CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Physical Space in Reconfigurable Interacting Systems

Implementing physical space in a multi agent system

Midterm report master's thesis in Computer science and engineering

Tom de Ridder

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

MASTER'S THESIS 2024

Physical Space in Reconfigurable Interacting Systems

Implementing physical space in a multi agent system

Tom de Ridder

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Physical Space in Reconfigurable Interacting Systems
Implementing physical space in a multi agent system
Tom de Ridder

Supervisor: Yehia Abd Alrahman, Department of Computer Science and Engineering
Examiner: Nir Piterman, Department of Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2024

Physical Space in Reconfigurable Interacting Systems
Implementing physical space in a multi agent system
Tom de Ridder
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Abstract text about your project in Computer Science and Engineering.

Keywords: Computer, science, computer science, engineering, project, thesis.

Acknowledgements

Here, you can say thank you to your supervisor(s), company advisors and other people that supported you during your project.

Name Familyname, Gothenburg, 2024-04-15

Contents

List of Figures

List of Tables

1

Introduction

In our high-tech society complex systems with large numbers of processes are becoming increasingly more common. You can find them in nearly all fields, each with different application purposes. There are various reasons for these advancements, such as safety, efficiency and costs. Many jobs require heaps of communication and cooperation to be executed correctly, whether it is communication between people or between machines. For humans, it is natural to communicate with the people in the same room while performing such tasks, while not having to deal with people in a different location doing an unrelated task. The reason for this is that our brain can comprehend physical space and can communicate within this space without the need for a system. For machines however, this is not the case. Often communication between machines goes through global channels. To these channels multiple groups of machines, also groups doing different jobs, are connected. Even if the machines doing their collaborative tasks, are in the same room, they still use these globally managed systems.

As you can imagine, handling the communication through such a global system costs a lot of resources. It takes time to communicate back and forth and in addition you also need more processing space for the large protocols filtering the relevant information from the irrelevant information. Short-range communication also reduces the amount of noise and information loss[17], which can make a crucial difference in certain situations. For these reasons making the communication of these multi-process systems local in regards to physical space, helps to better utilize the often limited resources and increase the overall performance.

1.1 Problem and goals

The main problem which needs to be solved is on how to model physical space in reconfigurable interacting systems. These type of models describe how a system consisting of multiple processes behaves. This includes both the individual behavior of the processes as well as the communication and interactions between them. [15].

The presented solution will have to specify how the physical location of a system affects its own behavior, as well as how it affects its behavior in relation to systems co-located in the same space. To be able to do this, the physical space will have to be implemented as a first-class citizen. This would allow us to reason about their environment based behavior in the same way as the interactions amongst machines themselves. They should be able to share the information amongst each other, deduce it from their environment, and use it to perform the appropriate actions. This all should be implemented in a discrete way to allow for scaling and adjustments.

1.2 Thesis layout

After this introduction, we will first go into the background of the topic. We will start by explaining some relevant, recurring, terms for this thesis. After that we will go into more detail about the various types of systems and their building blocks. From this point we will look at known research and results. This includes a literature review on relevant papers and identifying where progress can be made.

Next up the syntax and the semantics of our system will be presented. This will be done in a step by step manner with the help of examples and important lemmas will be formulated with their formal proofs presented in the appendix. We will start with the syntax, then the semantics and finally the locality of communication will be discussed and implemented.

We will then, with the help of the aforementioned lemmas and examples, we will evaluate the developed system on space and time efficiency.

Lastly we will conclude our research, summarize what has been achieved, and why the relevance of it relevant. We will also discuss what areas can still improve and talk about possible future work.

2

Literature review

In this chapter we will go over background information related to our topic and look at relevant papers to identify gaps in the research. We will first go over the kind of systems we are interested in and how these are defined. This includes the calculi these models are based on. Secondly we will specify the definition of physical space, some possible representations of this in computer systems and how they can be related to the system. Lastly we will look at specific relevant research, and identify where advancements can be made. This should give us a good starting point for our own work.

2.1 Background information

2.1.1 System types

As the title of this thesis suggests, the type of systems this research is focused on are reconfigurable interacting systems. These type of systems consist of a number of entities which are able to adapt their configuration in alignment with their current objective(s) [2]. The different entities are able to communicate together, all be it competitive or collaborative. We will focus on collaborative systems, meaning the communication in our system will used to reach a common goal.

2.1.1.1 Multi-agent systems

One type of reconfigurable interecting systems is the multi-agent system(MAS). The concept of MAS has been around since the early 1970s, when it arose from the interest in distributive artificial intelligence [3]. In MAS the earlier mentioned entities are called agents. The agents represent individual autonomous systems, each with their own subgoals, knowledge and behavior. When they first arose, the focus laid mainly on the communication between these agents. When they caught traction in the 1990s [4], their frameworks became more complex allowing for expansion of their applications.
MAS are now used in a large variety of fields. Some examples are: economical market simulation, system diagnosis, and surveillance [7]. As their application range widens, the demand for more expressive ways of modeling grows. It is not just about communication anymore, but also about more adaptability and environmental awareness. The concept of self-driving cars is an example of MAS. Each car is an autonomous agents, which for safety reasons needs to have environmental awareness and also has to communicate with other nearby cars to adjust their behavior on time. In most places, including Sweden, the self-driving function of cars is still prohibited to use on the public road with the exception of approved safety trials [8]. This shows us advancements can still be made in this field and highlights the relevance of our research.

2.1.1.2 Collective-adaptive systems

Collective-adaptive systems (CAS) are a kind of MAS. These systems consist of a large number of heterogeneous components, which organize themselves to form collectives [9]. Just like MAS, the collectives in CAS work together to achieve their individual and global goals. This is ideally done with minimal to no human interaction, requiring complex behavior and communication modelling.
CAS focus on optimizing the limited resources of a system. During runtime the different components are able to join and leave collectives at anytime, distributing their resources based on demand. Behavior like this makes CAS scalable, but also makes their boundaries continuous. When adding an environment, especially an unpredictable one like the earlier example of self-driving cars, can make the modelling quite tedious. Especially for arguing about correctness (and therefore safety) of the integration of CAS, new techniques are needed. This is therefore the type of systems this thesis will focus on.

2.1.1.3 Channeled communication systems

Communication between components in a system is done over a network. For communication to happen, they have to be connected to the same network. Smaller systems can use singular communication channels, but for large scale systems, this would also mean all components receive all the information in a system and have to filter out the relevant messages, this communication style is called broadcasting. For the organization of components, this can come in useful, but we want to limit the amount of noise in a system. A way to deal with this, is by adding channels to a system's network. Channels ensure only components connected to the specific channel receive the information sent [10].
Channeled communication can be done in different ways, the earlier mentioned broadcast being the main method. Broadcast means everyone receives the sent message, think of a TV-broadcast everyone can watch. The second type of communication is multicast, in this case everyone connected to a certain channel receives the message. The last method we will use is called unicast, as the name suggest, this is a communication where the message only reaches a single receiver. Unicast in communication systems is often a request sent by one process and a response to that request by an other process.
Channeled communication can also be used for CAS, but as limited human interaction is desired, it has to be modelled in such a way that components and processes can organize the joining and leaving of these channels by themselves. As mentioned earlier broadcast communication can help with this self-organization. As the example in [13] shows, a line-agent requests the required type and number of robots to join his multicast channel via the broadcast channel. Only after gathering the correct number and types, the production starts. With the specification, no human interference is necessary for them to complete the task. This is the kind of self-organization we want to make possible.

2.1.2 Caculi

In modern MAS, agents are not limited to simple tasks and behavior. Their models can become quite complex and as a results reasoning about their behavior becomes more difficult. Proving the correctness of programs is essential, for some systems even more important than others [5]. For CAS this is no different, whether it is for economical reasons or safety reasons, as the agents in MAS are not always just machines [6]. To be able to prove general correctness for complex systems, discrete models can be used. A popular way for modeling MAS, and therefore also CAS, is by process calculus. A process calculus is a formal mathematical framework for modeling concurrent systems. A large number of process calculi have been developed over

the years, often for slightly different purposes. We will go over two of the most prominent calculi, calculus of communicating systems and Π-calculus, often used as corner stones for the development of new ones.

2.1.2.1 Calculus of communicating systems

The calculus of communicating systems(CCS) was first introduced by Milner in the 1980s [11]. Since then it has contributed greatly to the development of MAS and other communicating systems. CCS is a mathematical framework to define the behavior of communication of non-deterministic finite transition systems. The framework allows for the defining these systems and arguing about its correctness and its behavior [20]. As CCS deals with transition systems, it shows how over a time such systems change and adapt depending on actions taken. Even this older calculus already used channeled communication, however the channels were predefined. Processes could communicate if they shared a channel and a compatible action in the model, which restricted the expressiveness of the models. Milner himself realized this as well and continued on contributing to this area.

2.1.2.2 Pi-calculus

The continuation of his research made him later develop one of the most famous process calculi, Π-calculus [12]. This calculus is an extension of CCS and tackled the problem of limited expressiveness. Instead of predefined actions, it utilizes variables which can take on a wide range of formats and values and can be sent between processes. These values can be used while performing actions, such a defining which channel to send information on or when to perform an action. As no predefined channels have to be used, processes can dynamically be deleted or created according to the demand. Our goal is to create a model for CAS, requiring the ability to self-organize and adjust accordingly. For that reason our framework will be based on Π-calculus. Besides creating our own model, we will also have to extend it to include the ability to reason about physical space.

2.1.3 Physical space

Physical space is the space humans perceive. Think of the room you are in, the building the room is in, etc. For us it is easy to limit our interactions to the space we are in and also to stop the communication with people in this space when we leave. For machines this is not as simple as they perceive space in a different way [16]. Often they have to be instructed on where to go and what movements to make, and communication is done over global channels or channels assigned on a global level. Just like people, machines prefer short-distance communication. [17]. Short-distance communication limits both noise and information loss. Think of talking to someone next to you in comparison to talking to someone across the room.
The physical space will have to be implemented as a first-class citizen. This allows different components of our system to move without the need for other components to wait for the movement to be completed. It also provides the components the ability to store it in a data structure and to adjust their behavior based on it. The different spaces we will call contexts, each with their own environmental properties. Just like in real-life, these contexts are continuous, this means that not only the processes can move and change contexts, but that these spaces can also move themselves. This means the initial layout of our space can change and we will have to find a way to model both their own behavior as well as their relation to other contexts. There have been various papers on logical reasoning of spatial relation, such as [21]. A number of spatial relations are

possible, but due to limited time, the focus will lie on whether a pair of contexts is (externally) connected or disconnected. These two relations provide the basis for reasoning about the ability of processes to change from one environment to an other, while limiting the space completeness.

2.2 Related papers

2.2.1 R-Check

One of the recently developed agent-based modelling programs in this field is R-CHECK[14]. As the name suggests, it is able to check the model of MAS on correctness. R-CHECK allows support for high-level input language with matching enumerative and symbolic semantics. Currently it supports communication between machines to allow them to self-organize and form coalitions. The program allows its users to build simulations and analyze these agent-based models. The user is able to check their models for synchronization, interaction protocols and self-organization. These behaviors are something we want in our model as well, therefore using R-CHECK as inspiration for our framework can be beneficial. It might even allow for the extension of the model checker based on the results of this paper.
R-CHECK is build on `LTOL`, this is an extension of `LTL` (Linear Time Temporal) logic. Where `LTOL` differs from `LTL` is the next operator. In `LTOL` this operator is replaced by observation descriptors *possible*$\langle O \rangle$ and *necessary[O]*. These descriptors are used to refer to messages or the intended set of receivers. This extension allows for reasoning about agent-interaction. In R-CHECK, only a single message can be sent at a time and the information carried in these messages can influence the behavior of the other agents. Moreover, agents can establish connections themselves and can therefore organize themselves.
What is however missing from R-CHECK is native support for reasoning about physical space. In other words, machine locations are currently still second-class citizens [18]. A second-class citizen in terms of computational science is a job which is either done manually or by simple implementations. First-class citizens on the other hand allow for queuing, scheduling and managing amongst other things, just like R-CHECK support for computational jobs. By having this native support, movements of systems and their context based interactions can be modeled in a discrete way. As R-CHECK only allows a single message being sent at a time, having the context behavior as a second-class citizen, would result in significantly slower systems as processes would have to wait on each other to complete this action. By separating the context behavior from the system behavior, we can have seperate reasoning, influenced by the context definition rather than the system defintion. The implementation of this would make it easier and more reliable for various machines to establish and have connections with each other while in the same space. It would make collaborations between multiple machines both more efficient and also more reliable and thus perform collaborative tasks better. This will therefore be the focus of this thesis.

