \tag{SG37$_{\text{L10}}$}$$

$$\mathsf{conf}_{i2} = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle cc, B, T \rangle^{(j,u)}\}, IC \rangle \tag{SG38$_{\text{L10}}$}$$

Now we apply $\mathbb{CC}$ *to* conf_{i2}.

$$cc \rightsquigarrow_C^{u'?v_a, b^{i'}} cc' \tag{SG39$_{\text{L10}}$}$$

$$\texttt{WebServices}(u') = (\mathsf{serviceinit}, h') \tag{SG40$_{\text{L10}}$}$$

$$sc_{v_a} = \mathsf{serviceinit}(\mathsf{v_a}) \tag{SG41$_{\text{L10}}$}$$

$$\mathsf{conf}_2 = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc' \rangle^{((h,j),i)}, sc_{v_a}^{((h',j),i')}\}, CC' \cup \{\langle cc', B \cup \{b^{i'}\}, T \rangle^{(j,u)}\}, IC \rangle \tag{SG42$_{\text{L10}}$}$$

The subgoal match with the hypotheses. For this subcase, $\mathbb{CC}$ *commutes with* $\mathbb{SS}$.

$k \neq j$ *This case is trivial.* $\mathbb{CC}$ *commutes with* $\mathbb{SS}$.

$\mathbb{X} = \mathbb{RN}$ *This case considers the commutativity between the* WEBIOTrule $\mathbb{RN}$ *and* $\mathbb{SS}$. *We rewrite hypothesys* $H7_{L10}$ *as*

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k,l}_{\mathbb{RN}} \mathsf{conf}_{i1} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_1 \tag{H84$_{\text{L10}}$}$$

We now analyze the possible values that k *and* l *may assume.*

$k = j \wedge l = i$ *For Hypothesis* $H6_{L10}$, *this case cannot happen, as if we can run a client with* j *and* i, *it means that the service signed by the same indices has returned already a result.*

All the other cases *Lets pick a case. We consider the case* $k = j \wedge l \neq i$. *All the others follow the same pattern and result.*

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,l}_{\mathbb{RN}} \mathsf{conf}_{i1} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_1 \tag{H85$_{\text{L10}}$}$$

we apply $\mathbb{RN}$ *to* conf.

$$\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, \{\langle sc \rangle^{((h,j),i)}\} \uplus SS', \{\langle\langle v'_c, \mu_c\rangle, B, T \uplus \{(\lambda x.P, v_c)^l\}\rangle^{(j,u)}\} \uplus CC', IC\rangle \tag{H86$_{\text{L10}}$}$$

$$\mathsf{conf}_{i1} = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc \rangle^{((h,j),i)}\}, CC' \cup \{\langle\langle P\{x \mapsto v_c\}, \mu_c\rangle, B, T\rangle^{(j,u)}\}, IC\rangle \tag{H87$_{\text{L10}}$}$$

Now we apply $\mathbb{SS}$ *on* conf_{i1}.

$$\langle sc, HM(h)\rangle \rightsquigarrow_S \langle sc', \mu'_h\rangle \tag{H88$_{\text{L10}}$}$$

$$\mathsf{conf}_1 = \langle \mathcal{L}, \mathcal{I}, HM[h \mapsto \mu'_h], SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle\langle P\{x \mapsto v_c\}, \mu_c\rangle, B, T\rangle^{(j,u)}\}, IC\rangle \tag{H89$_{\text{L10}}$}$$

Because of Hypothesis $H2$*:*

$$HM(h) = \mu'_h \tag{H90$_{\text{L10}}$}$$

$$HM[h \mapsto \mu'_h] = HM \tag{H91$_{\text{L10}}$}$$

$$\mathsf{conf}_1 = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle\langle P\{x \mapsto v_c\}, \mu_c\rangle, B, T\rangle^{(j,u)}\}, IC\rangle \tag{H92$_{\text{L10}}$}$$

Now let consider the goal.

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_{i1} \rightsquigarrow^{j,l}_{\mathbb{RN}} \mathsf{conf}_1 \wedge \mathsf{conf}_1 = conf_2$$

First we apply the $\mathbb{SS}$ *on* conf.

$$\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, \{\langle sc \rangle^{((h,j),i)}\} \uplus SS', \{\langle\langle v'_c, \mu_c\rangle, B, T \uplus \{(\lambda x.P, v_c)^l\}\rangle^{(j,u)}\} \uplus CC', IC\rangle \tag{SG43$_{\text{L10}}$}$$

$$\langle sc, HM(h)\rangle \rightsquigarrow_S \langle sc', \mu'_h\rangle \tag{SG44$_{\text{L10}}$}$$

$$\mathsf{conf}_{i2} = \langle \mathcal{L}, \mathcal{I}, HM[h \mapsto \mu'_h], SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle\langle v'_c, \mu_c\rangle, B, T \uplus \{(\lambda x.P, v_c)^l\}\rangle^{(j,u)}\}, IC\rangle \tag{SG45$_{\text{L10}}$}$$

Again, because of Hypothesis $H2$*:*

$$HM(h) = \mu'_h \tag{SG46$_{\text{L10}}$}$$

$$HM[h \mapsto \mu'_h] = HM \tag{SG47$_{\text{L10}}$}$$

$$\mathsf{conf}_{i2} = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle\langle v'_c, \mu_c\rangle, B, T \uplus \{(\lambda x.P, v_c)^l\}\rangle^{(j,u)}\}, IC\rangle \tag{SG48$_{\text{L10}}$}$$

Then, we apply $\mathbb{RN}$ *on* conf_{i2}.

$$\mathsf{conf}_2 = \langle \mathcal{L}, \mathcal{I}, HM, SS' \cup \{\langle sc' \rangle^{((h,j),i)}\}, CC' \cup \{\langle\langle P\{x \mapsto v_c\}, \mu_c\rangle, B, T\rangle^{(j,u)}\}, IC\rangle \tag{SG49$_{\text{L10}}$}$$

The Hypotheses derived match the SGs. It follow that, $\mathsf{conf}_1 = \mathsf{conf}_2$.

$\mathbb{X} = \mathbb{DS}$ *As the* $\mathbb{DS}$ *and the* $\mathbb{SS}$ *rules touch disjoint partitions of an configuration* conf, *for any choice of an initial* conf. *We can directly say that the transitions* $wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{DS}} \mathsf{conf}_{i1} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_1$ *and* $wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,i}_{\mathbb{SS}} \mathsf{conf}_{i2} \rightsquigarrow_{\mathbb{DS}} \mathsf{conf}_2$ *are commutative. Hence,* conf_1 *is equivalent to* conf_2.

Lemma 11 (Trace commutativity($\mathbb{RS}$)) *Given* conf*, a well-formed* WEBIOT*configuration, if*

$$\forall \langle v_s \rangle^{((h,j),i)} \in SS: \ cc^{(j,u)} \in CC \Rightarrow (cc = \langle c, B, T \rangle \wedge \ b^i \in B) \vee (cc = \langle \texttt{boot} \rangle) \quad (\text{H1}_{\text{L11}})$$

$$\mathbb{X} \in \{\mathbb{RS}, \mathbb{CS}, \mathbb{CC}, \mathbb{RN}, \mathbb{DS}\} \quad (\text{H2}_{\text{L11}})$$

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}'' \rightsquigarrow^{j,i}_{\mathbb{RS}} \mathsf{conf}' \quad (\text{H3}_{\text{L11}})$$

then

$$\exists \mathsf{conf}''' : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j,i}_{\mathbb{RS}} \mathsf{conf}''' \rightsquigarrow_{\mathbb{X}} \mathsf{conf}'' \wedge (\mathsf{conf}' = \mathsf{conf}'')$$

Proof 3

Lemma 12 (Trace commutativity($\mathbb{DS}$)) *Given* conf*, a well-formed* WEBIOT*configuration, if*

$$\mathbb{X} \in \{\mathbb{CS}, \mathbb{CC}, \mathbb{RN}, \mathbb{DS}\} \quad (\text{H1}_{\text{L12}})$$

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}'' \rightsquigarrow_{\mathbb{DS}} \mathsf{conf}' \quad (\text{H2}_{\text{L12}})$$

then

$$\exists \mathsf{conf}''' : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{DS}} \mathsf{conf}''' \rightsquigarrow_{\mathbb{X}} \mathsf{conf}'' \wedge (\mathsf{conf}' = \mathsf{conf}'')$$

Proof 4

Lemma 13 (Trace commutativity($\mathbb{DR}$ & $\mathbb{DA}$)) *Given* conf*, a well-formed* WEBIOT*configuration:*

$$\begin{gathered}
\forall \langle ss \rangle^{((h,j),i)} \in SS: \ cc^{(j,u)} \in CC \Rightarrow (cc = \langle c, B, T \rangle \wedge \ b^i \in B) \vee (cc = \langle \texttt{boot} \rangle) \\
\wedge \\
\mathbb{X} \in \{\mathbb{CS}, \mathbb{CC}, \mathbb{RN}\} \\
\wedge \\
wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}'' \rightsquigarrow^{j,i}_{\mathbb{DR}} \mathsf{conf}' \\
\wedge \\
wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}'' \rightsquigarrow^{j,i}_{\mathbb{DA}} \mathsf{conf}' \\
\Rightarrow \\
\exists \mathsf{conf}''' : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{\mathsf{conf}''' \rightsquigarrow_{\mathbb{X}} \mathsf{conf}''}_{\mathbb{DR}_{j,i}} \ \wedge (\mathsf{conf}' = \mathsf{conf}'') \\
\vee \\
\exists \mathsf{conf}''' : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{\mathsf{conf}''' \rightsquigarrow_{\mathbb{X}} \mathsf{conf}''}_{\mathbb{DA}_{j,i}} \ \wedge (\mathsf{conf}' = \mathsf{conf}'')
\end{gathered}$$

$$\forall \langle ss \rangle^{((h,j),i)} \in SS: \ cc^{(j,u)} \in CC \Rightarrow (cc = \langle c, B, T \rangle \wedge \ b^i \in B) \vee (cc = \langle \texttt{boot} \rangle) \quad (\text{H1}_{\text{L13}})$$

$$\mathbb{X} \in \{\mathbb{CS}, \mathbb{CC}, \mathbb{RN}\} \quad (\text{H2}_{\text{L13}})$$

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}'' \rightsquigarrow^{j,i}_{\mathbb{DR}} \mathsf{conf}' \quad (\text{H3}_{\text{L13}})$$

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}'' \rightsquigarrow^{j,i}_{\mathbb{DA}} \mathsf{conf}' \quad (\text{H4}_{\text{L13}})$$

then

$$\begin{gathered}
\exists \mathsf{conf}''' : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{\mathsf{conf}''' \rightsquigarrow_{\mathbb{X}} \mathsf{conf}''}_{\mathbb{DR}_{j,i}} \ \wedge (\mathsf{conf}' = \mathsf{conf}'') \\
\vee \\
\exists \mathsf{conf}''' : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{\mathsf{conf}''' \rightsquigarrow_{\mathbb{X}} \mathsf{conf}''}_{\mathbb{DA}_{j,i}} \ \wedge (\mathsf{conf}' = \mathsf{conf}'')
\end{gathered}$$

Proof 5

Lemma 14 (Trace commutativity($\mathbb{CS}$)) *Given* $\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, SS, CC, IC \rangle$, *if*

$$\mathbb{X} \in \{\mathbb{CS}, \mathbb{CC}, \mathbb{RN}\}$$
$$\wedge$$
$$\mathit{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}_{i1} \rightsquigarrow^{j}_{\mathbb{CS}} \mathsf{conf}_1$$
$$\Rightarrow$$
$$\exists \mathsf{conf}_{i2} : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j}_{\mathbb{CS}} \mathsf{conf}_{i2} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}_2 \wedge (\mathsf{conf}_1 = \mathsf{conf}_2)$$

Proof 6

Lemma 15 (Trace commutativity($\mathbb{CC}$)) *Given* $\mathsf{conf} = \langle \mathcal{L}, \mathcal{I}, HM, SS, CC, IC \rangle$, *if*

$$\mathbb{X} \in \{\mathbb{CC}, \mathbb{RN}\}$$
$$\wedge$$
$$(\mathit{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k,i}_{\mathbb{RN}} \mathsf{conf}_{i1} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_1 \Rightarrow j \neq k)$$
$$\wedge$$
$$\mathit{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}_{i1} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_1$$
$$\Rightarrow$$
$$\exists \mathsf{conf}_{i2} : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_{i2} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}_2 \wedge (\mathsf{conf}_1 = \mathsf{conf}_2)$$

Proof 7 *Hypothesis of the lemma are:*

$$\mathbb{X} \in \{\mathbb{CC}, \mathbb{RN}\} \tag{H1$_{\text{L15}}$}$$
$$\mathit{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}_{i1} \rightsquigarrow^{j}_{\mathbb{CS}} \mathsf{conf}_1 \tag{H2$_{\text{L15}}$}$$
$$\mathit{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{X}} \mathsf{conf}_{i1} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_1 \tag{H3$_{\text{L15}}$}$$

Let's proceed by strong induction on the shape of $\mathbb{X}$.

