CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Physical Space in Reconfigurable Interacting Systems

Implementing physical space in a multi agent system

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

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

In this thesis, we provide the expressive LaC process calculus and accompanying semantics for modeling multi-agent systems (MAS). LaC-calculus is an extension of R-CHECK. It allows for modeling of MAS inside a discrete physical space environment, which influences the behavior of the systems inside. In addition, it proves three different communication methods (broadcast, multicast and unicast) and the movement of systems inside the physical space. The calculus consists of three levels: the environment, the system and the process level. The systems can self-organize tasks, communication and movement based on their configurations. The environment influences the behavior of the systems as well, by blocking movement that cannot be made and by providing local communication methods.

Keywords: Computer, science, computer science, engineering, project, thesis, process calculus, semantics, MAS.

Acknowledgements

To begin I would like to thank my supervisor Yehia Abd Alrahman. Throughout the thesis process we had weekly meetings in which we went over the work done, discussed possible next steps and also solved some problems together. Besides work-related help, he was also very understanding of personal situations which occurred and accommodated the situation.
Next, I would like to thank my examiner Nir Piterman. He was also very accommodating and provided essential feedback to improve the work presented in this thesis.
I also want to thank my partner, parents, family and friends. In the last few months, they have helped me take my mind off of the work when it became overwhelming. When planning or motivation problems came up they also helped me, by either giving me planning tips or helping me to find the motivation again.

Tom de Ridder, Gothenburg, 2024-09-20

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 industries, with wildly 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. While performing such tasks, it is natural for humans to communicate with the people in the same room without 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 specified protocol. For machines, however, this is not the case. Often communication between machines goes through global channels. To these channels multiple groups of machines, often 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 communication through such a global system is expensive. It uses a lot of resources to build these systems, takes time to communicate back and forth, and in addition you also need more processing space for the large protocols. The reason for this is that each system is 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. By leaving a communication channel when leaving a space, rather than doing it manually is just an example of this. To achieve this, the systems have to be able to reconfigure themselves and adjust their behavior based on the other processes and their environment. With physical space as part of the implementation, systems will be able to self-organize these actions, providing a desirable efficiency [23].

An important aspect of creating reconfigurable communicating systems is modeling the specification. These specifications are usually written in a process calculus [11]. Many such languages exist, but the scope of the behavior they model differs. Currently, there do not exist many languages that have native support for localized communication. This is where the research presented in this thesis comes in. It will introduce physical space as a first-class citizen, allowing for reasoning on the influence the location of a system has on its behavior, which has not often been done before. The spacial relations will be modeled in a manner which allows shortest path navigation without the need to store a full map. Having an expressive language to model such communicating systems and their environments can help make advancements in the area. Especially if the specification of such a model can easily be checked by a program for its correctness. Such a model checker exists, R-CHECK [2], however, its current process calculus does not offer native support for reasoning about space. To use communication based on locality, it has to be hardcoded. This is a tedious process and prone to mistakes, while also taking more resources. We will take R-CHECK as our base and

build our work from there.

1.1 Problem and goals

The main problem which needs to be solved is 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]. For this thesis, it includes the influence a location inside a space has on the behavior as well. In which ways does it limit or extend the possible behavior of the system.

The goal are as follows:

1. Starting from R-CHECK, create a process calculus consisting of a syntax and accompanying semantics. This calculus should allow for modeling of the systems' individual behaviors, communications and interactions.
2. Extend the process calculus with physical space as a first-class citizen. This should model how a system's location inside a space influences their behavior. Having space as a first-class citizen allows for reasoning about environment-based behavior in the same way as the interactions amongst systems themselves.
3. The systems should be able to share information amongst each other, deduce information from their environment and use the information to perform the appropriate actions. This should be done 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 of relevant papers and highlighting where progress can be made.

Next up will will present the LaC-calculus, the syntax and the semantics of our system. This will be done in a step-by-step manner with the help of examples. Important lemmas will be formulated with their formal proofs presented in the appendix. We will start with some design choices. Next, the syntax will be presented, after which the semantics follow. Finally, the physical space, or environment, will be presented, including its semantics. We will reason about the communication and the movement within the space. To show the purpose of our results, we will present a scenario with its corresponding specification and behavior.

Lastly, we will conclude our research, summarize what has been achieved, and highlight the relevance. We will also discuss what areas can still be improved 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 a specific relevant research, R-CHECK, and identify where advancements can be made. This should give us a good starting point for our 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 types of systems consist of several entities that 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 be used to reach a common goal. The automata in Figure 2.1 shows two systems interacting. Here the transitions with a label ending in an exclamation mark and the transition ending in a questions mark are complimentary, when the rest of the label is identical. The first transition is the sent message, where the latter is the reaction to the corresponding message.