3

The system

3.1 The syntax

The first step to implementing physical space into multi agent systems is having a syntax to express our systems. We will explain the syntax with the support of a step by step example for clarification.

Systems	$S ::=$	$\Gamma : P \quad \mid \quad S_1 \mid\mid S_2$
Environment	$\Gamma ::=$	$\langle \gamma : \mathscr{X} \to \mathscr{V}, \quad \mathsf{LS} \rangle$
Processes	$P ::=$	$0 \quad \mid \quad \Pi!(\tilde{E})^E.U \quad \mid \quad \Pi?(\tilde{x})^E.U \quad \mid \quad \mathsf{S}(\tilde{E})^E.U \quad \mid$
		$\Pi\mathsf{G}(\tilde{x})^E.U \quad \mid \quad \langle \Pi \rangle P \quad \mid \quad P_1 + P_2 \quad \mid \quad K(x_1,...,x_n)$
Updates	$U ::=$	$U[\tilde{E}/\tilde{x}] \quad \mid \quad P \quad \mid \quad \{\mathtt{OP}(\mathsf{LS})\}U$
Expressions	$E ::=$	$v \quad \mid \quad x \quad \mid \quad ch \quad \mid \quad self.x \quad \mid \quad E_1 \otimes E_2$

Table 3.1: The syntax of our calculus.

At the top level we can see the systems S. A system is either a process with an environment or a parallel composition of two systems, $S_1 \mid\mid S_2$. The environment of a process, Γ, consists of two things. The first one is γ, with the form $\mathscr{X} \to \mathscr{V}$, a set of attributes mapped to a set of values of any format. The second one is LS, a set of channels the process is listening to. We will explain more about the properties and function over this set later on in this section.
Now that we have gone over the top layer of our syntax, we will present the setup for our example. We will mention the channels, but will go further into how the different types of channels work later on.

Example 3.1.1 *[Product fetching (step 1/5)] For our example, we will take a simple setup to make it clear how the channeled communication works. Imagine a warehouse where certain products will have to be collected. Before sending a system to collect the product, it will have to be checked if the needed product is available. If the inventory is available, a manager will have to instruct a worker to fetch the requires product. If the product is unavailable it will have to be refilled. In this section we will not focus on the case of it being empty for simplicity reasons. Later in the thesis we will present the example in full, but as the purpose now is to show how the transitions and examples would look on a process and on a system level, we will keep it simplistic. It will still include all three kinds of communication, namely: broadcast, multicast and unicast. The example will build itself up from the composition, to the specifications, to the process level transitions and finally to the system level transitions. We add the joining and leaving of channels when location comes into play. For now all the processes already have the necessary process in their environment.*

As can be seen in 3.1, a process can take 8 different forms. The first form is the empty process. As the name suggests, this is a process without any function and therefore will stay an empty process. At first glance it

might seem useless, but this is very useful as it means we can terminate a process after it has done its part. The process being empty does not mean that the environment is empty, however we will not be able to access this environment anymore.
Before we go over the other processes, we will look at their building blocks. Starting with the predicate Π, which can either evaluate to $True$ or $False$. It is a logical expression over the environment, and can include the basic logical boolean operators. Only processes satisfying the predicate will be able to receive a message or perform their action.
An expression E can be a value, an attribute or a channel. It can also include a pointer to an attribute in its own environment, $self.x$, or consists of an operation over expressions. $\tilde{E}$ presents a sequence of one or more expressions, allowing for passing multiple values. With these clarifications, we will first go over the last 3 process expressions.
We have a guarded process $\langle\Pi\rangle P$, this process has an internal guard. It can only continue with the process P if Π is satisfied in its own environment. As we do not have parallel process in our syntax, when $\langle\Pi\rangle P$ and $\Pi = False$ under gamma, the process will be blocked for the remainder of the runtime, unless there is a non-syntax related cause, for example it will have to be charged until a certain percentage. Our focus is more on the interaction of processes, and for that where a guard can be helpful is if you want an internal value to determine the next action, which brings us to our next process form.
The summation, or $P_1 + P_2$, here the process will either continue with P_1 or P_2, but not both. This choice can be non-deterministic and the non-performed process will be dropped, but more on that later. Combining this with the guarded process can allow us to influence which process will be continued with based on the internal environment. We will also use this in our example later on.
The definition process, $K(x_1,...,x_n)$, is a process with its defined attributes. By having this in our syntax, we can express what attributes a process has and where this is internally. Note that for attributes, we cannot have duplicated or the pointers would get messed up. This means we require $x_i \neq x_j iff i \neq j$ for all $1 \leq i, j \leq n$.

For evaluating Π under the environment there are two possible notations. The standard one $\gamma \models \{\Pi\}_\gamma$, which takes the current environment and determines if the predicate holds in that environment. The other notation is $\gamma \models \{\Pi[\tilde{v}/\tilde{x}]\}_\gamma$, this means the predicate should hold with the newly received information. $[\tilde{v}/\tilde{x}]$ stands for substituting the value of x with the corresponding v for each value and attribute in the sequences.

Example 3.1.2 *[Product fetching (step 1/5)] Our system S consists of 4 processes and can be defined as $S \triangleq C \mid\mid I \mid\mid O \mid\mid R$. Here we have C which checks the inventory with I and then sends the request on to O, which orders R to retrieve the desired product.*

This brings us to the communication processes. We will start with the standard send and receive processes, which cover broadcast and multicast, and then cover the get and supply processes of unicast.
To begin $\Pi!(\tilde{E})^E.U$, the sending process. Here Π is not an internal guard but rather a predicate for which processes should be able to receive the message. It will be concertized under the environment as $\{\Pi\}_\gamma = \pi'$ and this predicate will be sent along with the message. π' only depends on the current values in the receiving process, think of example, their role, battery level, or anything else which can determine if the process should receive the message. The sending process will evaluate the expression $\tilde{E}$ to $\tilde{v}$, the values to send, and also the expression E to ch, for which channel it will send on. Note that this is not a sequence, so a message can only be sent on a single channel at a time. After sending it will proceed to update its environment about which we will give more details later on what this means.
Now for the other end of this interaction, the receiving process $\Pi?(\tilde{x})^E.U$. Here Π is different from π', which the process will receive with the message. This guard is of the earlier explained form $\{\Pi[\tilde{v}/\tilde{x}]\}_\gamma$. Both Π

and π' will have to be satisfied in the environment for the process to be able to receive the information. As we can see by $\tilde{x}$, when a process is expecting a message, it is expecting values for certain attributes. Meaning it is important the combination of the two guards make sure a message only reaches certain processes. This however, is about the specification, not the syntax itself, and differs per setting. After receiving the values, the process will update its current attributes with the received values.
The send and receive actions can be done on either broadcast or unicast. As the name suggests, broadcast means all processes will get the message as the broadcast channel $*$, is always in LS. if $ch \neq *$, we are dealing with unicast. Only processes with $ch \in \mathsf{LS}$ will receive this message, but this can still be multiple processes.
For unicast we have the process requesting to get information, $\Pi\mathsf{G}(\tilde{x})^E.U$. We have Π, which is the same as Π in the sending process and is concertized to π' and sent along with the request. It sends a sequence of attributes it would like the values of over the channel evaluated from E. After receiving the requested information, it updates its attributes with the received values, like the previous receiving process.
The get request from the previous process is answered by the supply process, $\mathsf{S}(\tilde{E})^E.U$. As you can see it does not have a guard or predicate, but it still has to satisfy π'. The reason this process does not have its own guard is that it is not receiving any values to substitute like the receiving process and as it is answering a request, it does not have to send a predicate with its message, like the send and get processes do. The process evaluates $\tilde{E}$ in accordance to the requested attribute values and sends them back over the same channel, which can be evaluated from E. It then performs any internal updates, but these are not necessarily based on the sent information.

Now that we have explained the syntax, we can specify the process from our example.

Example 3.1.3 *[Product fetching (step 3/5)] Here are the specifications of the processes in 3.1.2.*

$$
\begin{array}{ll}
C \triangleq & (product = 2)G(x)^{self.id}.[self.available := x] \\
& \big(\langle self.available \geq 1\rangle(role = \text{"}manager\text{"})!(\text{"}start\text{"}, 2)^*.0 + \langle self.available < 1\rangle u.0\big) \\
I \triangleq & S(self.inventory)^{self.ch}.[self.inventory := self.inventory - 1].I \\
O \triangleq & (x = \text{"}start\text{"})?(x,y)^*.[self.product := y] \\
& (status = 0, role = \text{"}receiver\text{"})!(self.product)^{self.id}.O \\
R \triangleq & (x = 1)?(x)^{self.ch}.[self.product := x, self.status := 1]p_1 \quad + \\
& (x = 2)?(x)^{self.ch}.[self.product := x, self.status := 1]p_2 \quad + \\
& (x = 3)?(x)^{self.ch}.[self.product := x, self.status := 1]p_3
\end{array}
$$

We can see that we have some attributes, which we will explain. *id* is the id of a process and is unique. *product* is the number of the requested product. *available* is the number of products of the requested product currently available. *role* is the role of a process and is can be *manager,receiver,commander,inventory*. *inventory* is the current inventory size observed, which is the local name of *available*. *Status* is the status of the process which is either 0 or 1, for busy or not busy. *ch* is the current channel a process is on and should be in *ch*.

3.2 The semantics

Now that we have our syntax down, we can move on to the rules, or the semantics. We will go over process level semantics first, where the transitions are noted by $\mapsto$. Here the possible transition labels are λ, the send,

receive, get and supply transitions.

$$\lambda ::= \Pi!(\tilde{v})^{ch} \quad | \quad \Pi?(\tilde{v})^{ch} \quad | \quad \Pi\mathsf{G}(\tilde{v})^{ch} \quad | \quad \Pi\mathsf{S}(\tilde{v})^{ch}$$

The discard transition, only possible on the receive transition is denoted by $\widetilde{\Pi?(\tilde{v})^{ch}}$, in the rule naming this will be denoted by an N in front of the name.

$$\texttt{Snd}\ \frac{[\![\tilde{E}]\!]_\gamma = \tilde{v} \quad \{\Pi\}_\gamma = \pi' \quad [\![E']\!]_\gamma = ch}{\langle \gamma, \mathsf{LS}\rangle : \Pi!(\tilde{E})^{E'}.U \xrightarrow{\pi'!(\tilde{v})^{ch}} \{\!|\langle \gamma, \mathsf{LS}\rangle : U|\!\}}$$

$$\texttt{Rec}\ \frac{[\![E']\!]_\gamma = ch \quad \gamma \models \{\Pi[\tilde{v}/\tilde{x}]\}_\gamma \quad \gamma \models \pi' \quad ch \in \mathsf{LS}}{\langle \gamma, \mathsf{LS}\rangle : \Pi?(\tilde{x})^{E'}.U \xrightarrow{\pi'?(\tilde{v})^{ch}} \{\!|\langle \gamma, \mathsf{LS}\rangle : U[\tilde{v}/\tilde{x}]|\!\}}$$

$$\texttt{Get}\ \frac{[\![E]\!]_\gamma = ch \quad \{\Pi\}_\gamma = \pi'}{\langle \gamma, \mathsf{LS}\rangle : \Pi\mathsf{G}(\tilde{x})^{E}.U \xrightarrow{\pi'\mathsf{G}(\tilde{v})^{ch}} \{\!|\langle \gamma, \mathsf{LS}\rangle : U[\tilde{v}/\tilde{x}]|\!\}}$$

$$\texttt{Sup}\ \frac{[\![\tilde{E}]\!]_\gamma = \tilde{v} \quad [\![E']\!]_\gamma = ch \quad \gamma \models \pi' \quad ch \in \mathsf{LS}}{\langle \gamma, \mathsf{LS}\rangle : \mathsf{S}(\tilde{E})^{E'}.U \xrightarrow{\pi'\mathsf{S}(\tilde{v})^{ch}} \{\!|\langle \gamma, \mathsf{LS}\rangle : U|\!\}}$$