$\mathbb{X} = \mathbb{CC}$ *We first rewrite the hypothesis H3$_{L15}$ as*

$$\mathit{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k}_{\mathbb{CC}} \mathsf{conf}_{i1} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_1$$

We proceed by analyzing the value of k. *We distinguish two cases, the first has* $k = j$ *and the other has* $k \neq j$.

$k = j$ *By* $k = j$, *we can split the hypothesis H3$_{L15}$ as*

$$\mathit{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_{i1} \tag{H4$_{\text{L15}}$}$$
$$\mathit{wo}, \mathcal{E} \vdash \mathsf{conf}_{i1} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_1 \tag{H5$_{\text{L15}}$}$$

. *By applying the* `ClientCall` *rule to the hypothesis* $H22_{L15}$*, and by knowing that a client is unique(H6) we derive:*

$$CC = \{\langle cc, B, T\rangle^{(j,u)}\} \uplus CC' \tag{H6$_{L15}$}$$
$$cc \rightsquigarrow_{C}^{u'?v_a,b^i} cc' \tag{H7$_{L15}$}$$
$$\texttt{WebServices}(u') = (\mathsf{serviceinit}, h) \tag{H8$_{L15}$}$$
$$sc = \mathsf{serviceinit}(\mathsf{v_a}) \tag{H9$_{L15}$}$$
$$B' = B \cup \{b^i\} \tag{H10$_{L15}$}$$
$$CC'' = CC' \cup \{\langle cc', B', T\rangle^{(j,u)}\} \tag{H11$_{L15}$}$$
$$SS' = SS \cup \{sc^{((h,j),i)}\} \tag{H12$_{L15}$}$$
$$conf_{i1} = \langle \mathcal{L}, \mathcal{I}, HM, SS', CC'', IC\rangle \tag{H13$_{L15}$}$$

Then, by applying `ClientCall` *rule to* $conf_{i1}$ *we derive the following*

$$CC'' = \{\langle cc', B', T\rangle^{(j,u)}\} \uplus CC' \tag{H14$_{L15}$}$$
$$cc' \rightsquigarrow_{C}^{u''?v''_a,b'^{i'}} cc'' \tag{H15$_{L15}$}$$
$$\texttt{WebServices}(u'') = (\mathsf{serviceinit}', h') \tag{H16$_{L15}$}$$
$$sc' = \mathsf{serviceinit}'(\mathsf{v}''_\mathsf{a}) \tag{H17$_{L15}$}$$
$$B'' = B' \cup \{b'^{i'}\} \tag{H18$_{L15}$}$$
$$CC''' = CC' \cup \{\langle cc'', B'', T\rangle^{(j,u)}\} \tag{H19$_{L15}$}$$
$$SS'' = SS' \cup \{sc''^{((h',j),i')}\} \tag{H20$_{L15}$}$$
$$conf_1 = \langle \mathcal{L}, \mathcal{I}, HM, SS'', CC''', IC\rangle \tag{H21$_{L15}$}$$

The result of these derivaitons on the Hypotesis $H3_{L15}$*, produce a final configuration presented as* $H21_{L15}$*. Notice that we do not consider* Π *as member of the configuration as it is an artifact for storing a trace of the evaluation.*

Now, lets consider the goal for this subcase

$$\exists \mathsf{conf}_{i2} : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_{i2} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_2 \wedge (\mathsf{conf}_1 = \mathsf{conf}_2)$$

. *If we consider the left proposition in the conjuction,* $\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_{i2} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_2$*, if we apply ClientCall, knowing that the client semantics is deterministic by H5 and a client* j *is unique by H6, we get the same hypothesis derived from the* $H3_{L15}$*. Hence,* $conf_{i1} = conf_{i2}$ *and* $conf_1 = conf_2$*.*

$k \neq j$ *Because of* $k \neq j$ *and* $\mathbb{X} = \mathbb{CC}$*, we can write* $H3_{L15}$ *as*

$$\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k}_{\mathbb{CC}} \mathsf{conf}_{i1} \tag{H22$_{L15}$}$$
$$\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf}_{i1} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_1 \tag{H23$_{L15}$}$$

.

As in the previous sub-case, we destructure the hypothesis $H3_{L15}$ *by applying the* `ClientCall` *rule, first to*

the client k and then to client j. We derive the following hypothesis.

$$CC = \{\langle cc_k, B_k, T_k\rangle^{(k,u_k)}, \langle cc_j, B'_j, T_j\rangle^{(j,u_j)}\} \uplus CC' \tag{H24$_{\text{L15}}$}$$

$$cc_k \rightsquigarrow_C^{u'_k ? v_k, b_k^{i_k}} cc'_k \tag{H25$_{\text{L15}}$}$$

$$\texttt{WebServices}(u'_k) = (\mathsf{serviceinit}_k, h_k) \tag{H26$_{\text{L15}}$}$$

$$sc_k = \mathsf{serviceinit_k(v_k)} \tag{H27$_{\text{L15}}$}$$

$$cc_j \rightsquigarrow_C^{u'_j ? v_j, b_j^{i_j}} cc'_j \tag{H28$_{\text{L15}}$}$$

$$\texttt{WebServices}(u'_j) = (\mathsf{serviceinit}_j, h_j) \tag{H29$_{\text{L15}}$}$$

$$sc_j = \mathsf{serviceinit_j(v_j)} \tag{H30$_{\text{L15}}$}$$

$$\begin{aligned} conf_1 = \langle \mathcal{L}, \mathcal{I}, HM, SS \cup \{sc_k^{((h_k,k),i_k)}, sc_j^{((h_j,j),i_j)}\}, \\ CC' \cup \{\langle cc'_k, B_k \cup \{b_k^{i_k}\}, T_k\rangle^{(k,u_k)}, \langle cc'_j, B_j \cup \{b_j^{i_j}\}, T_j\rangle^{(j,u_j)}\}, IC\rangle \end{aligned} \tag{H31$_{\text{L15}}$}$$

Now lets consider the goal

$$\exists \mathsf{conf}_{i2} : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{CC}}^{j} \mathsf{conf}_{i2} \rightsquigarrow_{\mathbb{CC}}^{k} \mathsf{conf}_2 \wedge (\mathsf{conf}_1 = \mathsf{conf}_2)$$

. We can apply twice the `ClientCall` *to the left proposition of the goal, first on the client j and then on the client k. We derive the following*

$$CC = \{\langle cc_j, B'_j, T_j\rangle^{(j,u_j)}, \langle cc_k, B_k, T_k\rangle^{(k,u_k)}\} \uplus CC' \tag{SG1$_{\text{L15}}$}$$

$$cc_j \rightsquigarrow_C^{u'_j ? v_j, b_j^{i_j}} cc'_j \tag{SG2$_{\text{L15}}$}$$

$$\texttt{WebServices}(u'_j) = (\mathsf{serviceinit}_j, h_j) \tag{SG3$_{\text{L15}}$}$$

$$sc_j = \mathsf{serviceinit_j(v_j)} \tag{SG4$_{\text{L15}}$}$$

$$cc_k \rightsquigarrow_C^{u'_k ? v_k, b_k^{i_k}} cc'_k \tag{SG5$_{\text{L15}}$}$$

$$\texttt{WebServices}(u'_k) = (\mathsf{serviceinit}_k, h_k) \tag{SG6$_{\text{L15}}$}$$

$$sc_k = \mathsf{serviceinit_k(v_k)} \tag{SG7$_{\text{L15}}$}$$

$$\begin{aligned} conf_2 = \langle \mathcal{L}, \mathcal{I}, HM, SS \cup \{sc_j^{((h_j,j),i_j)}, sc_k^{((h_k,k),i_k)}\}, \\ CC' \cup \{\langle cc'_j, B_j \cup \{b_j^{i_j}\}, T_j\rangle^{(j,u_j)}, \langle cc'_k, B_k \cup \{b_k^{i_k}\}, T_k\rangle^{(k,u_k)}\}, IC\rangle \end{aligned} \tag{SG8$_{\text{L15}}$}$$

. Knowing that if it exists, the client is unique(H6), and that the client language semantics is deterministic(H5), we can apply to the subgoals $SG2_{L15}$ and $SG5_{L15}$ respectively $H25_{L15}$ and $H28_{L15}$. Then, by $H24_{L15}$ and H6 we solve the subgoal $SG1_{L15}$. Basically the all subgoals and the hypothesis concide. Finally, we can say that $conf_1 = conf_2$ as they both are

$$\langle \mathcal{L}, \mathcal{I}, HM, SS \cup \{sc_j^{((h_j,j),i_j)}, sc_k^{((h_k,k),i_k)}\}, CC' \cup \{\langle cc'_j, B_j \cup \{b_j^{i_j}\}, T_j\rangle^{(j,u_j)}, \langle cc'_k, B_k \cup \{b_k^{i_k}\}, T_k\rangle^{(k,u_k)}\}, IC\rangle$$

. The configuration $conf_{i2}$ exists and it is

$$\langle \mathcal{L}, \mathcal{I}, HM, SS \cup \{sc_j^{((h_j,j),i_j)}\}, CC' \cup \{\langle cc'_j, B_j \cup \{b_j^{i_j}\}, T_j\rangle^{(j,u_j)}\}, IC\rangle$$

.

$\mathbb{X} = \mathbb{RN}$ *We first rewrite the hypothesis $H3_{L15}$ as*

$$\mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow_{\mathbb{RN}}^{k,i} \mathsf{conf}_{i1} \rightsquigarrow_{\mathbb{CC}}^{j} \mathsf{conf}_1 \tag{H32$_{\text{L15}}$}$$

We proceed by analyzing the value of k. We distinguish two cases, the first has $k = j$ and the other has $k \neq j$.

$j = k$ *This case cannot happen, by hypothesis* $H2_{L15}$. *The latter has not been artificially defined. It does emerge from evolution.*

$j \neq k$ *We first rewrite the Hypothesis* $H32_{L15}$ *as*

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k,i}_{\mathbb{RN}} conf_{i1} \tag{H33$_{L15}$}$$

$$wo, \mathcal{E} \vdash \mathsf{conf}_{i1} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_1 \tag{H34$_{L15}$}$$

, for some $k \neq j$*, and some* i*. We can derive the following, by applying the* **Run** *rule on* $H33_{L15}$*, and by applying the* `ClientCall` *rule on* $H34_{L15}$.

$$CC = \{\langle\langle v'_c, \mu\rangle, B_k, T_k \uplus \{(\lambda x.P, v_c)^i\}\rangle^{(k,u_k)}, \langle cc_j, B_j, T_j\rangle^{(j,u_j)}\} \uplus CC' \tag{H35$_{L15}$}$$

$$cc_k = \langle P\{x \mapsto v_c\}, \mu\rangle \tag{H36$_{L15}$}$$

$$cc_j \rightsquigarrow_C^{u'_j ? v_a, b^{i_j}} cc'_j \tag{H37$_{L15}$}$$

$$\texttt{WebServices}(u'_j) = (\mathsf{serviceinit}, h) \tag{H38$_{L15}$}$$

$$sc = \mathsf{serviceinit}(\mathsf{v_a}) \tag{H39$_{L15}$}$$

$$conf_1 = \langle \mathcal{L}, \mathcal{I}, HM, SS \cup \{sc^{((h,j),i_j)}\}, CC' \cup \{\langle cc_k, B_k, T_k\rangle^{(k,u_k)}, \langle cc'_j, B_j \cup \{b^{i_j}\}, T_j\rangle^{(j,u_j)}\}, IC\rangle \tag{H40$_{L15}$}$$

Now, lets consider the goal for this subcase. We can rewrite it as

$$\exists \mathsf{conf}_{i2} : \mathsf{wo}, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j}_{\mathbb{CC}} \mathsf{conf}_{i2} \rightsquigarrow^{k,i}_{\mathbb{RN}} \mathsf{conf}_2 \wedge (\mathsf{conf}_1 = \mathsf{conf}_2)$$

, and if we consider the left-most proposition, it can be split as:

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{j}_{\mathbb{CC}} conf_{i2} \tag{SG9$_{L15}$}$$

$$wo, \mathcal{E} \vdash \mathsf{conf}_{i2} \rightsquigarrow^{k,i}_{\mathbb{RN}} \mathsf{conf}_2 \tag{SG10$_{L15}$}$$

We can consecutively apply the rules `ClientCall` *and* **Run**, *respectively to* $SG9_{L15}$ *and* $SG10_{L15}$, *resulting in the following.*

$$CC = \{\langle cc_j, B_j, T_j\rangle^{(j,u_j)}, \langle\langle v'_c, \mu\rangle, B_k, T_k \uplus \{(\lambda x.P, v_c)^i\}\rangle^{(k,u_k)}\} \uplus CC' \tag{SG11$_{L15}$}$$

$$cc_j \rightsquigarrow_C^{u'_j ? v_a, b^{i_j}} cc'_j \tag{SG12$_{L15}$}$$

$$\texttt{WebServices}(u'_j) = (\mathsf{serviceinit}, h) \tag{SG13$_{L15}$}$$

$$sc = \mathsf{serviceinit}(\mathsf{v_a}) \tag{SG14$_{L15}$}$$

$$cc_k = \langle P\{x \mapsto v_c\}, \mu\rangle \tag{SG15$_{L15}$}$$

$$conf_2 = \langle \mathcal{L}, \mathcal{I}, HM, SS \cup \{sc^{((h,j),i_j)}\}, CC' \cup \{\langle cc_k, B_k, T_k\rangle^{(k,u_k)}, \langle cc'_j, B_j \cup \{b^{i_j}\}, T_j\rangle^{(j,u_j)}\}, IC\rangle \tag{SG16$_{L15}$}$$

- *By* $H6$ *and* $H35_{L15}$ *we solve* $SG11_{L15}$
- *By* $H5$ *and* $H36_{L15}$ *we solve* $SG12_{L15}$
- *By* $H38_{L15}$ *we solve* $SG13_{L15}$
- *By* $H39_{L15}$ *we solve* $SG14_{L15}$
- *By* $H36_{L15}$ *we solve* $SG15_{L15}$
- *Lets consider the goal* $conf_1 = conf_2$
  - *We know* $conf_1$*, and its structure, by* $H40_{L15}$

– *By* $SG16_{L15}$ *on* $conf_1 = conf_2$ *we see that:*

$$conf_1$$
$$=$$
$$\langle \mathcal{L}, \mathcal{I}, HM, SS \cup \{sc^{((h,j),i_j)}\}, CC' \cup \{\langle cc_k, B_k, T_k\rangle^{(k,u_k)}, \langle cc'_j, B_j \cup \{b^{i_j}\}, T_j\rangle^{(j,u_j)}\}, IC\rangle$$
$$=$$
$$\langle \mathcal{L}, \mathcal{I}, HM, SS \cup \{sc^{((h,j),i_j)}\}, CC' \cup \{\langle cc_k, B_k, T_k\rangle^{(k,u_k)}, \langle cc'_j, B_j \cup \{b^{i_j}\}, T_j\rangle^{(j,u_j)}\}, IC\rangle$$
$$=$$
$$conf_2$$

. *The equivalence holds. We derive the following.*

$$conf_1 = conf_2 \tag{H41$_{\text{L15}}$}$$

- *By* $H41_{L15}$ *and* $H40_{L15}$ *we solve* $SG16_{L15}$

Because of $conf_1 = conf_2$ *(*$H41_{L15}$*), we know that* $conf_{i2}$ *exists, and it is*

$$\langle \mathcal{L}, \mathcal{I}, HM, SS \cup \{sc^{((h,j),i_j)}\}, CC' \cup \{\langle cc'_j, B_j \cup \{b^{i_j}\}, T_j\rangle^{(j,u_j)}\}, IC\rangle$$

.

Theorem 1 *Given a well-formed initial configuration* conf*, assuming that*

- *the server language semantics is deterministic*

$$\begin{aligned}\forall\langle\langle sc,\mu_{sc}\rangle,\mu_h\rangle:\ (\langle\langle sc,\mu_{sc}\rangle,\mu_h\rangle \rightsquigarrow_S \langle\langle sc',\mu'_{sc}\rangle,\mu'_h\rangle) \Rightarrow \\ (\langle\langle sc,\mu_{sc}\rangle,\mu_h\rangle \rightsquigarrow_S \langle\langle sc'',\mu''_{sc}\rangle,\mu''_h\rangle) \wedge (sc'=sc'') \wedge (\mu'_{sc}=\mu''_{sc}) \wedge (\mu'_h=\mu''_h)\end{aligned} \tag{H1}$$

- *the host memory is never touched by the execution*

$$\begin{aligned}\forall\langle sc,\mu_h\rangle:\ (\langle sc,\mu_h\rangle \rightsquigarrow_S \langle sc',\mu'_h\rangle) \Rightarrow \\ (\mu_h=\mu'_h)\end{aligned} \tag{H2}$$

- *a service is identified univoquely*

$$\begin{aligned}\forall\langle sc\rangle^{((h,j),i)},\langle sc'\rangle^{((h',k),l)} \in SS: j=k \wedge i=l \Rightarrow \\ h=h' \wedge sc=sc'\end{aligned} \tag{H3}$$

- *a running service is unique*

$$\begin{aligned}\forall\langle sc\rangle^{((h,j),i)} \in SS, sc', h': SS=\langle sc\rangle^{((h,j),i)} \uplus SS' \Rightarrow \\ \langle sc'\rangle^{((h',j),i)} \notin SS'\end{aligned} \tag{H4}$$

- *the client semantics is deterministic*

$$\forall cc:\ (cc \rightsquigarrow_C cc') \Rightarrow (cc \rightsquigarrow_C cc'') \wedge (cc'=cc'') \tag{H5}$$

- *A client* j *is unique*

$$\forall\langle cc,\_,\_\rangle^{(j,u)} \in CC: (\langle cc',\_,\_\rangle^{(j',u')} \in CC) \Rightarrow (j \neq j') \tag{H6}$$

then

$$\begin{aligned}&\forall\ \mathit{wo},\mathcal{E},n,\Pi_n,\mathsf{conf}_n,\mathsf{conf}'_n.\ \exists\ t: \\ &\qquad \mathit{wo},\mathcal{E} \vdash \mathsf{conf}\{[]\} \rightsquigarrow^n \mathsf{conf}_n\{\Pi_n\} \\ &\qquad\qquad \Leftrightarrow \\ &(\mathit{wo},\mathcal{E} \vdash [\![\mathsf{conf}]\!]_{\mathbb{IN}}^{\pi_{\mathfrak{L}}(\Pi_n),\pi_{\mathfrak{T}}(\Pi_n)} \rightsquigarrow_{\mathfrak{S}}^{(n+t)} [\![\mathsf{conf}'_{(n)}]\!]_{\mathtt{end}}^{[],[]}) \ \wedge\ (\mathsf{conf}'_n = \mathsf{conf}_n)\end{aligned}$$

Proof 8 *ohhh withouth anything written here, the proof tag goes crazy..*

$\Leftarrow$ *This is trivial. The non-deterministic semantics can always imitate a scheduled* WEBIOT*step. Indeed, if a* WEBIOT*transition happens in the scheduled* WEBIOT*, then it means that it is possible to do also for the non-deterministic* WEBIOT*.*

$\Rightarrow$ *We proceed by first transforming the trace* Π_n *to a re-arranged(partially-ordered) trace* Π'_n *we show equivalent. Then we prove the* Π'_n *can be executed by both the* WEBIOT*and the scheduler semantics. They both terminate with identical final configurations. The latter are equivalent also to the final configuration of a* WEBIOT*execution that follow the trace* Π_n*. Let's consider the hypothesis for* $\Rightarrow$*.*

$$\mathit{wo},\mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^n \mathsf{conf}_n \tag{217}$$

$$\mathsf{conf} = \langle [],\mathcal{I},HM,\{\emptyset,\emptyset,\emptyset,\emptyset\},\{\emptyset,\emptyset,\emptyset,\emptyset\},IC,[]\rangle \tag{218}$$

$$\mathsf{conf}_n = \langle \mathcal{L},[],HM,\{\emptyset,\emptyset,\emptyset,\emptyset\},\{\emptyset,\emptyset,\emptyset,CC_e\},IC',\Pi_n\rangle \tag{219}$$

We proof almost algorithmically the trace permutation, Indeed, we first show the first iteration, the we prove correctness and termination of the further iterations.

First iteration

Step 0 *Depending on the cardinality of $\mathcal{I}$, we can have two cases.*

$\mathcal{I} = \emptyset$ *In this case we can apply the termination lemma 8 to the Hypotheses 2, 2, 2, $|\mathcal{I}| = |CC_e|$. We know that* $\mathsf{conf} = \mathsf{conf}_n$.