$$\texttt{Grd}\ \frac{\gamma \models \Pi \quad \langle \gamma, \mathsf{LS}\rangle : P \xmapsto{\lambda} \langle \gamma', \mathsf{LS}'\rangle : P'}{\langle \gamma, \mathsf{LS}\rangle : \langle \Pi\rangle P \xmapsto{\lambda} \langle \gamma', \mathsf{LS}'\rangle : P'}$$

$$\texttt{Stc}\ \frac{\gamma \models \Pi \quad \langle \gamma, \mathsf{LS}\rangle : P \xmapsto{\widetilde{\pi'?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}\rangle : P}{\langle \gamma, \mathsf{LS}\rangle : \langle \Pi\rangle P \xmapsto{\widetilde{\pi'?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}\rangle : \langle \Pi\rangle P}$$

$$\texttt{Nsnd}\ \frac{}{\langle \gamma, \mathsf{LS}\rangle : \Pi!(\tilde{E})^{E'}.U \xrightarrow{\widetilde{\pi'?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}\rangle : \Pi!(\tilde{E})^{E'}.U}$$

$$\texttt{Nrec}\ \frac{ch \notin \mathsf{LS} \ \vee \ \big(ch = * \ \wedge \ (\gamma \not\models \{\Pi[\tilde{v}/\tilde{x}]\} \ \vee \ \gamma \not\models \pi')\big)}{\langle \gamma, \mathsf{LS}\rangle : \Pi?(\tilde{x})^{E'}.U \xrightarrow{\widetilde{\pi'?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}\rangle : \Pi?(\tilde{x})^{E'}.U}$$

$$\texttt{Nget}\ \frac{}{\langle \gamma, \mathsf{LS}\rangle : \Pi\mathsf{G}(\tilde{x})^{E}.U \xrightarrow{\widetilde{\pi'?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}\rangle : \Pi\mathsf{G}(\tilde{x})^{E}.U}$$

$$\texttt{Nsup}\ \frac{ch \notin \mathsf{LS} \ \vee \ ch = *}{\langle \gamma, \mathsf{LS}\rangle : \mathsf{S}(\tilde{E})^{E'}.U \xrightarrow{\widetilde{\pi'?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}\rangle : \mathsf{S}(\tilde{E})^{E'}.U}$$

$$\texttt{Blk}\ \frac{\gamma \not\models \Pi}{\langle \gamma, \mathsf{LS}\rangle : \langle \Pi\rangle P \xrightarrow{\widetilde{\pi'?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}\rangle : \langle \Pi\rangle P}$$

Table 3.2: The process level semantics 1/2.

We can see that certain rules can always be performed. Either because they only evaluate something or because they have no conditions. These are `Snd`, `Nsnd`, `Get`, `Nget`. For `Snd` and `Get` the reason is that when a process is ready to send or request information, it means it is already able to do this and should proceed when possible. This also means they are not expecting to receive anything and should be able to discard any incoming messages, hence `Nsnd` and `Nget`. This has to do with non-blocking behavior and we will go more in dept on that later on in our lemmas.
As previously explained for receiving a message through `Rec` or `Sup` we have some conditions to be able to receive it. If these do not hold however, we cannot discard the incoming message like for the previous two cases. As we can see one of the conditions is that a process is not listening to a certain channel, which would means they will not even be aware of the message. The other possible condition for discarding a message requires it to be over the broadcast channel. This has to do with the blocking of multicast and unicast, which we will show later in our lemmas. The reason for this blocking behavior is that we do not want processes to be on a non-broadcast channel unless they are participating and ready to continue to the next step.
The guard works like explained before, where the condition of discarding in `Stc` shows that it is only possible when the process is ready and can discard, however if the internal guard does not hold, `Blk` shows it can discard either way. The example of charging comes to mind, where we do not want a charging process to

block.
From the rules it shows that after performing an action, it can update its environment with $\{\!|A|\!\}$.

The update function $\{\!|A|\!\}$ is of the following form:

$$\{\!|A|\!\} = \begin{cases} \{\!|\langle \gamma[\tilde{x} \mapsto [\![\tilde{E}]\!]_\gamma], \mathsf{LS}\rangle : U|\!\}, & A = \langle \gamma, \mathsf{LS}\rangle : U[\tilde{v}/\tilde{x}] \\ \{\!|\langle \gamma, \mathtt{OP}(\mathsf{LS})\rangle : U|\!\}, & A = \langle \gamma, \mathsf{LS}\rangle : \{\mathtt{OP}(\mathsf{LS})\}U \\ \langle \gamma, \mathsf{LS}\rangle : P & A = \langle \gamma, \mathsf{LS}\rangle : P \end{cases}$$

The operation $\mathtt{OP}(\mathsf{LS})$ on the set of listening channels, can add or remove channels from this set, but LS will always include the broadcast channel $*$.

$$\mathtt{Lor}\ \frac{\langle \gamma, \mathsf{LS}\rangle : P_1 \overset{\lambda}{\mapsto} \langle \gamma', \mathsf{LS}'\rangle : P_1'}{\langle \gamma, \mathsf{LS}\rangle : P_1 + P_2 \overset{\lambda}{\mapsto} \langle \gamma', \mathsf{LS}'\rangle : P_1'} \qquad \mathtt{Ror}\ \frac{\langle \gamma, \mathsf{LS}\rangle : P_2 \overset{\lambda}{\mapsto} \langle \gamma', \mathsf{LS}'\rangle : P_2'}{\langle \gamma, \mathsf{LS}\rangle : P_1 + P_2 \overset{\lambda}{\mapsto} \langle \gamma', \mathsf{LS}'\rangle : P_2'}$$

$$\mathtt{Nor}\ \frac{\langle \gamma, \mathsf{LS}\rangle : P_1 \overset{\widetilde{\pi'?(\tilde{v})^{ch}}}{\longmapsto} \langle \gamma, \mathsf{LS}\rangle : P_1 \quad \langle \gamma, \mathsf{LS}\rangle : P_2 \overset{\widetilde{\pi'?(\tilde{v})^{ch}}}{\longmapsto} \langle \gamma, \mathsf{LS}\rangle : P_2}{\langle \gamma, \mathsf{LS}\rangle : P_1 + P_2 \overset{\widetilde{\pi'?(\tilde{v})^{ch}}}{\longmapsto} \langle \gamma, \mathsf{LS}\rangle : P_1 + P_2}$$

$$\mathtt{Def}\ \frac{\langle \gamma, \mathsf{LS}\rangle : P \overset{\lambda}{\mapsto} \langle \gamma', \mathsf{LS}'\rangle : P' \quad K(\tilde{x}) \triangleq P}{\langle \gamma, \mathsf{LS}\rangle : K(\tilde{x}) \overset{\lambda}{\mapsto} \langle \gamma', \mathsf{LS}'\rangle : P'} \qquad \mathtt{Ndef}\ \frac{\langle \gamma, \mathsf{LS}\rangle : P \overset{\widetilde{\pi'?(\tilde{v})^{ch}}}{\longmapsto} \langle \gamma, \mathsf{LS}\rangle : P \quad K(\tilde{x}) \triangleq P}{\langle \gamma, \mathsf{LS}\rangle : K(\tilde{x}) \overset{\widetilde{\pi'?(\tilde{v})^{ch}}}{\longmapsto} \langle \gamma, \mathsf{LS}\rangle : P}$$

$$\mathtt{Nnul}\ \frac{}{\langle \gamma, \mathsf{LS}\rangle : 0 \overset{\widetilde{\pi'?(\tilde{v})^{ch}}}{\longmapsto} \langle \gamma, \mathsf{LS}\rangle : 0}$$

Table 3.3: The process level semantics 2/2.

The other half of the semantics show how the different cases of the summation notation work, it shows that the 0 process will never be blocking and that the definition process can be treated like any other process.

Example 3.2.1 *[Product fetching (step 4/5)] We can now look at the process level transitions from our example. The environments are as follows:*
$\Gamma_C = \langle \gamma : id = 1, product = 2, available = 0, role = \text{"}commander\text{"}, ch = this.id, \mathsf{LS} = *, 1\rangle$
$\Gamma_I = \langle \gamma : id = 2, product = 2, available = 4, role = \text{"}inventory\text{"}, ch = 1, \mathsf{LS} = *, 1\rangle$
$\Gamma_O = \langle \gamma : id = 3, product = 0, role = \text{"}manager\text{"}, ch = self.id, \mathsf{LS} = *, 3\rangle$
$\Gamma_R = \langle \gamma : id = 4, product = 0, status = 0, role = \text{"}receiver\text{"}, ch = 3, \mathsf{LS} = *, 3\rangle$
We get the following transitions in order, with U_i being the continuation of the specification of i:
$\langle \gamma_C, \mathsf{LS}_C\rangle : C \overset{(product=2)\mathsf{G}(4)^1}{\longmapsto} \{\!|\langle \gamma_C', \mathsf{LS}_C'\rangle : [self.available := x]U_C|\!\}$ *by* `Get`
$\langle \gamma_I, \mathsf{LS}_I\rangle : I \overset{(product=2)\mathsf{S}(4)^1}{\longmapsto} \{\!|\langle \gamma_I', \mathsf{LS}_I'\rangle : [self.inventory := self.inventory - 1].I|\!\}$ *by* `Sup`
$\langle \gamma_C', \mathsf{LS}_C'\rangle : U_C \overset{(role=\text{"}manager\text{"})!(\text{"}start\text{"},2)^*}{\longmapsto} \{\!|\langle \gamma_C'', \mathsf{LS}_C''\rangle : 0|\!\}$ *by* `Lor`
$\langle \gamma_O, \mathsf{LS}_O\rangle : O \overset{(role=\text{"}manager\text{"})?(\text{"}start\text{"},2)^*}{\longmapsto} \{\!|\langle \gamma_O', \mathsf{LS}_O'\rangle : [self.product := 2].U_O|\!\}$ *by* `Rec`
$\langle \gamma_I', \mathsf{LS}_I'\rangle : I \overset{\widetilde{(role=\text{"}manager\text{"})?(\text{"}start\text{"},2)^*}}{\longmapsto} \langle \gamma_I', \mathsf{LS}_I'\rangle : I$ *by* `Nrec`

$\langle \gamma_R, \mathsf{LS}_R \rangle : R \xmapsto{(role=\text{"}\widetilde{manager}\text{"})?(\text{"}start\text{"},2)^*} \langle \gamma_R, \mathsf{LS}_R \rangle : R$ *by* `Nrec`

$\langle \gamma'_O, \mathsf{LS}'_O \rangle : U_O \xmapsto{(status=0,role=\text{"}receiver\text{"})!(2)^3} \{\!| \langle \gamma''_O, \mathsf{LS}''_O \rangle : O |\!\}$ *by* `Snd`

$\langle \gamma_R, \mathsf{LS}_R \rangle : R \xmapsto{(status=0,role=\text{"}receiver\text{"})!(2)^3} \{\!| \langle \gamma'_R, \mathsf{LS}'_R \rangle : [self.product := 2, self.status := 1]p_2 |\!\}$ *by* `Lor`, `Ror` *and* `Rec`

For the system level we introduce one more transition label, the unicast transition τ. For the possible transition labels we then have:

$$\alpha ::= \quad \lambda \quad | \quad \tau$$

We can then define the following system level semantics:

$$\texttt{Sys}\ \frac{\langle \gamma, \mathsf{LS} \rangle : P \overset{\lambda}{\mapsto} \langle \gamma', \mathsf{LS}' \rangle : P'}{\langle \gamma, \mathsf{LS} \rangle : P \xrightarrow{\lambda} \langle \gamma', \mathsf{LS}' \rangle : P'} \qquad \texttt{Nsys}\ \frac{\langle \gamma, \mathsf{LS} \rangle : P \xmapsto{\widetilde{\Pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P}{\langle \gamma, \mathsf{LS} \rangle : P \xrightarrow{\Pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS} \rangle : P}$$

$$\texttt{Prec}\ \frac{S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S'_1 \quad S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S'_2}{S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S'_1 \mid\mid S'_2}$$

$$\texttt{Lpar}\ \frac{S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S'_1 \quad S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S'_2}{S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S'_1 \mid\mid S'_2} \qquad \texttt{Rpar}\ \frac{S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S'_1 \quad S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S'_2}{S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S'_1 \mid\mid S'_2}$$

$$\texttt{Lsup}\ \frac{S_1 \xrightarrow{\Pi\mathsf{S}(\tilde{v})^{ch}} S'_1}{S_1 \mid\mid S_2 \xrightarrow{\Pi\mathsf{S}(\tilde{v})^{ch}} S'_1 \mid\mid S_2} \qquad \texttt{Rsup}\ \frac{S_2 \xrightarrow{\Pi\mathsf{S}(\tilde{v})^{ch}} S'_2}{S_1 \mid\mid S_2 \xrightarrow{\Pi\mathsf{S}(\tilde{v})^{ch}} S_1 \mid\mid S'_2}$$

$$\texttt{Lget}\ \frac{S_1 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S'_1}{S_1 \mid\mid S_2 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S'_1 \mid\mid S_2} \qquad \texttt{Rget}\ \frac{S_2 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S'_2}{S_1 \mid\mid S_2 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S_1 \mid\mid S'_2}$$

$$\texttt{Luni}\ \frac{S_1 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S'_1 \quad S_2 \xrightarrow{\Pi\mathsf{S}(\tilde{v})^{ch}} S'_2}{S_1 \mid\mid S_2 \xrightarrow{\tau} S'_1 \mid\mid S'_2} \qquad \texttt{Runi}\ \frac{S_1 \xrightarrow{\Pi\mathsf{S}(\tilde{v})^{ch}} S'_1 \quad S_2 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S'_2}{S_1 \mid\mid S_2 \xrightarrow{\tau} S'_1 \mid\mid S'_2}$$

$$\texttt{Ltau}\ \frac{S_1 \xrightarrow{\tau} S_1}{S_1 \mid\mid S_2 \xrightarrow{\tau} S'_1 \mid\mid S_2} \qquad \texttt{Rtau}\ \frac{S_2 \xrightarrow{\tau} S'_2}{S_1 \mid\mid S_2 \xrightarrow{\tau} S_1 \mid\mid S'_2}$$

Table 3.4: The system level semantics.

We can see in 3.4 that on a system level, denoted by $\rightarrow$, we cannot see the discard transition. In addition we get a new transition τ, this is also called the hidden transition and it occurs when unicast communication is performed.

Example 3.2.2 *[Product fetching (step 5/5)] We can now look at the system level transitions from our example.*
$C \mid\mid I \mid\mid O \mid\mid R \xrightarrow{\tau} U_C \mid\mid I' \mid\mid O \mid\mid R$
$U_C \mid\mid I' \mid\mid O \mid\mid R \xrightarrow{(role=\text{"manager"})!(\text{"start"},2)^*} 0 \mid\mid I' \mid\mid U_O \mid\mid R$
$U_C \mid\mid I' \mid\mid O \mid\mid R \xrightarrow{(status=0,role=\text{"receiver"})!(2)^3} 0 \mid\mid I' \mid\mid O \mid\mid p_2$

As we can see this is a lot clearer of an overview. But we do not know what C and I communicated, nor do we know how the environments updated.

3.3 Lemmas

We will first state some essential lemmas, to prove certain behaviorial properties of our system. The proofs of these lemmas can be found in the appendix.

Lemma 3.3.1
1. $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$
2. $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$
3. $S_1 \mid\mid \Gamma : 0 \equiv S_1$

Given 2 systems S_1 and S_2 proof:

Lemma 3.3.2 *Lemma 2 is regarding the non-blocking behavior of broadcasted messages.*
$ch = * \wedge S_1 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \implies S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ *for any* S_2.

Lemma 3.3.3 *Lemma 3 shows the blocking of multi cast channels.*
$ch \neq * \wedge S_1 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \implies S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ *for any* S_2 *such that:*

1. $ch \in \mathsf{LS} \wedge S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$
   $\vee$
2. $ch \notin \mathsf{LS}$

Lemma 3.3.4 *This lemma is about the blocking of unicast, to show it should never go unanswered*
$S_1 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S_1' \implies S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$ *for any* S_2 *such that* $S_2 \xrightarrow{\Pi\mathsf{S}(\tilde{v})^{ch}} S_2'$.

3.4 Context

Now that we have the basis of our model, we can add the reasoning about physical space. For this we will define an other layer to our syntax, the Context layer.

Context	$C ::=$	$[S]^{\gamma,\mathsf{LS}}$	$\mid$	$C_1 \mid\mid C_2$

Table 3.5: The syntax of our context.

A context is a subsection of the physical space in which our system exists. To clarify some of the definition we will use an example space as shown in 3.1. This is a simplified layout of a physical space, we will go further into how we can divide and reason about the context for spaces.

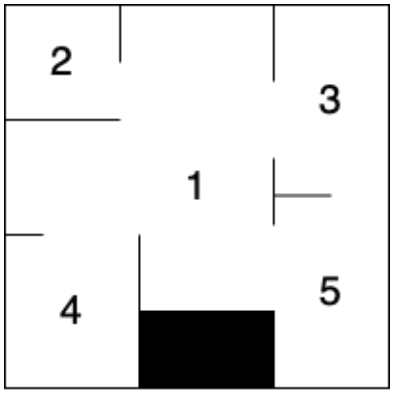

Figure 3.1: Example of space layout

In Table 3.5 we first have the singular context $[S]^{\gamma,\mathsf{LS}}$. Here the context would be one of the numbered rooms and S would exist in that room. Note that S can be an empty process, in other words, a context does not require a system to be present to exist. We can also see that each context has an environment, just like the systems. γ stores the local variables and the physical space representation and LS stores the local channels. In the semantics we will see how this environment influences the systems present in the context. A context can also be a parallel composition of contexts, these parallel compositions allow for representing a full physical space, without the need for specified relations. For non-continuous, or discrete, space, this can be helpful as we do not have to keep track of the relations between contexts as their relation will no change, but they still exist in parallel. As our example has no defined continuous spaces, here the spaces would exist in parallel.
The other two contexts define the relations between contexts. For this we have two different options, namely the externally connected relation $\mathsf{EC}(C_1,C_2)^{\gamma,\mathsf{LS}}$, this relation argues that systems from 1 context could move to the other context and depending on the specified model, could also communicate between the contexts. Relating it to 3.1, we would have $\mathsf{EC}(1,2)$ for example, as space 1 is directly connected to space 2. The environment of this context is a union of the environment of C_1 and C_2. The opposite of externally connected is disconnected $\mathsf{DC}(C_1,C_2)^{\gamma,\mathsf{LS}}$. As we do not allow for movements to non-externally connected contexts, disconnected contexts limit the possible actions which can be taken. The environment is again a combination of both contexts, and for continuous contexts could be used for moving closer to eventually allow for the exchanging of processes. In relation to 3.1, this would be $\mathsf{DC}(2,4)$. We would most likely not define this relation for discrete physical spaces like these rooms, as just saying they exist in parallel also argues that they cannot directly exchange processes.
For our context semantics, to distinguish them from the other transition levels, we will use $\Rightarrow$ for context level transitions. We will use the transition label $\mu(S,E)$ to model movements, where S is the system which moves and E an expression which can be concretized to a destination context.

$$\texttt{Out}\ \frac{}{[\langle \gamma_S, \mathsf{LS}_S\rangle : S]^{\gamma_C,\mathsf{LS}_C} \xrightarrow{\mu(S,C')} [\quad]^{\gamma'_C,\mathsf{LS}_C}}$$

$$\texttt{In}\ \frac{}{[\quad]^{\gamma_C,\mathsf{LS}_C} \xrightarrow{\mu(S,C)} [\langle \gamma_S \cup \gamma_C, \mathsf{LS}_S \cup \mathsf{LS}_C\rangle : S']^{\gamma'_C,\mathsf{LS}_C}}$$

$$\texttt{LPOut}\ \frac{}{[\langle \gamma_{S_1}, \mathsf{LS}_{S_1}\rangle : S_1 \mid\mid \langle \gamma_{S_2}, \mathsf{LS}_{S_2}\rangle : S_2]^{\gamma_C,\mathsf{LS}_C} \xrightarrow{\mu(S_1,E)} [\langle \gamma_{S_2}, \mathsf{LS}_{S_2}\rangle : S_2]^{\gamma'_C,\mathsf{LS}_C}}$$

$$\texttt{RPOut}\ \frac{}{[\langle \gamma_{S_1}, \mathsf{LS}_{S_1}\rangle : S_1 \mid\mid \langle \gamma_{S_2}, \mathsf{LS}_{S_2}\rangle : S_2]^{\gamma_C,\mathsf{LS}_C} \xrightarrow{\mu(S_2,E)} [\langle \gamma_{S_1}, \mathsf{LS}_{S_1}\rangle : S_1]^{\gamma'_C,\mathsf{LS}_C}}$$

$$\texttt{RPIn}\ \frac{}{[\langle \gamma_{S_1}, \mathsf{LS}_{S_1}\rangle : S_1]^{\gamma_C,\mathsf{LS}_C} \xrightarrow{\mu(S_2,C)} [\langle \gamma_{S_1}, \mathsf{LS}_{S_1}\rangle : S_1 \mid\mid \langle \gamma_{S_2} \cup \gamma_C, \mathsf{LS}_{S_2} \cup \mathsf{LS}_C\rangle : S']^{\gamma'_C,\mathsf{LS}_C}}$$

$$\texttt{LMov}\ \frac{EC(C_1,C_2)}{[\langle \gamma_S, \mathsf{LS}_S\rangle : S]_1^{\gamma_{C_1},\mathsf{LS}_{C_1}} \mid\mid [\quad]_2^{\gamma_{C_2},\mathsf{LS}_{C_2}} \xrightarrow{\mu(S,C_2)} [\quad]_1^{\gamma_{C_1},\mathsf{LS}_{C_1}} \mid\mid [\langle (\gamma_S \setminus \gamma_{C_1}) \cup \gamma_{C_2}, (\mathsf{LS}_S \setminus \mathsf{LS}_{C_1}) \cup \mathsf{LS}_{C_2}\rangle : S']_2^{\gamma_{C_2},\mathsf{LS}_{C_2}}}$$

$$\texttt{RMov}\ \frac{EC(C_1,C_2)}{[\quad]_1^{\gamma_{C_1},\mathsf{LS}_{C_1}} \mid\mid [\langle \gamma_S, \mathsf{LS}_S\rangle : S]_2^{\gamma_{C_2},\mathsf{LS}_{C_2}} \xrightarrow{\mu(S,C_1)} [\langle (\gamma_S \setminus \gamma_{C_2}) \cup \gamma_{C_1}, (\mathsf{LS}_S \setminus \mathsf{LS}_{C_2}) \cup \mathsf{LS}_{C_1}\rangle : S']_1^{\gamma_{C_1},\mathsf{LS}_{C_1}} \mid\mid [\quad]_2^{\gamma_{C_2},\mathsf{LS}_{C_2}}}$$

Table 3.6: The system level semantics.

4

Conclusion

You may consider to instead divide this chapter into discussion of the results and a summary.

4.1 Discussion

4.2 Conclusion

Bibliography

[1] G. Brajnik, "A comparative test of web accessibility evaluation methods," in *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility - Assets '08*, New York, New York, USA: ACM Press, 2008, p. 113, ISBN: 9781595939760. DOI: 10.1145/1414471.1414494. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1414471.1414494.

[2] Y. Abd Alrahman, S. Azzopardi, and N. Piterman, "Model checking reconfigurable interacting systems," in *International Symposium on Leveraging Applications of Formal Methods*, Springer, 2022, pp. 373–389.

[3] V. Vittikh and P. Skobelev, "Multi-agent systems for modelling of self-organization and cooperation processes," *WIT Transactions on Information and Communication Technologies*, vol. 20, 1970.

[4] J. Ferber and G. Weiss, *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison-wesley Reading, 1999, vol. 1.

[5] M. J. Beeson, "Proving programs and programming proofs," vol. 114, pp. 51–82, 1986.

[6] W. Van der Hoek and M. Wooldridge, "Multi-agent systems," *Foundations of Artificial Intelligence*, vol. 3, pp. 887–928, 2008.

[7] J. Xie and C.-C. Liu, "Multi-agent systems and their applications," *Journal of International Council on Electrical Engineering*, vol. 7, no. 1, pp. 188–197, 2017.

[8] *Transport Styrelsen automated vehicles*, Accessed: 2024-04-03. [Online]. Available: https://www.transportstyrelsen.se/en/road/Vehicles/self-driving-vehicles/#:~:text=The%20Swedish%20Transport%20Agency%20issues,in%20a%20traffic%2Dsafe%20manner.

[9] A. Ferscha, "Collective adaptive systems," pp. 893–895, 2015.

[10] P. Busetta, A. Dona, and M. Nori, "Channeled multicast for group communications," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, 2002, pp. 1280–1287.