$\mathcal{I} \neq \emptyset$ *We know by Lemma 1 that if we consider a* WEBIOT*execution trace represeted in Hypothesis 2, we can manipulate it, moving all the inits in the beginning of the trace, re-execute* WEBIOT*the new trace Π'_n, resulting in a new final configuration conf'_n, where $\mathsf{conf}_n = \mathsf{conf}'_n$. So, we apply this lemma to the Hypotheses 2, 2, and 2, resulting in the following.*

$$|\Pi^{\mathbb{IN}}_{k1}| + |\Pi^{\neg\mathbb{IN}}_{k2}| = |\Pi_n| \tag{220}$$

$$|\Pi^{\neg\mathbb{IN}}_{k2}| < |\Pi_n| \tag{221}$$

$$\mathsf{conf}_{k1} = \langle [], \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, \emptyset, \emptyset\}, IC, \Pi^{\mathbb{IN}}_{k1}\rangle \tag{222}$$

$$SS_r \neq \emptyset \tag{223}$$

$$CC_r \neq \emptyset \tag{224}$$

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^{k1}_{\Pi^{\mathbb{IN}}_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathbb{IN}}_{k2}} \mathsf{conf}'_n \tag{225}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}}_{k1} :: \Pi^{\neg\mathbb{IN}}_{k2}\rangle \tag{226}$$

We indeed know that by applying $k1$ steps on the sub-trace $\Pi^{\mathbb{IN}}_{k1}$, we consume all the possible $\mathcal{I}$ of the initial WEBIOT*configuration* conf. *Then, because of the rule* INITSERVICE, *each time this logical relation is applied means that an init is consumed and a service and a booting client are added into the* WEBIOT*configuration. Moreover, we know that both $\Pi_n \neq \Pi^{\mathbb{IN}}_{k1} :: \Pi^{\neg\mathbb{IN}}_{k2}$ and $\Pi^{\mathbb{IN}}_{k1} \neq \Pi_{[0,k1]}$. But both traces keep preserved the observations that can be made on the final $\mathcal{L}$. We define this non-structural equivalence relation between traces as $\cong$.*

Formally, Given two traces Π and Π', and a well-formed WEBIOT*initial configuration*

$$\forall n, n', \Pi_n, \Pi'_{n'}, \mathsf{conf}, \mathsf{conf}_n, \mathsf{conf}_{n'}, \mathcal{L}.$$
$$\Pi_n \cong \Pi'_{n'}$$
$$\Leftrightarrow$$
$$\mathsf{conf} \rightsquigarrow^{n}_{\Pi_n} \mathsf{conf}_n\{\mathcal{L}\} \quad \wedge \quad \mathsf{conf} \rightsquigarrow^{n'}_{\Pi'_{n'}} \mathsf{conf}_n\{\mathcal{L}\}$$

, where n and n' might be different . We add this relation as an hypothesis.

$$\Pi_n \cong \Pi^{\mathbb{IN}}_{k1} :: \Pi^{\neg\mathbb{IN}}_{k2} \tag{227}$$

. The only case in which Π_n and $\Pi^{\mathbb{IN}}_{k1} :: \Pi^{\neg\mathbb{IN}}_{k2}$ are equal, is the one in which no permutation happened, as the non-deterministic execution of WEBIOT*has all the inits in front.*

Now after having consumed all the $\mathcal{I}$ we can move on the rest of the trace $\Pi^{\neg\mathbb{IN}}_{k2}$.

From now on we consider only cases deriving from the Hypothesis that for the initial WEBIOT*configuration* conf*(Hypothesis 2),*

$$\mathcal{I} \neq \emptyset \tag{228}$$

.

Step 1 *In this step, we consider the case where SS_r is not an empty set. So by Hypotheses 220(*$|\Pi^{\mathfrak{S}}| = 0$*), 221,*

225, *222*, *223*, *226*, *and* $|\mathcal{I}| > |\emptyset|$, *we can apply Lemma 2.*

$$|\Pi_{k2}^{\neg\mathbb{IN}}| = |\Pi_{k1'}^{\mathbb{SS}}| + |\Pi_{k2'}^{\neg\mathfrak{S}1_0}| \tag{229}$$

$$|\Pi_{k2'}^{\neg\mathfrak{S}1_0}| < |\Pi_{k2}^{\neg\mathbb{IN}}| < |\Pi_n| \tag{230}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}} \mathsf{conf}_{k1} \rightsquigarrow^{k1'}_{\Pi^{\mathbb{SS}}_{k1'}} \mathsf{conf}_{(k1+k1')} \rightsquigarrow^{k2'}_{\Pi^{\neg\mathfrak{S}1_0}_{k2'}} \mathsf{conf}'_n \tag{231}$$

$$\mathsf{conf}_{(k1+k1')} = \langle [], \emptyset, HM, \{\emptyset, SS_a, SS_g, SS_e\}, \{CC_r, \emptyset, \emptyset, \emptyset\}, IC, \Pi^{\mathbb{IN}} :: \Pi^{\mathbb{SS}}_{k1'} \rangle \tag{232}$$

$$(SS_a \neq \emptyset \vee SS_g \neq \emptyset \vee SS_e \neq \emptyset) \tag{233}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathbb{SS}}_{k1'} :: \Pi^{\neg\mathfrak{S}1_0}_{k2'} \rangle \tag{234}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathbb{SS}}_{k1'} :: \Pi^{\neg\mathfrak{S}1_0}_{k2'} \tag{235}$$

Notice, that in case SS_r *is empty and* $k2 > 0$, *the Lemma 2 cannot apply, and we pass directly to* ***step 2****. Let's call* $\Pi^{\mathbb{SS}}_{k1'}$ *as* $\Pi^{\mathfrak{S}1_0}$, *and let* $k1$ *now be* $|\Pi^{\mathbb{IN}}| + |\Pi^{\mathfrak{S}1_0}|$ *and* $k2$ *be* $k2'$. *The renaming help in not spreading to much in using sub or overscripts. We rewrite Hypothesis 238 as*

$$|\Pi^{\neg\mathbb{IN}}| = |\Pi^{\mathfrak{S}1_0}| + |\Pi^{\neg\mathfrak{S}1_0}| \tag{236}$$

$$|\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{237}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}1_0})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}1_0}_{k2}} \mathsf{conf}'_n \tag{238}$$

$$\mathsf{conf}_{k1} = \langle [], \emptyset, HM, \{\emptyset, SS_a, SS_g, SS_e\}, \{CC_r, \emptyset, \emptyset, \emptyset\}, IC, \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}1_0} \rangle \tag{239}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\neg\mathfrak{S}1_0}_{k2} \rangle \tag{240}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}1_0} :: \Pi^{\neg\mathfrak{S}1_0}_{k2} \tag{241}$$

.

Step 2 *Here we distinguish two cases, one in which* $SS_e = \emptyset$ *and the other in which it is not.*

$SS_e = \emptyset$ *Lemma 3 cannot be applied, then go to step 3.*

$SS_e \neq \emptyset$ *Similarly to* `Step 1`, *we can apply Lemma 3 to the Hypotheses, to bunch toghether all the possible* $\mathbb{RS}$.

$$|\Pi_{k2}^{\neg\mathfrak{S}1_0}| = |\Pi_{k1'}^{\mathbb{RS}}| + |\Pi_{k2'}^{\neg\mathfrak{S}2_0}| \tag{242}$$

$$|\Pi_{k2'}^{\neg\mathfrak{S}2_0}| < |\Pi_{k2}^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{243}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}1_0})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k1'}_{\Pi^{\mathbb{RS}}_{k1'}} \mathsf{conf}_{(k1+k1')} \rightsquigarrow^{k2'}_{\Pi^{\neg\mathfrak{S}2_0}_{k2'}} \mathsf{conf}'_n \tag{244}$$

$$\mathsf{conf}_{(k1+k1')} = \langle \mathcal{L}', \emptyset, HM, \{\emptyset, SS_a, SS_g, \emptyset\}, \{CC'_r, \emptyset, \emptyset, \emptyset\}, IC'', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}1_0} :: \Pi^{\mathbb{RS}}_{k1'} \rangle \tag{245}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}1_0} :: \Pi^{\mathbb{RS}}_{k1'} :: \Pi^{\neg\mathfrak{S}2_0}_{k2'} \rangle \tag{246}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}1_0} :: \Pi^{\mathbb{RS}}_{k1'} :: \Pi^{\neg\mathfrak{S}2_0}_{k2'} \tag{247}$$

Now, as we previously did, we merge the traces $\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}1_0}$ *with* $\Pi^{\mathbb{RS}}$ *letting* $k1$ *be* $k1 + k1'$ *and* $k2$ *be* $k2'$. *We do rename* $\Pi^{\mathfrak{S}1_0} :: \Pi^{\mathbb{RS}}$ *as* $\Pi^{\mathfrak{S}2_0}$.

$$|\Pi^{\neg\mathfrak{S}1_0}| = |\Pi^{\mathbb{RS}}| + |\Pi^{\neg\mathfrak{S}2_0}| \tag{248}$$

$$|\Pi^{\neg\mathfrak{S}2_0}| < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{249}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}2_0})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}2_0}_{k2}} \mathsf{conf}'_n \tag{250}$$

$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{\emptyset, SS_a, SS_g, \emptyset\}, \{CC'_r, \emptyset, \emptyset, \emptyset\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}2_0})_{k1} \rangle \tag{251}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}2_0} :: \Pi^{\neg\mathfrak{S}2_0}_{k2} \rangle \tag{252}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}2_0} :: \Pi^{\neg\mathfrak{S}2_0} \tag{253}$$

Now we can proceed to the next step.

Step 3 *The third step of the trace transformation regards device rules. We do not know if there are actuations or readings on devices to be performed. It might be that both the partitions SS_a and SS_g are empty in the configuration conf_{k1}. In this case, we still cannot exclude that there are some $\mathbb{DS}$ in $\Pi^{\neg\mathfrak{S}2_0}$ that can be reordered in the trace. In both cases, we know that, if Lemma 4 applies on the Hypotheses 248 to 258, then we derive the following.*

$$|\Pi^{\neg\mathfrak{S}2_0}_{k2}| = |\Pi^{(\mathbb{DV}+\mathbb{DS})}_{k1'}| + |\Pi^{\neg\mathfrak{S}3_0}_{k2'}| \tag{254}$$

$$|\Pi^{\neg\mathfrak{S}3_0}| \leq |\Pi^{\neg\mathfrak{S}2_0}| < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{255}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}2_0})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k1'}_{\Pi^{(\mathbb{DV}+\mathbb{DS})}_{k1'}} \mathsf{conf}_{(k1+k1')} \rightsquigarrow^{k2'}_{\Pi^{\neg\mathfrak{S}3_0}_{k2'}} \mathsf{conf}'_n \tag{256}$$

$$\mathsf{conf}_{(k1+k1')} = \langle \mathcal{L}'', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC''', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}2_0} :: \Pi^{\mathbb{DV}}_{k1'} \rangle \tag{257}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}2_0} :: \Pi^{(\mathbb{DV}+\mathbb{DS})}_{k1'} :: \Pi^{\neg\mathfrak{S}3}_{k2'} \rangle \tag{258}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}2_0} :: \Pi^{(\mathbb{DV}+\mathbb{DS})}_{k1'} :: \Pi^{\neg\mathfrak{S}3_0}_{k2'} \tag{259}$$

Notice that hypothesis 261 do not state forcely a trace size-reduction, as it might happen that the Lemma 4 applies indipendently of what SS_a and SS_g are. Actually if both of the partitions are empty, we can still reorder the trace looking for $\mathbb{DS}$s that happen in $\Pi^{\neg\mathfrak{S}2_0}$ and are placed before the first device actuation or reading in the trace.

As we previously did, we can rename $k1$ to be $k1 + k1'$ and $k2$ to be $k2'$. Morover, now $\Pi^{\mathfrak{S}3_0} = \Pi^{\mathfrak{S}2_0} :: \Pi^{(\mathbb{DV}+\mathbb{DS})}$. We rewrite the previous hypotheses as:

$$|\Pi^{\neg\mathfrak{S}2_0}| = |\Pi^{(\mathbb{DV}+\mathbb{DS})}| + |\Pi^{\neg\mathfrak{S}3_0}| \tag{260}$$

$$|\Pi^{\neg\mathfrak{S}3_0}| \leq |\Pi^{\neg\mathfrak{S}2_0}| < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{261}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}3_0})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}3_0}_{k2}} \mathsf{conf}'_n \tag{262}$$

$$\mathsf{conf}_{k1} = \langle \mathcal{L}'', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC''', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}3_0} \rangle \tag{263}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}3_0} :: \Pi^{\neg\mathfrak{S}3}_{k2} \rangle \tag{264}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}3_0} :: \Pi^{\neg\mathfrak{S}3_0} \tag{265}$$

We can comment the influence that the rule $\mathbb{DS}$ can have on the configuration. The rule, changes the log $\mathcal{L}$, but does it influence its meaning? Indeed the answer is no. In the model there is not a notion of time,so, the occurance of a device sensing the world state via the world oracle wo can happen whenever. If a sensing happens before an actuation, then reordering it backwards would not change its meaning. Obviously, if a sensing happens after an actuation, swapping the two would result in a different reading of the world state.

Step 4 *Now we look for client steps in the non ordered trace $\Pi^{\neg\mathfrak{S}3_0}$. Since we are in the first iteration of the reordering we know that if at least an init in the initial conf, then there is at least an active client in conf_{k1} (hypothesis 276). Because of hypothesis 224, 260, 261, 262, 276, 264, we can apply Lemma 5.*

$$|\Pi^{\neg\mathfrak{S}3_0}_{k2}| = |\Pi^{\mathbb{CS}}_{k1'}| + |\Pi^{\neg\mathfrak{S}4_0}_{k2'}| \tag{266}$$

$$|\Pi^{\neg\mathfrak{S}4_0}_{k2'}| < |\Pi^{\neg\mathfrak{S}3_0}_{k2}| \tag{267}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}3_0})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k1'}_{\Pi^{\mathbb{CS}}_{k1'}} \mathsf{conf}_{(k1+k1')} \rightsquigarrow^{k2'}_{\Pi^{\neg\mathfrak{S}4_0}_{k2'}} \mathsf{conf}'_n \tag{268}$$

$$\mathsf{conf}_{(k1+k1')} = \langle \mathcal{L}', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{\emptyset, CC_c, CC_t, CC''_e\}, IC'', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}3_0} :: \Pi^{\mathbb{CS}}_{k1'} \rangle \tag{269}$$

$$(|CC'_e| \leq |CC''_e| \leq |\mathcal{I}|) \tag{270}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}3_0} :: \Pi^{\mathbb{CS}}_{k1'} :: \Pi^{\neg\mathfrak{S}4_0}_{k2'} \rangle \tag{271}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}3_0} :: \Pi^{\mathbb{CS}}_{k1'} :: \Pi^{\neg\mathfrak{S}4_0}_{k2'} \tag{272}$$

. *We rename* $k1$ *to be* $k1 + k1'$ *and* $k2$ *to be* $k2'$*. Morover, now* $\Pi^{\mathfrak{S}4_0} = \Pi^{\mathfrak{S}3_0} :: \Pi^{(\mathbb{CS})}$*. We rewrite the previous hypotheses as:*

$$|\Pi^{\neg\mathfrak{S}3_0}| = |\Pi^{\mathbb{CS}}| + |\Pi^{\neg\mathfrak{S}4_0}| \tag{273}$$

$$|\Pi^{\neg\mathfrak{S}4_0}| < |\Pi^{\neg\mathfrak{S}3_0}| \leq |\Pi^{\neg\mathfrak{S}2_0}| < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{274}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}4_0})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}4_0}_{k2}} \mathsf{conf}'_n \tag{275}$$

$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{\emptyset, CC_c, CC_t, CC''_e\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}4_0})_{k1}\rangle \tag{276}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}4_0} :: \Pi^{\neg\mathfrak{S}4_0}_{k2}\rangle \tag{277}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\neg\mathfrak{S}4_0} \tag{278}$$

. *Let's consider Hypothesis 270.*

$|CC''_e| = |\mathcal{I}|$ *In this case we can apply the termination lemma, Lemma 8. Because of the lemma, we know that* $k1 = n$*,* $\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}}$*, and* $\Pi^{\neg\mathfrak{S}4} = []$*. Let's explain a bit the relation between the cardinality of* $\mathcal{I}$ *and the one of* CC_e*.*
The number of inits represents the number of clients that will be active in WEBIOT*. While they are active, their code steps, calls, runs their thunks etc. At a certain point, each client terminates all its tasks and finishes. In this case, the clients stop being an active member of* WEBIOT*execution. Once the cardinality of these two sets is equal, it means that all the clients have finished to compute. It follows that having no clients active, then no service can be running or waiting to return a result to a client. This means that we have permuted all the trace. obtaining one that is equivalent to* Π_n

$|CC'_e| \leq |CC''_e| < |\mathcal{I}|$ *go to* ***step 5****.*

step 5 *In this step we, if any, reorder clients calls existing in* $\Pi^{\neg\mathfrak{S}4_0}$*.*

$CC_c = \emptyset$ *Go to* ***step 5****. In the first iteration, this case is not possible. Indeed, if there is no call made by at least a client it means that the clients finished their computation and returned a result, with empty callbacks and thunks.*

$CC_c \neq \emptyset$ *In this case we can apply Lemma 6 on Hypotheses 273 to 277.*

$$|\Pi^{\neg\mathfrak{S}4_0}_{k2}| = |\Pi^{\mathbb{CC}}_{k1'}| + |\Pi^{\neg\mathfrak{S}5_0}_{k2'}| \tag{279}$$

$$|\Pi^{\neg\mathfrak{S}5_0}_{k2'}| < |\Pi^{\neg\mathfrak{S}4_0}_{k2}| < |\Pi^{\neg\mathfrak{S}3_0}| \leq |\Pi^{\neg\mathfrak{S}2_0}| < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{280}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}4_0})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k1'}_{\Pi^{\mathbb{CC}}_{k1'}} \mathsf{conf}_{(k1+k1')} \rightsquigarrow^{k2'}_{\Pi^{\neg\mathfrak{S}5_0}_{k2'}} \mathsf{conf}'_n \tag{281}$$

$$\mathsf{conf}_{(k1+k1')} = \langle \mathcal{L}', \emptyset, HM, \{SS'_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, CC'_t, CC''_e\}, IC'', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}4_0} :: \Pi^{\mathbb{CC}}_{k1'}\rangle \tag{282}$$

$$(|CC'_e| \leq |CC''_e| < |\mathcal{I}|) \tag{283}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}4_0} :: \Pi^{\mathbb{CC}}_{k1'} :: \Pi^{\neg\mathfrak{S}5_0}_{k2'}\rangle \tag{284}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}4_0} :: \Pi^{\mathbb{CC}}_{k1'} :: \Pi^{\neg\mathfrak{S}5_0}_{k2'} \tag{285}$$

We rename $k1$ *to be* $k1 + k1'$ *and* $k2$ *to be* $k2'$*. Morover, now* $\Pi^{\mathfrak{S}5_0} = \Pi^{\mathfrak{S}4_0} :: \Pi^{(\mathbb{CC})}$*. We rewrite the previous hypotheses as:*

$$|\Pi^{\neg\mathfrak{S}4_0}| = |\Pi^{\mathbb{CC}}| + |\Pi^{\neg\mathfrak{S}5_0}| \tag{286}$$

$$|\Pi^{\neg\mathfrak{S}5_0}| < |\Pi^{\neg\mathfrak{S}4_0}| < |\Pi^{\neg\mathfrak{S}3_0}| \leq |\Pi^{\neg\mathfrak{S}2_0}| < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{287}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}5_0})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}5_0}_{k2}} \mathsf{conf}'_n \tag{288}$$

$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{SS'_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, CC'_t, CC''_e\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}5_0})_{k1}\rangle \tag{289}$$

$$(|CC'_e| \leq |CC''_e| < |\mathcal{I}|) \tag{290}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}5_0} :: \Pi^{\neg\mathfrak{S}5_0}_{k2}\rangle \tag{291}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}5_0} :: \Pi^{\neg\mathfrak{S}5_0}_{k2} \tag{292}$$

step 6 *If we consider* conf_{k1} *in this first iteration, we know that after the initialization, all the clients have stepped and some might have called services, storing callbacks in their set* B*. But till now, there a no services called in the* ***step 5*** *that have stepped or finished their computation. This means, for each client it does not exists a callback of that has been made thunk. Hence, the* RUN *rule cannot apply. This means that we cannot apply the Lemma 7 to reorder* $\mathbb{RN}$*s to evaluate at this point of the trace. Hence* $CC_t = \emptyset$*.*

The first iteration now is completed.

Further iterations *Let's consider the hypothesis of generic i-th iteration after* ***step 6****. We can rewrite the Hypotheses 286 to 291 as the following.*

$$|\Pi^{\neg\mathfrak{S}5_{(i-1)}}| = |\Pi^{\mathbb{RN}_{(i-1)}}| + |\Pi^{\neg\mathfrak{S}6_{(i-1)}}| \tag{293}$$

$$|\Pi^{\neg\mathfrak{S}6_{(i-1)}}| \leq |\Pi^{\neg\mathfrak{S}5_{(i-1)}}| \leq |\Pi^{\neg\mathfrak{S}4_{(i-1)}}| < \cdots < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{294}$$

$$\Pi^{\mathfrak{S}6_{(i-1)}} = \Pi^{\mathfrak{S}5_{(i-1)}} :: \Pi^{\mathbb{RN}_{(i-1)}} \tag{295}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}6_{(i-1)}})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}6_{(i-1)}}_{k2}} \mathsf{conf}'_n \tag{296}$$

$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}6_{(i-1)}})_{k1} \rangle \tag{297}$$

$$|CC'_e| < |\mathcal{I}| \tag{298}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\neg\mathfrak{S}6_{(i-1)}}_{k2} \rangle \tag{299}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}6_{(i-1)}} :: \Pi^{\neg\mathfrak{S}6_{(i-1)}}_{k2} \tag{300}$$

The trace $\Pi^{\mathbb{RN}}$ *is a piece of the trace containing all the runs that could be reordered on the trace* $|\Pi^{\neg\mathfrak{S}5_{(i-1)}}|$*, in the* ***step 6*** *of the previous iteration* $(i-1)$*. The* $\Pi^{\neg\mathfrak{S}6_{(i-1)}}$ *is the part of the trace that still has to be reordered after* ***step 6****, and* $\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}6_{(i-1)}}$ *is the reordered trace, which includes also* $\Pi^{\mathbb{RN}}$*. Notice, that* $\Pi^{\mathbb{RN}}$ *can be also the empty trace. In that case the meaning is that there has been no run available to run in that iteration, hence no possible reordering on the trace* $\Pi^{\neg\mathfrak{S}}$*.*

step 1 *This step is similar to the one defined in the first iteration. In this case we should consider either* SS_r *is empty or not.*

- $SS_r = \emptyset$ *Considering Hypothesis 297, there are two cases regarding the partition set* CC_r*.*
  - $CC_r = \emptyset$ *Then* $|CC'_e|$ *has to be equal to* $|\mathcal{I}|$*. This means that the termitation Lemma 8 applied in* ***step 4*** *of the previous iteration*$(i-1)$*. This case is not possible as no reordering is possible, because* $|\Pi^{\neg\mathfrak{S}6_{(i-1)}}| = 0$*. This means that the trace has been already all succesfully permuted.*
  - $CC_r \neq \emptyset$ *go to* ***step 3*** *as the partition subsets* SS_e*,* SS_a*, and* SS_g *are empty. We go to this step as it could be possible to reorder some* $\mathbb{DS}$ *in the trace* $\Pi^{\neg\mathfrak{S}6_{(i-1)}}$*, with the guarantee that no returns, acts or readings can be performed by any service, as there are none active in* conf_{k1}*.*
    This case is possible if all the services have returned in the previous iteration, and no client called some new service. The Clients partition CC_r *is not empty because there was still at least a client that run a thunk in the previous iteration.*
- $SS_r \neq \emptyset$ *We can apply Lemma 2, similarly to the* ***step 1*** *of the first iteration, on the Hypotheses 293 to 299. Notice that in this case, the current iteration cannot be the last one, as each running service is related to some client's callback. Then it is possible, as did in the first iteration, to first do the* ***step 2****(with* $SS_e \neq \emptyset$*) of the trace reordering, and then* ***step 3****.*
  We show below the derivations of ***step 1*** *and* ***step 2****, considering as initial hypotheses the ones from 293 to 299*