[11] R. Milner, *A calculus of communicating systems*. Springer, 1980.

[12] R. Milner, *The polyadic π-calculus: a tutorial*. Springer, 1993.

[13] Y. Abd Alrahman and N. Piterman, "Modelling and verification of reconfigurable multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 35, no. 2, p. 47, 2021.

[14] Y. A. Alrahman, S. Azzopardi, L. Di Stefano, and N. Piterman, "Language support for verifying reconfigurable interacting systems," *International Journal on Software Tools for Technology Transfer*, pp. 1–20, 2023.

[15] C. D. Tran, H. Ezzedine, and C. Kolski, "Eiseval, a generic reconfigurable environment for evaluating agent-based interactive systems," *International Journal of Human-Computer Studies*, vol. 71, no. 6, pp. 725–761, 2013.

[16] J. Bay, "Fundamentals of linear state space systems," 1999.

[17] X. Yin, Z. Gao, D. Yue, and Y. Fu, "Convergence of velocities for the short range communicated discrete-time cucker–smale model," *Automatica*, vol. 129, p. 109 659, 2021.

[18] T. Kosar and M. Livny, "Stork: Making data placement a first class citizen in the grid," in *24th International Conference on Distributed Computing Systems, 2004. Proceedings.*, IEEE, 2004, pp. 342–349.

[19] S. Bandini, S. Manzoni, and G. Vizzari, "Multi-agent approach to localization problems: The case of multilayered multi-agent situated system," *Web Intelligence and Agent Systems: An International Journal*, vol. 2, no. 3, pp. 155–166, 2004.

[20] C. Koomen and C. Koomen, "Calculus of communicating systems," *The Design of Communicating Systems: A System Engineering Approach*, pp. 11–26, 1991.

[21] D. A. Randell, Z. Cui, and A. G. Cohn, "A spatial logic based on regions and connection.," *KR*, vol. 92, pp. 165–176, 1992.

A

Appendix 1

A.1 Lemma 1

Lemma A.1.1 *1.* $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$

2. $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$

3. $S_1 \mid\mid \Gamma : 0 \equiv S_1$

We will proof this lemma by case analysis on α. We will have 5 cases with multiple subcases each. To proof a case we need to find a matching transition on both sides which are possible by the semantic rules. Only when all cases and subcases have been covered have we proven the statement.
1. The first point is proving commutativity. We can see in the syntax that the cases we have to cover are: $\Pi!(\tilde{v})^{ch}$, $\Pi?(\tilde{v})^{ch}$, $\Pi\mathsf{G}(\tilde{v})^{ch}$, $\Pi\mathsf{S}(\tilde{v})^{ch}$ and τ
Case 1: Consider $\alpha = \Pi!(\tilde{v})^{ch}$.
Subcase 1.1: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2'$. Assume $S_1 \neq S_1'$ and $S_2 \neq S_2'$, as those cases are covered by subcases 1.2, 1.3 and 1.4. This transition then has two subcases of its own.
Subcase 1.1.1: Consider `Lpar`, we have $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. Giving us $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$. By `Rpar` on $S_2 \mid\mid S_1$, we also get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$, resulting in $S_2 \mid\mid S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2' \mid\mid S_1'$. Showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$.
Subcase 1.1.2: Consider `Rpar`, we have $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$. Giving us $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$. By `Lpar` on $S_2 \mid\mid S_1$, we also get $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$ and $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$, resulting in $S_2 \mid\mid S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2' \mid\mid S_1'$. Showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$.
Subcase 1.2: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2$. In this case only S_1 transitions. Looking at the semantics, it does not immediately seem like such a transition exists. However, by `Nsys` we can discard the $S \xrightarrow{\Pi?(\tilde{v})^{ch}} S'$ transition making it $\langle \gamma, \mathsf{LS} \rangle : P \xmapsto{\widetilde{\Pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P$.
Subcase 1.2.1: Consider `Lpar`, we get $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$, but for this case by `Nsys`, we get $S_2 = \langle \gamma, \mathsf{LS} \rangle : P$ and $\langle \gamma, \mathsf{LS} \rangle : P \xmapsto{\widetilde{\Pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P$. This results in $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2$. By `Rpar` and `Nsys`, we then get $\langle \gamma, \mathsf{LS} \rangle : P \xmapsto{\widetilde{\Pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P$ for $S_2 = \langle \gamma, \mathsf{LS} \rangle : P$ and $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ as well. This results in $S_2 \mid\mid S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2 \mid\mid S_1'$. Showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$.
Subcase 1.2.2: Consider `Rpar`, we would get S_2 with the sending transition, but by our IH, we have that S_2 does not transition. By `Nsys`, we see that it is only possible for the receiving, and thus this is not possible.

Subcase 1.3: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2'$. In this case we have the opposite of the previous case.
Subcase 1.3.1: Consider `Lpar`, just like in case 1.2.2, this is not possible, because S_1 would be the sending system, but we know S_1 does not transition.
Subcase 1.3.2: Consider `Rpar`, we would get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$, but for this case by `Nsys`, we get $S_1 = \langle \gamma, \mathsf{LS} \rangle : P$ and $\langle \gamma, \mathsf{LS} \rangle : P \xmapsto{\widetilde{\Pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P$. This results in $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1 \mid\mid S_2'$. By `Lpar` and `Nsys`, we then also get $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$ and $\langle \gamma, \mathsf{LS} \rangle : P \xmapsto{\widetilde{\Pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P$ with $S_1 = \langle \gamma, \mathsf{LS} \rangle : P$ for $S_2 \mid\mid S_1$. This results in $S_2 \mid\mid S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2 \mid\mid S_1'$. Showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$.
Subcase 1.4: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2$, this case is a combination of subcases 1.2.2 and 1.3.1. We have explained why for a send transition, at least one system has to transition. This means this transition is not possible.

Case 2: Consider $\alpha = \Pi?(\tilde{v})^{ch}$.
Subcase 2.1: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2'$. Assume $S_1 \neq S_1'$ and $S_2 \neq S_2'$, as those cases are covered by subcases 1.2, 1.3 and 1.4. Consider `Prec`, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Prec`, we also get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ for $S_2 \mid\mid S_1$, showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$.
Subcase 2.2: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2$. In this case only S_1 transitions. By `Nsys` we can discard the $S \xrightarrow{\Pi?(\tilde{v})^{ch}} S'$ transition making it $\langle \gamma, \mathsf{LS} \rangle : P \xmapsto{\widetilde{\Pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P$. Consider `Prec`, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$, by `Nsys` on $S_2 = \langle \gamma, \mathsf{LS} \rangle : P$, we get $\langle \gamma, \mathsf{LS} \rangle : P \xmapsto{\widetilde{\Pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P$. This results in $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2$. By `Prec` and `Nsys` on S_2, we also get $S_2 \mid\mid S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2 \mid\mid S_1'$ for $S_2 \mid\mid S_1$, showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$.
Subcase 2.3: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2'$. In this case we have the opposite of the previous case. Consider `Prec`, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$, by `Nsys` on $S_1 = \langle \gamma, \mathsf{LS} \rangle : P$, we get $\langle \gamma, \mathsf{LS} \rangle : P \xmapsto{\widetilde{\Pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P$. This results in $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1 \mid\mid S_2'$. By `Prec` and `Nsys` on S_1, we also get $S_2 \mid\mid S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2' \mid\mid S_1$ for $S_2 \mid\mid S_1$, showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$.
Subcase 2.4: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2$. This is a combination of subcase 2.2 and 2.3. Consider `Prec`, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Nsys` on $S_1 = \langle \gamma, \mathsf{LS} \rangle : P_1$ and $S_2 = \langle \gamma, \mathsf{LS} \rangle : P_2$, we get $\langle \gamma, \mathsf{LS} \rangle : P_1 \xmapsto{\widetilde{\Pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P_1$ and $\langle \gamma, \mathsf{LS} \rangle : P_2 \xmapsto{\widetilde{\Pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P_2$. This gives us $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1 \mid\mid S_2$. By `Prec` and `Nsys` on both S_2 and S_1, we also get $S_2 \mid\mid S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2' \mid\mid S_1$ for $S_2 \mid\mid S_1$, showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$.

Case 3: Consider $\alpha = \Pi\mathsf{G}(\tilde{v})^{ch}$.
Subcase 3.1: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2'$. Assume $S_1 \neq S_1'$ and $S_2 \neq S_2'$, as those cases are covered by subcases 1.2, 1.3 and 1.4. The rules which have $\Pi\mathsf{G}(\tilde{v})^{ch}$ exposed are `Lget` and `Rget`. Each of these only transition one of the systems, while the other stays in the same state. This means it is not possible for $S_1 \neq S_1'$ and $S_2 \neq S_2'$. This makes sense, as only a single process can ask to get information at a time, just like only 1 send transition can be send at a time. If the second system also transitions, this would have to be by supplying information, however this would result in a hidden, τ, transition. Therefore this case is not possible.
Subcase 3.2: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2$. In this case only S_1 transitions.
Subcase 3.2.1: Consider `Lget`, we get $S_1 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S_1'$, resulting in $S_1 \mid\mid S_2 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S_1' \mid\mid S_2$. By `Rget` on

$S_2 \mid\mid S_1$, we also get $S_1 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_1'$, resulting in $S_2 \mid\mid S_1 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_2 \mid\mid S_1'$, showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$ in this case.
Subcase 3.2.2: Consider `Rget`, this would require S_2 to transition, but we know it does not. So this subcase is not possible.
Subcase 3.3: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2'$. In this case we have the opposite of the previous case.
Subcase 3.2.1: Consider `Lget`, this would require S_2 to transition, but we know it does not. So this subcase is not possible.
Subcase 3.2.2: Consider `Rget`, we get $S_2 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_2'$, resulting in $S_1 \mid\mid S_2 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_1 \mid\mid S_2'$. By `Lget` on $S_2 \mid\mid S_1$, we also get $S_2 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_2'$, resulting in $S_2 \mid\mid S_1 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_2' \mid\mid S_1$, showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$ in this case.
Subcase 3.4: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2$. Consider `Nsys`, is states the only transition which can be discarded is the send transition. As that is not our α, this case is not possible.

Case 4: Consider $\alpha = \Pi S(\tilde{v})^{ch}$.
Subcase 4.1: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2'$. Assume $S_1 \neq S_1'$ and $S_2 \neq S_2'$, as those cases are covered by subcases 1.2, 1.3 and 1.4. Just like in subcase 3.1, this case is not possible. Only 1 system can supply at a time, meaning we cannot have two systems transition with an exposed supply transition.
Subcase 4.2: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2$. In this case only S_1 transitions.
Subcase 4.2.1: Consider `Lsup`, we get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$, resulting in $S_1 \mid\mid S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1' \mid\mid S_2$. By `Rsup` on $S_2 \mid\mid S_1$, we also get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$, resulting in $S_2 \mid\mid S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2 \mid\mid S_1'$, showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$ in this case.
Subcase 4.2.2: Consider `Rsup`, this would require S_2 to transition, but we know it does not. So this subcase is not possible.
Subcase 4.3: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2'$. In this case we have the opposite of the previous case.
Subcase 4.3.1: Consider `Lsup`, this would require S_2 to transition, but we know it does not. So this subcase is not possible.
Subcase 4.3.2: Consider `Rsup`, we get $S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2'$, resulting in $S_1 \mid\mid S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1 \mid\mid S_2'$. By `Lsup` on $S_2 \mid\mid S_1$, we also get $S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2'$, resulting in $S_2 \mid\mid S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2' \mid\mid S_1$, showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$ in this case.
Subcase 4.4: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2$. Consider `Nsys`, is states the only transition which can be discarded is the send transition. As that is not our α, this case is not possible.

Case 5: Consider $\alpha = \tau$.
Subcase 5.1: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2'$. Assume $S_1 \neq S_1'$ and $S_2 \neq S_2'$, as those cases are covered by subcases 1.2, 1.3 and 1.4. This leaves us with 2 subcases, as `Ltau` and `Rtau` only have 1 system transitioning.
Subcase 5.1.1: Consider `Luni`, we get $S_1 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2'$ resulting in $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$. By `Runi`, we also get $S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2'$ and $S_1 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_1'$ for $S_2 \mid\mid S_1$. Resulting in $S_2 \mid\mid S_1 \xrightarrow{\tau} S_2' \mid\mid S_1'$. Therefore $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$ holds for this case.
Subcase 5.1.2: Consider `Runi`, we get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_2'$ resulting in $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$. By `Luni`, we also get $S_2 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_2'$ and $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$ for $S_2 \mid\mid S_1$. Resulting in $S_2 \mid\mid S_1 \xrightarrow{\tau} S_2' \mid\mid S_1'$. Therefore $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$ holds for this case.

Subcase 5.2: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2$. In this case only S_1 transitions with τ. Three of the τ-transition rules do not allow this, because by `Nsys` we cannot discard the transition. The rules `Luni`, `Runi` and `Rtau` require S_2 to transition, which for our subcase is not the case. This leaves us with one more case. Consider `Ltau`, we get $S_1 \xrightarrow{\tau} S_1'$ resulting in $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2$. By `Rtau`, also $S_2 \mid\mid S_1$ gives us $S_1 \xrightarrow{\tau} S_1'$ and $S_2 \mid\mid S_1 \xrightarrow{\tau} S_2 \mid\mid S_1'$. Meaning $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$ holds for this case.

Subcase 5.3: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2'$. In this case we have the opposite of the previous case. Once again we have 3 transitions which are not possible, namely `Luni`, `Runi` and `Ltau`. Consider `Rtau`, we get $S_2 \xrightarrow{\tau} S_2'$ resulting in $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1 \mid\mid S_2'$. By `Ltau`, also $S_2 \mid\mid S_1$ gives us $S_2 \xrightarrow{\tau} S_2'$ and $S_2 \mid\mid S_1 \xrightarrow{\tau} S_2' \mid\mid S_1$. Meaning $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$ holds for this case.

Subcase 5.4: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2$. This case is not possible, for a τ, transition to happen, we need at least 1 system to make that transition as by `Nsys`, this is not possible to discard.

As it holds for all cases, we have proven commutativity. □

2. We have shown commutativity and the next point is associativity, $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$. We will once again perform a case analysis to prove this. Because our rules only deal with parallel compositions of 2 systems, we will consider the two systems between parentheses as 1. By our grammer, we can then consider the possible other rules and match those.

Case 1: Consider $\alpha = \Pi!(\tilde{v})^{ch}$.

Subcase 1.1: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3'$. Assume $S_i \neq S_i'$ as that is covered by other subcases. As we have a sending transition, we will end up with 3 subcases, namely of each system being the sending system, because only 1 system can send at a time.

Subcase 1.1.1: Consider `Lpar`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$. We can now split the first transition into two subcases, one where S_1 is sending and one where S_2 is sending.

Subcase 1.1.1.1: Consider `Lpar` on $S_1 \mid\mid S_2$. We get $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$, together with $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, we then have $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} (S_1' \mid\mid S_2') \mid\mid S_3'$. By `Lpar` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we get $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. By `Prec`, we then get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.1.1.2: Now consider `Rpar` on $S_1 \mid\mid S_2$. We get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$, together with $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, we then have $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} (S_1' \mid\mid S_2') \mid\mid S_3'$. By `Rpar` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. By `Lpar`, we then get $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.1.2: Consider `Rpar`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_3'$. By `Prec` on $S_1 \mid\mid S_2$, we then get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Rpar` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. By `Rpar` on $S_2 \mid\mid S_3$, we then get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_3'$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.2: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3$. Here only S_1 transitions. From the commutativity proof, we know that for our α, the sending transition, the sending system has to transition. This means our sending transition has to be S_1.

Subcase 1.2.1: Consider `Lpar`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$.

Subcase 1.2.1.1: Consider `Lpar` on $S_1 \mid\mid S_2$. We get $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Nsys` on

$S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, we get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$. We then have $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}}$ $(S_1' \mid\mid S_2) \mid\mid S_3$. By `Lpar` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we get $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. By `Prec` and `Nsys` on both transitions, we then get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.2.1.1: Consider `Rpar` on $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$. This would require S_2 to be the sending system, however we know S_2 does not transition and thus this is not possible.

Subcase 1.2.2: Consider `Rpar`. This would require S_3 to be the sending system, however we know S_3 does not transition and thus this is not possible.

Subcase 1.3: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3$. This case is similar to the previous subcase, but this time we have S_2 which has to be the sending transition.

Subcase 1.3.1: Consider `Lpar`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$.

Subcase 1.3.1.1: Consider `Lpar` on $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$. This would require S_1 to be the sending system, however we know S_1 does not transition and thus this is not possible.

Subcase 1.3.1.2: Consider `Rpar` on $S_1 \mid\mid S_2$. We get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$. By `Nsys` on $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$. We then have $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}}$ $(S_1 \mid\mid S_2') \mid\mid S_3$. By `Rpar` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. By `Lpar` and `Nsys` on S_3, we then get $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.3.2: Consider `Rpar`. This would require S_3 to be the sending system, however we know S_3 does not transition and thus this is not possible.

Subcase 1.4: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3'$. This is once again a similar case, but this time with S_3 being the sending transition.

Subcase 1.4.1: Consider `Lpar`. This would require S_1 or S_2 to be the sending system, however we know neither transitions and thus this is not possible.

Subcase 1.4.2: Consider `Rpar`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_3'$. Now consider `Prec` on $S_1 \mid\mid S_2$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Nsys` on $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$. Resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} (S_1 \mid\mid S_2) \mid\mid S_3'$. By `Rpar` on $S_1 \mid\mid (S_2 \mid\mid S_3)$ we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. Then by `Rpar` again on the latter transition we get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_3'$. Resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} (S_1 \mid\mid S_2) \mid\mid S_3'$ with the same transitions. Showing $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.5: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3$. In this case S_1 and S_2 transition, meaning either of them can be the sending system.

Subcase 1.5.1: Consider `Lpar`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$.

Subcase 1.5.1.1: Consider `Lpar` on $S_1 \mid\mid S_2$. We get $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Nsys` on $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, we get $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$, we then have $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} (S_1' \mid\mid S_2') \mid\mid S_3$. By `Lpar` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we get $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. By `Prec`, we then get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and

$S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$ and by Nsys on the latter we get $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.5.1.2: Now consider Rpar on $S_1 \mid\mid S_2$. We get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$. By Nsys on $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, we get $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$, we then have $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} (S_1' \mid\mid S_2') \mid\mid S_3$. By Rpar on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. By Lpar, we then get $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, and by Nsys we have $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.5.2: Consider Rpar, this would require S_3 to be the sending system, but as it does not transition this is not possible.

Subcase 1.6: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3'$. This is similar to the previous case, but now with S_1 or S_3 being the sending transition.

Subcase 1.6.1: Consider Lpar, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$.

Subcase 1.6.1.1: Consider Lpar on $S_1 \mid\mid S_2$. We get $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By Nsys on $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$, we get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$. Together with $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$ we then have $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} (S_1' \mid\mid S_2) \mid\mid S_3'$. By Lpar on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we get $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. By Prec, we then get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$ and by Nsys on the latter we get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.6.1.2: Consider Rpar on $S_1 \mid\mid S_2$. This would require S_2 to be the sending transition but as it does not transition this is not possible.

Subcase 1.6.2: Consider Rpar, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_3'$. By Prec on $S_1 \mid\mid S_2$, we then get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$, combined with Nsys on the latter we get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$. Resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} (S_1' \mid\mid S_2) \mid\mid S_3'$. By Rpar on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. By Rpar on $S_2 \mid\mid S_3$, we then get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_3'$. By Nsys on $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ we get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.7: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3'$. Once again this is a similar case to the previous subcases but with S_2 or S_3 being the sending system.

Subcase 1.7.1: Consider Lpar, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$.

Subcase 1.7.1.1: Consider Lpar on $S_1 \mid\mid S_2$. This would require S_1 to be the sending transition but as it does not transition this is not possible.

Subcase 1.7.1.2: Consider Rpar on $S_1 \mid\mid S_2$. We get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$. By Nsys on $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$. Together with $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$ we then have $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} (S_1 \mid\mid S_2') \mid\mid S_3'$. By Rpar on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. By Rpar again, we then get $S_2 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$ and by Nsys on $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$. These are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.7.2: Consider Rpar, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_3'$. By Prec on $S_1 \mid\mid S_2$,

we then get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$, combined with `Nsys` on the first we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$. Resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} (S_1 \mid\mid S_2') \mid\mid S_3'$. By `Rpar` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \mid\mid S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_2' \mid\mid S_3'$. By `Rpar` on $S_2 \mid\mid S_3$, we then get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_3'$. By `Nsys` on $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 1.8: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3$. As none of the systems transition, this case is not possible as for a sending action, a system has to transition.

Case 2: Consider $\alpha = \Pi?(\tilde{v})^{ch}$.

Subcase 2.1: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3'$. Consider `Prec`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$. By `Prec` again on $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. This gives us $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} (S_1' \mid\mid S_2') \mid\mid S_3'$. By `Prec` twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$, $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 2.2: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3$. Consider `Prec`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$. By `Prec` again on $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Nsys` on S_2 and S_3 we get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$. This gives us $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} (S_1' \mid\mid S_2) \mid\mid S_3$. By `Prec` twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$ and `Nsys` on the resulting S_2 and S_3 transitions, we also get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$, $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 2.3: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3$. Consider `Prec`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$. By `Prec` again on $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Nsys` on S_1 and S_3 we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$. This gives us $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} (S_1 \mid\mid S_2') \mid\mid S_3$. By `Prec` twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$ and `Nsys` on the resulting S_1 and S_3 transitions, we also get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$, $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 2.4: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3'$. Consider `Prec`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$. By `Prec` again on $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Nsys` on S_1 and S_2 we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$. This gives us $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} (S_1 \mid\mid S_2) \mid\mid S_3'$. By `Prec` twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$ and `Nsys` on the resulting S_1 and S_2 transitions, we also get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$, $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 2.5: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3$. Consider `Prec`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$. By `Prec` again on $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Nsys` on

S_3 we get $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$. This gives us $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} (S_1' \mid\mid S_2') \mid\mid S_3$. By `Prec` twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$ and `Nsys` on the resulting S_3 transition, we also get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$, $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 2.6: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3'$. Consider `Prec`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$. By `Prec` again on $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Nsys` on S_2 we get $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$. This gives us $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} (S_1' \mid\mid S_2) \mid\mid S_3'$. By `Prec` twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$ and `Nsys` on the resulting S_2 transition, we also get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$, $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 2.7: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3'$. Consider `Prec`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$. By `Prec` again on $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Nsys` on S_1 we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$. This gives us $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} (S_1 \mid\mid S_2') \mid\mid S_3'$. By `Prec` twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$ and `Nsys` on the resulting S_1 transition, we also get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$, $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 2.8: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3$. Consider `Prec`, we get $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3'$. By `Prec` again on $S_1 \mid\mid S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2'$. By `Nsys` on all three transitions we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$, $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$. This gives us $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} (S_1 \mid\mid S_2) \mid\mid S_3$. By `Prec` twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$ and `Nsys` on the resulting transitions, we also get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$, $S_2 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_2$ and $S_3 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_3$, which are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Case 3: Consider $\alpha = \Pi\mathsf{G}(\tilde{v})^{ch}$.

Subcase 3.1: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3'$. We know only one system can send a get request at a time. We also know that if a system supplies, the transition is hidden, this means for this α only 1 system can transition and thus this subcase is not possible.

Subcase 3.2: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3$. As we know a system has to transition to send a get request, we know S_1 has to be the getting system. The only way to get this is by applying `Lget` twice. We get $S_1 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S_1'$. By `Lget` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_1 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S_1'$. which is the same transition and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 3.3: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3$. As we know a system has to transition to send a get request, we know S_2 has to be the getting system. The only way to get this is by applying `Lget` and `Rget`. We get $S_2 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S_2'$. By `Rget` and `Lget` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_2 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S_2'$. which is the same transition and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 3.4: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3'$. As we know a system has to transition to send a get request, we know S_3 has to be the getting system. The only way to get this is by applying `Rget`. We get $S_3 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S_3'$. By `Rget` twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_3 \xrightarrow{\Pi\mathsf{G}(\tilde{v})^{ch}} S_3'$. which is the same transition and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 3.5: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3$. We know only one system can send a get request at a time.