*For **step 1** we apply Lemma 2 on the hypotheses, obtaining the following.*

$$|\Pi^{\neg\mathfrak{S}6_{(i-1)}}| = |\Pi^{\mathbb{SS}_i}| + |\Pi^{\neg\mathfrak{S}1_i}| \tag{301}$$

$$|\Pi^{\neg\mathfrak{S}1_i}| < |\Pi^{\neg\mathfrak{S}6_{(i-1)}}| \leq |\Pi^{\neg\mathfrak{S}5_{(i-1)}}| \leq |\Pi^{\neg\mathfrak{S}4_{(i-1)}}| < \cdots < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{302}$$

$$\Pi^{\mathfrak{S}1_i} = \Pi^{\mathfrak{S}6_{(i-1)}} :: \Pi^{\mathbb{SS}_i} \tag{303}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}1_i})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}1_i}_{k2}} \mathsf{conf}'_n \tag{304}$$

$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{\emptyset, SS_a, SS_g, SS_e\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}1_i})_{k1} \rangle \tag{305}$$

$$|CC'_e| < |\mathcal{I}| \tag{306}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}1_i} :: \Pi^{\neg\mathfrak{S}1_i}_{k2} \rangle \tag{307}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}1_i} :: \Pi^{\neg\mathfrak{S}1_i} \tag{308}$$

Then, considering Hypothesis 305 representing the WEBIOTconfiguration conf_{k1}, if $SS_e = \emptyset$ then there are no service returns to reorder in the trace $\Pi^{\neg\mathfrak{S}1_i}$. Otherwise, we apply Lemma 3.

$$|\Pi^{\neg\mathfrak{S}1_i}| = |\Pi^{\mathbb{RS}_i}| + |\Pi^{\neg\mathfrak{S}2_i}| \tag{309}$$

$$|\Pi^{\neg\mathfrak{S}2_i}| < |\Pi^{\neg\mathfrak{S}1_i}| < |\Pi^{\neg\mathfrak{S}6_{(i-1)}}| \leq |\Pi^{\neg\mathfrak{S}5_{(i-1)}}| \leq |\Pi^{\neg\mathfrak{S}4_{(i-1)}}| < \cdots < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{310}$$

$$\Pi^{\mathfrak{S}2_i} = \Pi^{\mathfrak{S}1_i} :: \Pi^{\mathbb{RS}_i} \tag{311}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}2_i})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}2_i}_{k2}} \mathsf{conf}'_n \tag{312}$$

$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{\emptyset, SS_a, SS_g, \emptyset\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}2_i})_{k1} \rangle \tag{313}$$

$$|CC'_e| < |\mathcal{I}| \tag{314}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}2_i} :: \Pi^{\neg\mathfrak{S}2_i}_{k2} \rangle \tag{315}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}2_i} :: \Pi^{\neg\mathfrak{S}2_i} \tag{316}$$

Again, we see that after applying the lemma, conf_{k1} is an intermediate configuration resulting from the configuration defined by hypothesis 305, after $|\Pi^{\mathbb{RS}_i}|$ transition steps over the reordered trace $\Pi^{\mathbb{RS}_i}$. The resulting configuration has no more service returns available to be consumed.

step 3 *We consider the configuration conf_{k1} to be the one with both the SS_r and SS_e partitions empty.*

$$\mathsf{conf}_{k1} = \langle \mathcal{L}', \emptyset, HM, \{\emptyset, SS_a, SS_g, \emptyset\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC'', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}i})_{k1} \rangle$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} :: \Pi^{\neg\mathfrak{S}}_{k2} \rangle$$

We do not annotate $\Pi^{\mathfrak{S}i}$ with the precise step number as we do not know precisely where this configuration comes from. We previously showed that it cuould happen that the partition SS_r can be empty at the beginning of an iteration. Indeed, the only thing we do know about the subtrace $\Pi^{\mathfrak{S}i}$, is that it is an ordered trace, and it is built up by the first reordering iteration followed by (i-1) other iterations. We can apply, as we previously did in the first iteration, Lemma 4, indipendently from whether SS_a and SS_g are empty or not.

$$|\Pi^{\neg\mathfrak{S}i}| = |\Pi^{(\mathbb{DV}+\mathbb{DS})_i}| + |\Pi^{\neg\mathfrak{S}3_i}| \tag{317}$$

$$|\Pi^{\neg\mathfrak{S}3_i}| \leq |\Pi^{\neg\mathfrak{S}i}| \leq |\Pi^{\neg\mathfrak{S}5_{(i-1)}}| \leq |\Pi^{\neg\mathfrak{S}4_{(i-1)}}| < \cdots < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{318}$$

$$\Pi^{\mathfrak{S}3_i} = \Pi^{\mathfrak{S}i} :: \Pi^{(\mathbb{DV}+\mathbb{DS})_i} \tag{319}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}3_i})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}3_i}_{k2}} \mathsf{conf}'_n \tag{320}$$

$$\mathsf{conf}_{k1} = \langle \mathcal{L}'', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, \emptyset, CC'_e\}, IC''', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}3_i})_{k1} \rangle \tag{321}$$

$$|CC'_e| < |\mathcal{I}| \tag{322}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}3_i} :: \Pi^{\neg\mathfrak{S}3_i}_{k2} \rangle \tag{323}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}3_i} :: \Pi^{\neg\mathfrak{S}3_i} \tag{324}$$

Notice that if $|\Pi^{\neg\mathfrak{S}3_i}| = |\Pi^{\neg\mathfrak{S}i}|$ *(Hypothesis 326) it means that* SS_g *and* SS_r *are empty and no eligible* $\mathbb{DS}$ *has been found in the trace for this iteration. Then,* $\Pi^{\mathfrak{S}3_i}$ *is an empty trace. Then we go to the* ***step 4****.*

step 4 *We know for the Lemma 8 that if we started this iteration it means that at least a client is still active in the* WEBIOT *configuration* conf_{k1} *represented in Hypothesis 321. We can apply Lemma 5.*

$$|\Pi^{\neg\mathfrak{S}3_i}| = |\Pi^{(\mathbb{CS})_i}| + |\Pi^{\neg\mathfrak{S}4_i}| \tag{325}$$

$$|\Pi^{\neg\mathfrak{S}4_i}| < |\Pi^{\neg\mathfrak{S}3_i}| \leq \cdots \leq |\Pi^{\neg\mathfrak{S}5_{(i-1)}}| \leq |\Pi^{\neg\mathfrak{S}4_{(i-1)}}| < \cdots < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{326}$$

$$\Pi^{\mathfrak{S}4_i} = \Pi^{\mathfrak{S}3_i} :: \Pi^{(\mathbb{CS})_i} \tag{327}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}3_i})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}3_i}_{k2}} \mathsf{conf}'_n \tag{328}$$

$$\mathsf{conf}_{k1} = \langle \mathcal{L}'', \emptyset, HM, \{SS_r, \emptyset, \emptyset, \emptyset\}, \{\emptyset, CC_c, CC_t, CC''_e\}, IC''', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}4_i})_{k1} \rangle \tag{329}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}4_i} :: \Pi^{\neg\mathfrak{S}4_i}_{k2} \rangle \tag{330}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}4_i} :: \Pi^{\neg\mathfrak{S}4_i} \tag{331}$$

Now we can argue about CC''_e*. If its cardinality is the same as* $|\mathcal{I}|$*, then we can apply the termination lemma, Lemma 8 and state that* $\Pi^{\neg\mathfrak{S}4_i} = []$*,* $\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}4_i} \cong \Pi_n$*,* $\mathcal{L}'' = \mathcal{L}$*, and* $IC''' = IC'$ *. If not, we can go to* ***step 5****, as we did not finish yet to commute the trace, meaning that* $\Pi^{\neg\mathfrak{S}4_i} \neq []$*.*

Notice that a reordered trace can only finish with a client step as last element of the trace. Indeed, the latter ends evaluating the code of the last active client in WEBIOT*. On the other hand, the unordered trace* Π_n *can have also a* $\mathbb{DS}$ *as final element of the trace.*

step 5 *This step is similar to the one described in the first iteration. If* CC_c *is not empty then all the eligible calls are reordered in the beginning of the trace. If exists at least a client call then we know that the next iteration will not be the last one. We show directly the* ***step 6****. If it* CC_c *is empty, then* CC_t *must not. If not* $|CC_e| = |\mathcal{I}|$*, meaning that the termination lemma should have applied in the* ***step 4*** *of this iteration. For the latter case:*

$$|\Pi^{\neg\mathfrak{S}4_i}| = |\Pi^{(\mathbb{CC})_i}| + |\Pi^{\neg\mathfrak{S}5_i}| \tag{332}$$

$$|\Pi^{\neg\mathfrak{S}5_i}| < |\Pi^{\neg\mathfrak{S}4_i}| < \cdots \leq |\Pi^{\neg\mathfrak{S}5_{(i-1)}}| \leq |\Pi^{\neg\mathfrak{S}4_{(i-1)}}| < \cdots < |\Pi^{\neg\mathfrak{S}1_0}| < |\Pi^{\neg\mathbb{IN}}| < |\Pi_n| \tag{333}$$

$$\Pi^{\mathfrak{S}5_i} = \Pi^{\mathfrak{S}4_i} :: \Pi^{(\mathbb{CC})_i} \tag{334}$$

$$\mathsf{conf} \rightsquigarrow^{k1}_{(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}3_i})_{k1}} \mathsf{conf}_{k1} \rightsquigarrow^{k2}_{\Pi^{\neg\mathfrak{S}3_i}_{k2}} \mathsf{conf}'_n \tag{335}$$

$$\mathsf{conf}_{k1} = \langle \mathcal{L}'', \emptyset, HM, \{SS'_r, \emptyset, \emptyset, \emptyset\}, \{CC_r, \emptyset, CC_t, CC''_e\}, IC''', (\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}5_i})_{k1} \rangle \tag{336}$$

$$\mathsf{conf}'_n = \langle \mathcal{L}, \emptyset, HM, \{\emptyset, \emptyset, \emptyset, \emptyset\}, \{\emptyset, \emptyset, \emptyset, CC_e\}, IC', \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}5_i} :: \Pi^{\neg\mathfrak{S}5_i}_{k2} \rangle \tag{337}$$

$$\Pi_n \cong \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}5_i} :: \Pi^{\neg\mathfrak{S}5_i} \tag{338}$$

step 6 *We analyse two cases.*

- $CC_t = \emptyset$ *This iteration has ended. Go to* ***step 1***
- $CC_t \neq \emptyset$ *We can apply Lemma 7. If no client call has happened the we consider the Hypotheses 325 to 330, otherwise the ones from 332 to 337. after applying the lemma, we start with a new iteration, starting from* ***step 1****.*

Termination *If we consider the second hypothesis of each iteration, knowing that the trace is finite, we see that the size of the unordered trace decreases each time a commutation step is applies. This means that sooner or later the termination lemma will apply at some i-th iteration at* ***step 4****. Once no more clients are active in* WEBIOT*, the trace is ordered.*

So now we consider an ordered trace equivalent to Π_n*, called* $\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}}$*. Knowing that they are equivalent, we show that the latter trace can be simulated by the scheduler. We rewrite* $\Leftarrow$*.*

$$\forall\ \mathit{wo}, \mathcal{E}, n, \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}}, \mathsf{conf}_n, \mathsf{conf}'_n.\ \exists\ t :$$
$$\mathit{wo}, \mathcal{E} \vdash \mathsf{conf}\{[]\} \rightsquigarrow^{n} \mathsf{conf}_n\{\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}}\}$$
$$\Rightarrow$$
$$(\mathit{wo}, \mathcal{E} \vdash [\![\mathsf{conf}]\!]_{\mathbb{IN}}^{\pi_{\mathfrak{L}}(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}}),\pi_{\mathfrak{T}}(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}})} \rightsquigarrow_{\mathfrak{S}}^{(n+t)} [\![\mathsf{conf}'_{(n)}]\!]_{\mathtt{end}}^{[],[]}) \ \wedge\ (\mathsf{conf}'_n \ =\ \mathsf{conf}_n)$$

We will prove the theorem by first by case analysis on n. Let's call for simplicity $\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}}$ as $\Pi'^{\mathfrak{S}}$, where $|\Pi'^{\mathfrak{S}}| = n$.