We also know that if a system supplies, the transition is hidden, this means for this α only 1 system can transition and thus this subcase is not possible.
Subcase 3.6: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3'$. We know only one system can send a get request at a time. We also know that if a system supplies, the transition is hidden, this means for this α only 1 system can transition and thus this subcase is not possible.
Subcase 3.7: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3'$. We know only one system can send a get request at a time. We also know that if a system supplies, the transition is hidden, this means for this α only 1 system can transition and thus this subcase is not possible.
Subcase 3.8: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3$. We know by `Nsys`, that only a receive transition can be discarded. This means for our get transition, at least 1 system has to transition and thus this subcase is not possible.

Case 4: Consider $\alpha = \Pi S(\tilde{v})^{ch}$.
Subcase 4.1: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3'$. This case has similar reasoning to the previous case. We know only one system can supply at a time. We also know that if a system in the same parallel composition has sent a get request, the transition is hidden, this means for this α only 1 system can transition and thus this subcase is not possible.
Subcase 4.2: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3$. As we know a system has to transition to supply, we know S_1 has to be the supplying system. The only way to get this is by applying `Lsup` twice. We get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$. By `Lsup` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$. This is the same transition and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.
Subcase 4.3: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3$. As we know a system has to transition to supply, we know S_2 has to be the supplying system. The only way to get this is by applying `Lsup` and `Rsup`. We get $S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2'$. By `Rsup` and `Lsup` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2'$. This is the same transition and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.
Subcase 4.4: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3'$. As we know a system has to transition to supply, we know S_3 has to be the supplying system. The only way to get this is by applying `Rsup`. We get $S_3 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_3'$. By `Rsup` twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_3 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_3'$. This is the same transition and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.
Subcase 4.5: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3$. We know only one system can supply at a time. We also know that if a system in the same parallel composition has sent a get request, the transition is hidden, this means for this α only 1 system can transition and thus this subcase is not possible.
Subcase 4.6: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3'$. We know only one system can supply at a time. We also know that if a system in the same parallel composition has sent a get request, the transition is hidden, this means for this α only 1 system can transition and thus this subcase is not possible.
Subcase 4.7: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3'$. We know only one system can supply at a time. We also know that if a system in the same parallel composition has sent a get request, the transition is hidden, this means for this α only 1 system can transition and thus this subcase is not possible.
Subcase 4.8: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3$ We know by `Nsys`, that only a receive transition can be discarded. This means for our supply transition, at least 1 system has to transition and thus this subcase is not possible.

Case 5: Consider $\alpha = \tau$.
Subcase 1.1: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3'$. For a τ transition, at most two systems can transition, the getting and the supplying one. As we have three systems transition for this subcase, this is not possible.
Subcase 5.2: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3$. For a τ transition, we need a get and a supply transition. As we only have one transitioning system, S_1, we know this one has to do both and thus we have a hidden transition. The only way this is possible is by `Ltau` twice. We get $S_1 \xrightarrow{\tau} S_1'$, resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\tau} (S_1' \mid\mid S_2) \mid\mid S_3$. By `Ltau`, we also get $S_1 \xrightarrow{\tau} S_1'$ for $S_1 \mid\mid (S_2 \mid\mid S_3)$. This is the same transition and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.
Subcase 5.3: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3$. For a τ transition, we need a get and a supply transition. As we only have one transitioning system, S_2, we know this one has to do both and thus we have a hidden transition. The only way this is possible is by `Ltau` and `Rtau`. We get $S_2 \xrightarrow{\tau} S_2'$, resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\tau} (S_1 \mid\mid S_2') \mid\mid S_3$. By `Rtau` and `Ltau`, we also get $S_2 \xrightarrow{\tau} S_2'$ for $S_1 \mid\mid (S_2 \mid\mid S_3)$. This is the same transition and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.
Subcase 5.4: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3'$. For a τ transition, we need a get and a supply transition. As we only have one transitioning system, S_3, we know this one has to do both and thus we have a hidden transition. The only way this is possible is by `Rtau`. We get $S_3 \xrightarrow{\tau} S_3'$, resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\tau} (S_1 \mid\mid S_2) \mid\mid S_3'$. By `Rtau` twice, we also get $S_3 \xrightarrow{\tau} S_3'$ for $S_1 \mid\mid (S_2 \mid\mid S_3)$. This is the same transition and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.
Subcase 5.5: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3$. For a τ transition at most two systems can transition. One which send a get request and one which supplies. As we have two transitioning systems, S_1 and S_2, we know one has to supply and one has to get, giving us two subcases.
Subcase 5.5.1: Consider S_1 being the getting system, meaning S_2 supplies. Consider `Ltau` we get $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$. By `Luni`, we then get $S_1 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2'$, resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\tau} (S_1' \mid\mid S_2') \mid\mid S_3$. By `Luni` and `Lsup`, we also get $S_1 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2'$ for $S_1 \mid\mid (S_2 \mid\mid S_3)$. These are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.
Subcase 5.5.2: Consider S_2 being the getting system, meaning S_1 supplies. Consider `Ltau` we get $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$. By `Runi`, we then get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_2'$, resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\tau} (S_1' \mid\mid S_2') \mid\mid S_3$. By `Runi` and `Lget`, we also get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_2'$ for $S_1 \mid\mid (S_2 \mid\mid S_3)$. These are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.
Subcase 5.6: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3'$. For a τ transition at most two systems can transition. One which send a get request and one which supplies. As we have two transitioning systems, S_1 and S_3, we know one has to supply and one has to get, giving us two subcases.
Subcase 5.6.1: Consider S_1 being the getting system, meaning S_3 supplies. Consider `Luni` and `Lget` we get $S_1 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_1'$ and $S_3 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_3'$, resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\tau} (S_1' \mid\mid S_2) \mid\mid S_3'$. By `Luni` and `Rsup`, we also get $S_1 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_1'$ and $S_3 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_3'$ for $S_1 \mid\mid (S_2 \mid\mid S_3)$. These are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.
Subcase 5.6.2: Consider S_3 being the getting system, meaning S_1 supplies. Consider `Runi` and `Lsup` we get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$ and $S_3 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_3'$, resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\tau} (S_1' \mid\mid S_2) \mid\mid S_3'$. By `Runi` and `Rget`, we also get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$ and $S_3 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_3'$ for $S_1 \mid\mid (S_2 \mid\mid S_3)$. These are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.

Subcase 5.7: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3'$. For a τ transition at most two systems can transition. One which send a get request and one which supplies. As we have two transitioning systems, S_2 and S_3, we know one has to supply and one has to get, giving us two subcases.
Subcase 5.7.1: Consider S_2 being the getting system, meaning S_3 supplies. Consider `Luni` and `Rget` we get $S_2 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_3'$, resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\tau} (S_1 \mid\mid S_2') \mid\mid S_3'$. By `Rtau` and `Luni`, we also get $S_2 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_3'$ for $S_1 \mid\mid (S_2 \mid\mid S_3)$. These are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.
Subcase 5.7.2: Consider S_3 being the getting system, meaning S_2 supplies. Consider `Runi` and `Rsup` we get $S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_3'$, resulting in $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\tau} (S_1 \mid\mid S_2') \mid\mid S_3'$. By `Rtau` and `Runi`, we also get $S_2 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_3'$ for $S_1 \mid\mid (S_2 \mid\mid S_3)$. These are the same transitions and shows $(S_1 \mid\mid S_2) \mid\mid S_3 \equiv S_1 \mid\mid (S_2 \mid\mid S_3)$ for this case.
Subcase 5.8: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3$. For a τ transition, at least one system has to transition. As by `Nsys` only a receive transition can be discarded. Therefore this subcase is not possible.

These cover all cases and therefore we have proven that associativity also holds. □
3. For the final point we want to show that a parallel composition with one of the processes being the empty process, acts the same as just the non-empty system. We will once again go by case analysis on α. We know an empty process cannot do anything, besides discarding a receive transition as by `Nnul`. Besides the receiving transition, we cannot discard other transitions. The only possibilities are then S_1 performing the corresponding α transition.
Case 1: Consider $\alpha = \Pi!(\tilde{v})^{ch}$.
Subcase 1.1: Consider `Lpar`, we get $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$. By `Sys` and our α, we can also have that $S_1 \xrightarrow{\Pi!(\tilde{v})^{ch}} S_1'$ is the only possibility on the RHS.
Subcase 1.2: Consider `Rpar`, this would require the empty process to perform the sending action, which we know is not possible by `Nnul`.
Case 2: Consider $\alpha = \Pi?(\tilde{v})^{ch}$.
Subcase 2.1: Consider `Prec`, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and together with `Nsys` and `Nnul` also $\Gamma : 0 \xrightarrow{\Pi?(\tilde{v})^{ch}} \Gamma : 0$. By `Sys`, we also get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ on the RHS.
Subcase 2.2: Consider `Prec`, we get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1'$ and together with `Nsys` on both this becomes $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$ and by `Nnul` also $\Gamma : 0 \xrightarrow{\Pi?(\tilde{v})^{ch}} \Gamma : 0$. By `Nsys`, we also get $S_1 \xrightarrow{\Pi?(\tilde{v})^{ch}} S_1$ on the RHS.
Case 3: Consider $\alpha = \Pi G(\tilde{v})^{ch}$.
Subcase 3.1: Consider `Lget`, we get $S_1 \xrightarrow{\Pi G(\tilde{v})^{ch}} S_1'$. By `Sys` together with our α, we can also get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$ on the RHS.
Subcase 3.2: Consider `Rget`, this would require the empty process to send the get request, which we know is not possible by `Nnul`.
Case 4: Consider $\alpha = \Pi S(\tilde{v})^{ch}$.
Subcase 4.2: Consider `Lsup`, we get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$. By `Sys`, we can also get $S_1 \xrightarrow{\Pi S(\tilde{v})^{ch}} S_1'$ on the RHS.
Subcase 4.2: Consider `Rsup`, this would require the empty process to supply, which we know is not possible by `Nnul`.
Case 5: Consider $\alpha = \tau$.

Subcase 5.1: Consider Luni, this would require the empty process to supply, which is not possible by Nnul.
Subcase 5.2: Consider Runi, this would require the empty process to send a get request, which is not possible by Nnul.

Subcase 5.3: Consider Ltau, we get $S_1 \xrightarrow{\tau} S_1'$. By Sys, we can also get $S_1 \xrightarrow{\Pi \mathsf{S}(\tilde{v})^{ch}} S_1'$ on the RHS.
Subcase 5.4: Consider Rtau, this would require the empty process to perform the hidden τ transition, which is not possible by Nnul.
As we have covered all cases we have proven that also $S_1 \mid\mid \Gamma : 0 \equiv S_1$. □

A.2 Lemma 2

Given 2 systems S_1 and S_2 proof:

Lemma A.2.1 *Lemma 2 is regarding the non-blocking behavior of broadcasted messages.*
$ch = * \wedge S_1 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \implies S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ *for any* S_2.

To prove this lemma we will go over all possible cases of S_2. The cases for this can be deducted from the grammar. Our base induction case will be $S_2 = \langle \gamma, \mathsf{LS} \rangle : P$. Which means by Lpar we know that we need to prove $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. By Nsys, this also includes $\langle \gamma, \mathsf{LS} \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P$. Because it is about non-blocking behavior, we can assume that it is possible for $S_2' = S_2$.
Case 1: $P = 0$. An empty process can only discard a message as per Nnul, by Nsys we then get $\Gamma : 0 \xmapsto{\pi?(\tilde{v})^{ch}} \Gamma : 0$. As Nnul has no premises, we have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, making it non-blocking as required.

Case 2: $P = \Pi!(\tilde{E})^{E'}.U$. We can only apply Nsnd, as we know we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. We can simply apply Nsnd in this case, as it has no premises. We have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, making it non-blocking as required.
Case 3: $P = \Pi?(\tilde{x})^{E'}.U$.
Subcase 3.1: Consider Rec if $[\![E']\!]_\gamma = * \wedge gamma \models \{\Pi[\tilde{v}/\tilde{x}]\} \wedge \gamma \models \pi'$, S_2 will receive the message, resulting in $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$.
Subcase 3.2: Consider Nrec, we know that $ch \in \mathsf{LS}$, as the broadcast channel always is. This means if $\big(ch = * \ \wedge \ (\gamma \not\models \{\Pi[\tilde{v}/\tilde{x}]\} \ \vee \ \gamma \not\models \pi')\big)$, we can apply Nrec with Nsys resulting in $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2$.
As we have that $ch = *$ and we know the broadcast channel is always in LS, we get that either $\gamma \models \{\Pi[\tilde{v}/\tilde{x}]\} \wedge \gamma \models \pi'$ or $\gamma \not\models \{\Pi[\tilde{v}/\tilde{x}]\} \ \vee \ \gamma \not\models \pi'$ has to hold. This is a tautology and thus we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, making it non-blocking as required.
Case 4: $P = \Pi\mathsf{G}(\tilde{x})^{E'}.U$. Consider Nget, which is the only applicable rule for this P, as we know we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. There are no premises, meaning we can apply this rule regardless. We then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, making it non-blocking as required.
Case 5: $P = \mathsf{S}(\tilde{E})^{E'}.U$. Consider Nsup, which is the only applicable rule for this P, as we know we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. The premise holds as we know $ch = *$. We then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, making it

non-blocking as required.
Case 6: $P = \langle\Pi\rangle P$.
Subcase 6.1: Consider Grd, if $\gamma \models \Pi$ and $\langle\gamma, \mathsf{LS}\rangle : P \overset{\alpha}{\mapsto} \langle\gamma, \mathsf{LS}\rangle : P$, we can receive the message. As we know we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$, we know α, has to be the receiving transition, which we covered in case 3.
Subcase 6.2: Consider Blk, in case $\gamma \not\models \pi'$, we can discard.

Subcase 6.3: Consider Stc, if $\gamma \models \Pi$ and we can do $\langle\gamma, \mathsf{LS}\rangle : P \overset{\widetilde{\pi'?(\tilde{v})^{ch}}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P$, which is covered in the previous cases, we can discard.

We then have $\big(\gamma \models \Pi \wedge (\langle\gamma, \mathsf{LS}\rangle : P \overset{\widetilde{\pi'?(\tilde{v})^{ch}}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P \vee \langle\gamma, \mathsf{LS}\rangle : P \overset{\pi'?(\tilde{v})^{ch}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P)\big) \vee \gamma \not\models \Pi$, which is a tautology. We then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, making it non-blocking as required.
Case 7: $P = P_1 + P_2$.
Subcase 7.1: Consider Lor, if $\langle\gamma, \mathsf{LS}\rangle : P_1 \overset{\pi'?(\tilde{v})^{ch}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P_1'$, we have $\langle\gamma, \mathsf{LS}\rangle : P_1 + P_2 \overset{\pi'?(\tilde{v})^{ch}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P_1'$. Note we use $\alpha = \pi'?(\tilde{v})^{ch}$ because we know we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2$.
Subcase 7.2: Consider Ror, if $\langle\gamma, \mathsf{LS}\rangle : P_2 \overset{\pi'?(\tilde{v})^{ch}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P_2'$, we have $\langle\gamma, \mathsf{LS}\rangle : P_1 + P_2 \overset{\pi'?(\tilde{v})^{ch}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P_2'$. Note we use $\alpha = \pi'?(\tilde{v})^{ch}$ because we know we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2$.

Subcase 7.3: Consider Nor, by the previous cases we have shown $\langle\gamma, \mathsf{LS}\rangle : P \overset{\widetilde{\pi?(\tilde{v})^{ch}}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P$ is non-blocking. This means for $\langle\gamma, \mathsf{LS}\rangle : P_1 \overset{\widetilde{\pi?(\tilde{v})^{ch}}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P_1$ and $\langle\gamma, \mathsf{LS}\rangle : P_2 \overset{\widetilde{\pi?(\tilde{v})^{ch}}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P_2$ and thus for $\langle\gamma, \mathsf{LS}\rangle : P_1 + P_2 \overset{\widetilde{\pi'?(\tilde{v})^{ch}}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P_1 + P_2$ it is also non-blocking for broadcast.

As these are all cases for $P = P_1 + P_2$, we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, making it non-blocking as required. **Case 8:** $P = K(\tilde{x})$.

Subcase 8.1: Consider Def, if $\langle\gamma, \mathsf{LS}\rangle : P \overset{\pi'?(\tilde{v})^{ch}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P'$, we can apply this rule and receive the message.

Subcase 8.2: Consider Ndef, if $\langle\gamma, \mathsf{LS}\rangle : P \overset{\widetilde{\pi'?(\tilde{v})^{ch}}}{\longmapsto} \langle\gamma, \mathsf{LS}\rangle : P$, which by our prevous cases has been shown to be non-blocking for broadcast, we can discard it.

Both cases result in $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, making it non-blocking as required.
For the inductive step, we take $S_2 = S_i || S_j$. This inductive step follows from Prec. Which can be applied in combination with Lpar and together with the base cases showing it is non-blocking for broadcasted messages.□

A.3 Lemma 3

Lemma A.3.1 *Lemma 3 shows the blocking of multi cast channels.*
$ch \neq * \wedge S_1 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \implies S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ *for any S_2 such that:*

1. $ch \in \mathsf{LS} \wedge S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$
   $\vee$
2. $ch \notin \mathsf{LS}$

We will use the same cases as in **??** but divide them into the two subcases. Once again to prove this lemma we will go over all possible cases of S_2. The cases for this can be deducted from the grammar. Our base induction case will be $S_2 = \langle \gamma, \mathsf{LS} \rangle : P$. Which means by `Lpar` we know that we need to prove $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. By `Nsys`, this also includes $\langle \gamma, \mathsf{LS} \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : P$. Because it is about blocking behavior, we can assume that it is possible for $S_2' = S_2$ if appropriate.

Case 1: $P = 0$. Consider `Nnul`, this rule has not premises. We have that an empty process cannot block an other process. An empty process cannot progress to anything and therefore, we do not want it blocking a multi cast channel. For both subcases this is the same and we have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ as required.

Case 2: $P = \Pi!(\tilde{E})^{E'}.U$. Consider `Nsnd`, as we need to get $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. Just like the last case, the reasoning for both subcases is the same as we have no premises we can always apply this rule. This is desirable because we can only send one message at a time and do not want it to block other messages being send. We then have we have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ as required.

Case 3: $P = \Pi?(\tilde{E})^{E'}.U$.

Subcase 3.1:

Subcase 3.1.1: Consider `Rec`, we have that if $[\![E']\!]_\gamma = ch \wedge \gamma \models \{\Pi[\tilde{v}/\tilde{x}]\} \wedge \gamma \models \pi' \wedge ch \in \mathsf{LS}$ then $\langle \gamma, \mathsf{LS} \rangle : \Pi?(\tilde{x})^{E'}.U \xmapsto{\pi'?(\tilde{v})^{ch}} \{\!|\langle \gamma, \mathsf{LS} \rangle : [\tilde{v}/\tilde{x}]U|\!\}$. We know $ch \in \mathsf{LS}$, so we are left with the premises $[\![E']\!]_\gamma = ch \wedge \gamma \models \{\Pi[\tilde{v}/\tilde{x}]\} \wedge \gamma \models \pi'$.

Subcase 3.1.2: Consider `Nrec`, as we know $ch \in \mathsf{LS}$, leaving us with $\big(ch = * \wedge (\gamma \not\models \{\Pi[\tilde{v}/\tilde{x}]\} \vee \gamma \not\models \pi')\big)$. We know $ch \neq *$, meaning $\langle \gamma, \mathsf{LS} \rangle : \Pi?(\tilde{x})^{E'}.U \xmapsto{\widetilde{\pi'?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS} \rangle : \Pi?(\tilde{x})^{E'}.U$ is never possible in this case. What we are left with it that for it to be blocking either $[\![E']\!]_\gamma \neq ch$, meaning it is expecting a message from a different channel, which means the process it is doing multiple jobs which is undesirable and thus appropriate blocking. Or $\gamma \not\models \{\Pi[\tilde{v}/\tilde{x}]\} \wedge \gamma \not\models \pi'$, which means a process is not ready yet to receive and that the system should wait or something went wrong, which is why it is appropriate to be blocking.

Subcase 3.2

Subcase 3.2.1: Consider `Rec`, we have that $ch \notin \mathsf{LS}$, meaning this is not possible as $ch \in \mathsf{LS}$ is a premise.

Subcase 3.1.2: Consider `Nrec`, our premise is $ch \notin \mathsf{LS} \vee \big(ch = * \wedge (\gamma \not\models \{\Pi[\tilde{v}/\tilde{x}]\} \vee \gamma \not\models \pi')\big)$. We have that $ch \notin \mathsf{LS}$, thus we can always perform this transition, making it non-blocking if the process is not listening to the channel, which is desirable as they should not intervene with processes they are not part of.
This shows that in this case it is blocking the multi cast channels appropriately.

Case 4: $P = \Pi\mathsf{G}(\tilde{x})^{E'}.U$. Consider `Nget`, as we need to get $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. Just like the first two cases, the reasoning for both subcases is the same as we have no premises we can always apply this rule. This is desirable because we can only send one message at a time and do not want it to block other messages being send. We then have we have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ as required.

Case 5: $P = \mathsf{S}(\tilde{x})^{E'}.U$. **Subcase 5.1:** Consider `Nsup`, as we know we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. We know $ch \neq *$, leaving us with the premise $ch \notin \mathsf{LS}$, which we know is false as well. This means it would block the multi-cast message, which is desirable as unicast and multicast should be done on seperate channels.

Subcase 5.2: Consider `Nsup`, as we know we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. We know $ch \neq *$, leaving us with the premise $ch \notin \mathsf{LS}$, which we know is true. This means it does not block the message because when it is not listening to the channel, which is desirable as they should not intervene with processes they are not part of.
We then have we have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ as required.

Case 6: $P = \langle\Pi\rangle P$. None of the cases depend on *ch* directly, meaning we can cover both cases at the same time.

Subcase 6.1: Consider `Grd`, we have covered the $\gamma \models \Pi$ and $\langle \gamma, \mathsf{LS}\rangle : P \xmapsto{\pi'?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}\rangle : P$ premise in the previous cases, meaning if $\gamma \models \Pi$ holds, it will only block when appropriate. This is the case for both of our subcases

Subcase 6.1: Consider `Stc`, if $\gamma \not\models \Pi$, it will discard regardless. This is desirable because an internal guard means a process should not be participating (yet), meaning it should not block the other processes.

Subcase 6.1: Consider `Stc`, we have covered the $\gamma \models \Pi$ and $\langle \gamma, \mathsf{LS}\rangle : P \xmapsto{\widetilde{\pi'?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}\rangle : P$ premise in the previous cases, meaning if $\gamma \models \Pi$ holds, it will only discard when appropriate.

We have covered all of the subcases and we have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ as required, only blocking when appropriate as covered by the previous cases.

Case 7: $P = P_1 + P_2$. Just like the previous case none of the cases depend on *ch* directly, meaning we can cover both cases at the same time.

Subcase 7.1: Consider `Lor`, if the premise $\langle \gamma, \mathsf{LS}\rangle : P_1 \xmapsto{\pi'?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}\rangle : P_1'$ holds, we know it accepts and we get $\langle \gamma, \mathsf{LS}\rangle : P_1 + P_2 \xmapsto{\pi'?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}\rangle : P_2'$.

Subcase 7.2: Consider `Ror`, if the premise $\langle \gamma, \mathsf{LS}\rangle : P_2 \xmapsto{\pi'?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}\rangle : P_2'$ holds, we know it accepts and we get $\langle \gamma, \mathsf{LS}\rangle : P_1 + P_2 \xmapsto{\pi'?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}\rangle : P_1'$.

Subcase 7.3: Consider `Nor`, we know $\langle \gamma, \mathsf{LS}\rangle : P_1 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}\rangle : P_1$ and $\langle \gamma, \mathsf{LS}\rangle : P_2 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}\rangle : P_2$ is only possible when it should be non-blocking.

As these are all cases for $P = P_1 + P_2$, we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, making it blocking only if appropriate as required.

Case 8: $P = K(\tilde{x})$. Just like the previous cases none of the cases depend on *ch* directly, meaning we can cover both cases at the same time.

Subcase 8.1: Consider `Def`, if $\langle \gamma, \mathsf{LS}\rangle : P \xmapsto{\pi'?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}\rangle : P'$, we can apply this rule and receive the message.

Subcase 8.2: Consider `Ndef`, if $\langle \gamma, \mathsf{LS}\rangle : P \xmapsto{\widetilde{\pi'?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}\rangle : P'$, we can discard the message, which by the previous cases can only be done when appropriate.

As these are all cases for $P = K(\tilde{x})$, we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, making it blocking only if appropriate as required.

For the inductive step, we take $S_2 = S_i \mid\mid S_j$. This inductive step follows from `Prec`. Which can be applied in combination with `Lpar` and together with the base cases shows it is blocking for multi-cast messages when it should be.□