$n = 0$ *trivial.*

$n \neq 0$ *We define an induction predicate stating that the scheduler can always imitate the reordered trace.*

$$\forall n, k, \Pi'^{\mathfrak{S}}, \mathsf{conf}, \mathsf{conf}_k.$$
$$\mathsf{conf} \rightsquigarrow^{k}_{\Pi'^{\mathfrak{S}}_{[0,k]}} \mathsf{conf}_k$$
$$\Rightarrow$$
$$\exists t, \mathbb{X}. \tag{IH1}$$
$$[\![\mathsf{conf}]\!]_{\mathbb{IN}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}})} \rightsquigarrow_{\mathfrak{S}}^{(k+t)} [\![\mathsf{conf}'_k]\!]_{\mathbb{X}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1,n]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1,n]})} \qquad \wedge$$
$$\mathsf{conf}'_k = \mathsf{conf}_k$$

, where $\Pi'^{\mathfrak{S}}_{[i,j]}$ is a subtrace of $\Pi'^{\mathfrak{S}}$, from the i-th element to the j-th. If we write $\Pi'^{\mathfrak{S}}_{[i]}$ then it means we are considering the i-th element of the trace.

We assume that this predicate is true for k steps. We show that it is true also for $k+1$ steps. we go by case analysis on the k and $k+1$ transitions.

case k: Following this assumptions, we know the following hypotheses, derived from the induction predicate, hold. We remark that $\Pi'^{\mathfrak{S}}$ is a reordered trace equivalent to Π_n.

$$\Pi'^{\mathfrak{S}} \cong \Pi_n \tag{IH1}$$
$$\mathsf{conf} \rightsquigarrow^{k}_{\Pi'^{\mathfrak{S}}_{[0,k]}} \mathsf{conf}_k \tag{IH2}$$
$$[\![\mathsf{conf}]\!]_{\mathbb{IN}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}})} \rightsquigarrow_{\mathfrak{S}}^{(k+t)} [\![\mathsf{conf}'_k]\!]_{\mathbb{X}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1,n]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1,n]})} \tag{IH3}$$
$$\mathsf{conf}'_k = \mathsf{conf}_k \tag{IH4}$$

case (k+1): We rewrite the induction hypothesis for the case k + 1.

$$\forall n, k, \Pi'^{\mathfrak{S}}, \mathsf{conf}_k, \mathsf{conf}_{(k+1)}.$$
$$\mathsf{conf} \rightsquigarrow^{k}_{\Pi'^{\mathfrak{S}}_{[0,k]}} \mathsf{conf}_k \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[k+1]}} \mathsf{conf}_{(k+1)}$$
$$\Rightarrow$$
$$\exists t, t', \mathbb{X}, \mathbb{X}'.$$
$$[\![\mathsf{conf}]\!]_{\mathbb{IN}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}})} \rightsquigarrow_{\mathfrak{S}}^{(k+t)} [\![\mathsf{conf}'_k]\!]_{\mathbb{X}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1,n]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1,n]})} \rightsquigarrow_{\mathfrak{S}}^{(1+t')} [\![\mathsf{conf}'_{(k+1)}]\!]_{\mathbb{X}'}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+2,n]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+2,n]})}$$
$$\wedge \qquad \mathsf{conf}'_{(k+1)} = \mathsf{conf}_{(k+1)}$$

(IH5)

We proceed by case analysis on $\Pi'^{\mathfrak{S}}_{[k+1]}$.

Let's consider this hypothesis.

$$\mathsf{conf} \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[0]}} \dots \mathsf{conf}_{(k-1)} \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[k]}} \mathsf{conf}_k \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[k+1]}} \mathsf{conf}_{(k+1)} \tag{IH6}$$

We proceed by case analysis on first $\Pi'^{\mathfrak{S}}_{[k]}$*, and then* $\Pi'^{\mathfrak{S}}_{[k+1]}$*.*

$\Pi'^{\mathfrak{S}}_{[k]} = \mathbb{IN}$

$\Pi'^{\mathfrak{S}}_{[k+1]} = \mathbb{IN}$ *We rewrite the hypothesis below.*

$$\mathsf{conf} \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[0]}} \dots \mathsf{conf}_{(k-1)} \rightsquigarrow_{\mathbb{IN}} \mathsf{conf}_k \rightsquigarrow_{\mathbb{IN}} \mathsf{conf}_{(k+1)} \tag{IH7}$$

If we assume that after k transitions, both the semantics that followed $\Pi'^{\mathfrak{S}}_{[0,k]}$ *and the scheduler semantics led to the same configuration* conf_k*, then we know that the scheduler has still some* $\mathcal{I}$ *to consume, as* $\mathcal{I} \neq \emptyset$*. Then by applying the scheduler rule* INIT *on* conf_k*, we know that the transition in its premises can make the same choice as the one made by* WEBIOT*, as it is a transition. So, it holds the following.*

$$[\![\mathsf{conf}_{(k-1)}]\!]_{\mathbb{IN}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k]})} \rightsquigarrow^{\text{INIT}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]_{\mathbb{IN}}^{[],[]}$$
$$[\![\mathsf{conf}_k]\!]_{\mathbb{IN}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})} \rightsquigarrow^{\text{INIT}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]_{\mathbb{IN}}^{[],[]}$$
$$\mathsf{conf}_{(k+1)} = \mathsf{conf}'_{(k+1)}$$

$\Pi'^{\mathfrak{S}}_{[k+1]} = ((j,i), \mathbb{SS})$ *We rewrite the Hypothesis IH6 below.*

$$\mathsf{conf} \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[0]}} \dots \mathsf{conf}_{(k-1)} \rightsquigarrow_{\mathbb{IN}} \mathsf{conf}_k \rightsquigarrow_{\mathbb{SS}_{(j,i)}} \mathsf{conf}_{(k+1)} \tag{IH8}$$

Because of the reordering of the trace, we know that in case the $k+1$ *transition is a service step and the k-th transition is a* $\mathbb{IN}$*, then in* conf_k $\mathcal{I} = \emptyset$*, meaning that all the inits has been processed. In this context we know that after* $|\mathbb{IN}|$ *steps, also the scheduler consumed all the inits. We show that by applying the rules* INITDONE*,* SERVERSTEPSOK*, and one of* LOOPSERVERSTEP$_{(j,i)}$ *or* SERVERSTEP$_{(j,i)}$ *the scheduler result in the same configuration* $\mathsf{conf}_{(k+1)}$*.*

$$[\![\mathsf{conf}_{(k-1)}]\!]_{\mathbb{IN}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k]})} \rightsquigarrow^{\text{INIT}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]_{\mathbb{IN}}^{[],[]} \tag{IH9}$$

$$\begin{aligned}[\![\mathsf{conf}_k]\!]_{\mathbb{IN}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})} &\rightsquigarrow^{\text{INITDONE}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]_{\mathbb{SS}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})} \\ \rightsquigarrow^{\text{SERVERSTEPSOK}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]_{\mathbb{SS}_{(j,i)}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})} &\rightsquigarrow^{\text{LOOPSERVERSTEP}_{(j,i)}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]_{\mathbb{SS}_{(j,i)}}^{[],[]}\end{aligned} \tag{IH10}$$

$$\begin{aligned}[\![\mathsf{conf}_k]\!]_{\mathbb{IN}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})} &\rightsquigarrow^{\text{INITDONE}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]_{\mathbb{SS}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})} \\ \rightsquigarrow^{\text{SERVERSTEPSOK}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]_{\mathbb{SS}_{(j,i)}}^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})} &\rightsquigarrow^{\text{SERVERSTEP}_{(j,i)}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]_{\mathbb{SS}}^{[],[]}\end{aligned} \tag{IH11}$$

$$\mathsf{conf}_{(k+1)} = \mathsf{conf}'_{(k+1)} \tag{IH12}$$

Notice that the rules INITDONE *and* SERVERSTEPSOK *are* ϵ*-transitions. Then* $\mathsf{conf}_{(k+1)}$ *is reached by the scheduler semantics after* $(k+1)+(t1+2)$ *transitions.*

$\Pi'^{\mathfrak{S}}_{[k]} = ((j,i), \mathbb{SS})$ *in this case, the k-th element of the trace is a server step on the service signed by (j,i).*

$\Pi'^{\mathfrak{S}}_{[k+1]} = ((j,i), \mathbb{SS})$ *If the (k + 1) element of the trace is another transition of the same service (j,i). Let's consider this Hypothesis.*

$$\mathsf{conf} \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[0]}} \ldots \mathsf{conf}_{(k-1)} \rightsquigarrow_{\mathbb{SS}_{(j,i)}} \mathsf{conf}_k \rightsquigarrow_{\mathbb{SS}_{(j,i)}} \mathsf{conf}_{(k+1)} \tag{IH13}$$

As we have seen before (Hypothesis IH10), we know that the scheduler semantics consumes all the possible steps of a service, before passing to another by applying the ServerStepLoop *rule or the* ServerStep *one. Hence, we can derive the following.*

$$[\![\mathsf{conf}_{(k-1)}]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k]})}_{\mathbb{SS}_{(j,i)}} \rightsquigarrow^{\text{LoopServerStep}_{(j,i)}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{[],[]}_{\mathbb{SS}_{(j,i)}} \tag{IH14}$$

$$[\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}_{(j,i)}} \rightsquigarrow^{\text{LoopServerStep}_{(j,i)}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]^{[],[]}_{\mathbb{SS}_{(j,i)}} \tag{IH15}$$

$$[\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}_{(j,i)}} \rightsquigarrow^{\text{ServerStep}_{(j,i)}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]^{[],[]}_{\mathbb{SS}} \tag{IH16}$$

$$\mathsf{conf}_{(k+1)} = \mathsf{conf}'_{(k+1)} \tag{IH17}$$

$\Pi'^{\mathfrak{S}}_{[k+1]} = ((j',i'), \mathbb{SS})$ *We rewrite the main hypothesis.*

$$\mathsf{conf} \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[0]}} \ldots \mathsf{conf}_{(k-1)} \rightsquigarrow_{\mathbb{SS}_{(j,i)}} \mathsf{conf}_k \rightsquigarrow_{\mathbb{SS}_{(j',i')}} \mathsf{conf}_{(k+1)} \tag{IH18}$$

The k + 1 transition consider another service to step. By first applying ServerStepsOk*, and then one of the* ServerStepLoop *rule or the* ServerStep *one on* conf_k*.*

$$[\![\mathsf{conf}_{(k-1)}]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k]})}_{\mathbb{SS}_{(j,i)}} \rightsquigarrow^{\text{ServerStep}_{(j,i)}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{[],[]}_{\mathbb{SS}} \tag{IH19}$$

$$\begin{aligned}[\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}} &\rightsquigarrow^{\text{ServerStepsOk}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}_{(j',i')}} \\ &\rightsquigarrow^{\text{LoopServerStep}_{(j',i')}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]^{[],[]}_{\mathbb{SS}_{(j',i')}}\end{aligned} \tag{IH20}$$

$$\begin{aligned}[\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}} &\rightsquigarrow^{\text{ServerStepsOk}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}_{(j',i')}} \\ &\rightsquigarrow^{\text{ServerStep}_{(j',i')}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]^{[],[]}_{\mathbb{SS}}\end{aligned} \tag{IH21}$$

$$\mathsf{conf}_{(k+1)} = \mathsf{conf}'_{(k+1)} \tag{IH22}$$

$\Pi'^{\mathfrak{S}}_{[k+1]} = ((j',i'), \mathbb{RS})$ *We rewrite the main hypothesis.*

$$\mathsf{conf} \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[0]}} \ldots \mathsf{conf}_{(k-1)} \rightsquigarrow_{\mathbb{SS}_{(j,i)}} \mathsf{conf}_k \rightsquigarrow_{\mathbb{RS}_{(j',i')}} \mathsf{conf}_{(k+1)} \tag{IH23}$$

Then

$$[\![\mathsf{conf}_{(k-1)}]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k]})}_{\mathbb{SS}_{(j,i)}} \rightsquigarrow^{\text{ServerStep}_{(j,i)}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{[],[]}_{\mathbb{SS}} \tag{IH24}$$

$$\begin{aligned}[\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}} &\rightsquigarrow^{\text{ServerStepsDone}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{RS}} \\ &\rightsquigarrow^{\text{RetService}_{(j',i')}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]^{[],[]}_{\mathbb{RS}}\end{aligned} \tag{IH25}$$

$$\mathsf{conf}_{(k+1)} = \mathsf{conf}'_{(k+1)} \tag{IH26}$$

$\Pi'^{\mathfrak{S}}_{[k+1]} = \mathbb{DS}$ *We rewrite the main hypothesis.*

$$\mathsf{conf} \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[0]}} \ldots \mathsf{conf}_{(k-1)} \rightsquigarrow_{\mathbb{SS}_{(j,i)}} \mathsf{conf}_k \rightsquigarrow_{\mathbb{DS}} \mathsf{conf}_{(k+1)} \tag{IH27}$$

Then

$$[\![\mathsf{conf}_{(k-1)}]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k]})}_{\mathbb{SS}_{(j,i)}} \rightsquigarrow^{\textsc{ServerStep}_{(j,i)}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{[],[]}_{\mathbb{SS}} \tag{IH28}$$

$$\begin{aligned}[\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}} &\rightsquigarrow^{\textsc{ServerStepsDone}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{RS}} \\ &\rightsquigarrow^{\textit{RetServiceDone}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\mathbb{DS}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{DV}} \rightsquigarrow^{\textsc{DevSens}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]^{[],[]}_{\mathbb{DV}}\end{aligned} \tag{IH29}$$

$$\mathsf{conf}_{(k+1)} = \mathsf{conf}'_{(k+1)} \tag{IH30}$$

$\Pi'^{\mathfrak{S}}_{[k+1]} = ((j',i'),\mathbb{DV})$ *We rewrite the main hypothesis.*

$$\mathsf{conf} \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[0]}} \ldots \mathsf{conf}_{(k-1)} \rightsquigarrow_{\mathbb{SS}_{(j,i)}} \mathsf{conf}_k \rightsquigarrow_{\mathbb{DA}_{(j',i')}} \mathsf{conf}_{(k+1)} \tag{IH31}$$

$$\mathsf{conf} \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[0]}} \ldots \mathsf{conf}_{(k-1)} \rightsquigarrow_{\mathbb{SS}_{(j,i)}} \mathsf{conf}_k \rightsquigarrow_{\mathbb{DR}_{(j',i')}} \mathsf{conf}_{(k+1)} \tag{IH32}$$

Then

$$[\![\mathsf{conf}_{(k-1)}]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k]})}_{\mathbb{SS}_{(j,i)}} \rightsquigarrow^{\textsc{ServerStep}_{(j,i)}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{[],[]}_{\mathbb{SS}} \tag{IH33}$$

$$\begin{aligned}[\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}} &\rightsquigarrow^{\textsc{ServerStepsDone}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{RS}} \\ &\rightsquigarrow^{\textit{RetServiceDone}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\mathbb{DA}_{(j',i')}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{DV}} \rightsquigarrow^{\textsc{DevAct}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]^{[],[]}_{\mathbb{DV}}\end{aligned} \tag{IH34}$$

$$\begin{aligned}[\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}} &\rightsquigarrow^{\textsc{ServerStepsDone}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{RS}} \\ &\rightsquigarrow^{\textit{RetServiceDone}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\mathbb{DR}_{(j',i')}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{DV}} \rightsquigarrow^{\textsc{DevRead}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]^{[],[]}_{\mathbb{DV}}\end{aligned} \tag{IH35}$$

$$\mathsf{conf}_{(k+1)} = \mathsf{conf}'_{(k+1)} \tag{IH36}$$

$\Pi'^{\mathfrak{S}}_{[k+1]} = (j',\mathbb{CS})$ *We rewrite the main hypothesis.*

$$\mathsf{conf} \rightsquigarrow_{\Pi'^{\mathfrak{S}}_{[0]}} \ldots \mathsf{conf}_{(k-1)} \rightsquigarrow_{\mathbb{SS}_{(j,i)}} \mathsf{conf}_k \rightsquigarrow_{\mathbb{CS}_{j'}} \mathsf{conf}_{(k+1)} \tag{IH37}$$

Then

$$[\![\mathsf{conf}_{(k-1)}]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k]})}_{\mathbb{SS}_{(j,i)}} \rightsquigarrow^{\textsc{ServerStep}_{(j,i)}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{[],[]}_{\mathbb{SS}} \tag{IH38}$$

$$\begin{aligned}[\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}} &\rightsquigarrow^{\textsc{ServerStepsDone}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{RS}} \\ &\rightsquigarrow^{\textit{RetServiceDone}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}([]),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{DV}} \rightsquigarrow^{\textsc{DevStepDone}}_{\mathfrak{S}} \\ [\![\mathsf{conf}_k]\!]^{[],[]}_{\mathbb{CS}} &\rightsquigarrow^{\textsc{ClientStepsOk}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{[],[]}_{\mathbb{CS}_{j'}} \rightsquigarrow^{\textsc{LoopClientSteps}_{j'}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]^{[],[]}_{\mathbb{CS}_{j'}}\end{aligned} \tag{IH39}$$

$$\begin{aligned}[\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{SS}} &\rightsquigarrow^{\textsc{ServerStepsDone}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{RS}} \\ &\rightsquigarrow^{\textit{RetServiceDone}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{\pi_{\mathfrak{L}}([]),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1]})}_{\mathbb{DV}} \rightsquigarrow^{\textsc{DevStepDone}}_{\mathfrak{S}} \\ [\![\mathsf{conf}_k]\!]^{[],[]}_{\mathbb{CS}} &\rightsquigarrow^{\textsc{ClientStepsOk}}_{\mathfrak{S}} [\![\mathsf{conf}_k]\!]^{[],[]}_{\mathbb{CS}_{j'}} \rightsquigarrow^{\textsc{ClientStep}_{j'}}_{\mathfrak{S}} [\![\mathsf{conf}'_{(k+1)}]\!]^{[],[]}_{\mathbb{CS}}\end{aligned} \tag{IH40}$$

$$\mathsf{conf}_{(k+1)} = \mathsf{conf}'_{(k+1)} \tag{IH41}$$

other subases *The induction predicate holds for every subcase; hence we conclude the proof by saying that the scheduler's execution can produce every final configuration generated by a WEBIOTexecution, making the semantics equivalent.*

2 Resume of the proof

$\Leftarrow$ *This direction is trivial. The non-deterministic semantics can always imitate a scheduler's step. Indeed, if the scheduler semantics can step, it is possible to do a step with non-deterministic .*

$\Rightarrow$ *We divide the proof into two steps. The first step reorders a non-deterministic trace by applying the commutation lemmas presented in Section* **??** *to create a sequence of intervals in the trace. We show that the trace* Π_n *and the re-arranged (partially-ordered) trace* Π'_n *produce equivalent executions. We prove this step by defining two trace reordering iterations: one for the first iteration of the scheduler, comprehending the* `ServiceInits`*, and the other for the successive iterations, which do not consider the scheduler's first step. For each sub-case, the final configuration is equivalent to the one obtained by executing the on the original one. Hence, the trace transformation is correct for the non-deterministic semantics. The proof comes almost in an algorithmic fashion. Despite its size, this proof step comes straightforward, as we rely on the commutation and the interval lemmas. The initial hypotheses we consider are the following.*

$$wo, \mathcal{E} \vdash \mathsf{conf} \rightsquigarrow^n \mathsf{conf}_n$$
$$\mathsf{conf} = \langle [], \mathcal{I}, HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,\emptyset\}, IC, [] \rangle$$
$$\mathsf{conf}_n = \langle \mathcal{L}, [], HM, \{\emptyset,\emptyset,\emptyset,\emptyset\}, \{\emptyset,\emptyset,\emptyset,CC_e\}, IC', \Pi_n \rangle$$

Moreover, the hypotheses we consider are the one listed in Section 1.1.
For the second step, we consider an ordered trace equivalent to Π_n*, called* $\Pi'_n = \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}}$*. Knowing that they produce equivalent executions, we show that both the scheduler and can simulate the ordered trace. We rewrite the theorem in the following proposition.*

$$\begin{aligned}
&\forall\ wo, \mathcal{E}, n, \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}}, \mathsf{conf}_n, \mathsf{conf}'_n.\ \exists\ t : \\
&\qquad wo, \mathcal{E} \vdash \mathsf{conf}\{[]\} \rightsquigarrow^n \mathsf{conf}_n\{\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}}\} \\
&\qquad\qquad \Rightarrow \\
&(wo, \mathcal{E} \vdash [\![\mathsf{conf}]\!]^{\pi_{\mathfrak{L}}(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}}),\pi_{\mathfrak{T}}(\Pi^{\mathbb{IN}}::\Pi^{\mathfrak{S}})}_{\mathbb{IN}} \rightsquigarrow^{(n+t)}_{\mathfrak{S}} [\![\mathsf{conf}'_{(n)}]\!]^{[],[]}_{\mathtt{end}}) \ \wedge\ \mathsf{conf}'_n = \mathsf{conf}_n
\end{aligned}$$

For simplicity, we rename $\Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}}$ *as* $\Pi'^{\mathfrak{S}}$*, where* $|\Pi'^{\mathfrak{S}}| = n$*. If the trace size* n *is* 0*, the proof is trivial. Otherwise, we define an induction predicate stating that the scheduler can always imitate the reordered trace for some generic* $k + t$ *steps, where* t *is the number of scheduler transitions that do not change the configuration. We show the induction predicate in the following proposition:*

$$\begin{aligned}
&\forall n, k, \Pi'^{\mathfrak{S}}, \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}}, \mathsf{conf}, \mathsf{conf}_k. \\
&\quad \Pi^{\mathbb{IN}} :: \Pi^{\mathfrak{S}} = \Pi'^{\mathfrak{S}} \quad \wedge \quad |\Pi'^{\mathfrak{S}}| = n \quad \wedge \quad \mathsf{conf} \rightsquigarrow^{k}_{\Pi'^{\mathfrak{S}}_{[0,k]}} \mathsf{conf}_k \\
&\qquad\qquad \Rightarrow \\
&\qquad\qquad \exists t, \mathbb{X}. \\
&[\![\mathsf{conf}]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}})}_{\mathbb{IN}} \rightsquigarrow^{(k+t)}_{\mathfrak{S}} [\![\mathsf{conf}'_k]\!]^{\pi_{\mathfrak{L}}(\Pi'^{\mathfrak{S}}_{[k+1,n]}),\pi_{\mathfrak{T}}(\Pi'^{\mathfrak{S}}_{[k+1,n]})}_{\mathbb{X}} \quad \wedge \\
&\qquad\qquad \mathsf{conf}'_k = \mathsf{conf}_k
\end{aligned}$$

, where $\Pi'^{\mathfrak{S}}_{[0,k]}$ *is the subtrace of* $\Pi'^{\mathfrak{S}}$ *from the index* 0 *to* k*.*
We assume that this predicate hold for k *steps. We show that it also holds for every next step* $k + 1$*. We go by case analysis on* k*, showing that the scheduler and the non-deterministic semantics can do* $k + t$ *and* k *steps respectively, following the trace* $\Pi'^{\mathfrak{S}}_{[0,k]}$*.*
Then, for each possible sub-case, we show that both the semantics can perform one more execution step to produce the same $k + 1$ *configuration.*
The induction predicate holds for every subcase; hence we conclude the proof by saying that the scheduler's execution can produce every final configuration generated by a execution, making the semantics equivalent.