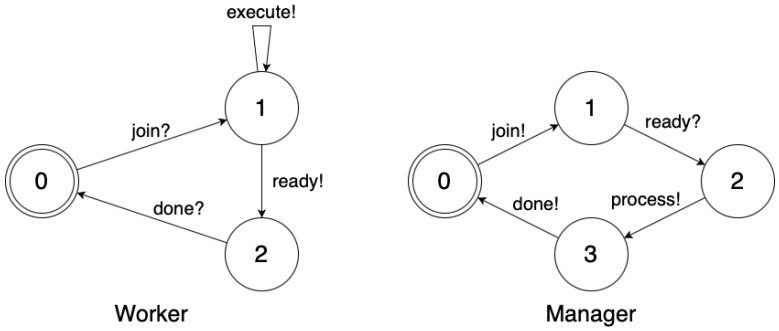

Figure 2.1: Two interacting systems

2.1.1.1 Multi-agent systems

One type of reconfigurable interacting systems is the multi-agent systems (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 agent, 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 for use on public roads with the exception of approved safety trials [8]. This shows us that 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 modeling.

CAS focus on optimizing the limited resources of a system. During runtime the different components are able to join and leave collectives at any time, 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 modeling quite tedious. Especially for arguing about the 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:

- Broadcast, mentioned earlier, is the main method. Broadcast means everyone who is able to receive a sent message, receives it on a global channel. Think of a TV broadcast everyone can watch, if they

turn on their device.

- Multicast is the second type of communication. In this case everyone connected to a certain channel receives the message. Relating it to the TV broadcast, this could be a paid channel. Only the people who have the subscription can watch it. For our system it will however be more like a team meeting, which can only be held if everyone on the team is available.
- Unicast is the last method we will use. As the name suggests, 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 another process. This could be compared to a one-on-one meeting with your supervisor. Same as for multicast, this should not be done if there are not two systems available to have unicast communication.

Channeled communication can also be used for CAS, but as limited human interaction is desired, it has to be modeled 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, an 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 of agents, 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 or behavior. Their models can become quite complex and as a result 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 cornerstones 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 their correctness and their behavior [20]. A transition system is a model consisting of nodes and edges, showing how a system behaves. As CCS deals with transition systems, it shows how over 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 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 tackles the problem of limited expressiveness. Instead of predefined actions, it utilizes variables that 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 as 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 with the ability to store it in a data structure and to adjust their behavior based on it. The different spaces we will call environments, each with their own environmental properties. In real life, these environments are continuous, this means that not only the processes can move and change environments, but that these spaces can also move themselves. The opposite of this continuous space is discrete space, which is where our focus will lie. The environments themselves cannot move locations in the physical space, meaning the space representation does not change. Even though this does limit the possibilities of our system, it creates a base for future research with continuous space. There have been various papers on logical reasoning of spatial relations, such as by Randell et al. [21]. A number of spatial relations are possible, but due to limited time, the focus will lie on whether a pair of environments is (externally) connected or disconnected. These two relations provide the basis for reasoning about the ability of processes to change from one environment to another, while limiting the space completeness.

2.2 R-CHECK

One of the recently developed agent-based modeling programs in this field is R-CHECK [14]. As the name suggests, it is able to check a model of MAS on correctness. R-CHECK allows support for a 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 uses `LTOL`, which is an extension of `LTL` (Linear-time Temporal Logic), to write high-level descriptions of behavior that can be checked against models. 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 that 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, the movements of systems and their environment-based interactions can be modeled in a discrete way. As R-CHECK only allows a single message being sent at a time, having the environment 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 environment behavior from the system behavior, we can have separate reasoning, influenced by the environment definition rather than the system definition. 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 also be the focus of this thesis. Starting from R-CHECK as a base, we will create a process-calculus inspired by it. We will extend this calculus with unicast and physical space as a first-class citizen. This includes reasoning about the movement of processes and the way their environment affects this.

3

Theoretical results

3.1 Design choices

To start we will go over some design choices made while developing this calculus. Starting with communication, every system should always be listening to the broadcast channel. Broadcast is meant to reach every system. In addition, we wanted broadcast to be non-blocking for this reason. Non-blocking means: Intended receivers will receive the broadcast message, but systems which are not supposed to receive the broadcast message, should not block it from going through.

Multicast on the other hand should be blocking, meaning if not every system listening to the channel is ready to receive, the message cannot go through. The reasoning for this is that multicast channels are designed to self-organize teams and communication. If a member of the team is not ready to receive yet, the system should wait to progress. This goes for every system that is listening to the channel, whether it is waiting to leave or to perform a unicast operation. This way the blocking behavior constrains the order in which certain actions will be performed, for example, a system will leave the room when it is done before the other processes continue.

Unicast should never go unanswered, this means unicast is blocking if no system is able to supply the requested information. The system sending the request for information will have to wait until at least one system can supply this information before it can continue, this helps to ensure that information does not get lost.

Movement is environment-dependent, meaning a system can request a move, but only if the environments included in this move are able to allow or block this movement. The reason for this is that we do not want a system to be able to enter a room if it has reached its capacity or to just update its location to one it is not currently in.

Regarding the physical space, we have decided to make it discrete. This makes it easier to reason about its behavior and correctness. In the future work section, we will go more in depth on the possible next steps.

Interspace communication is allowed, with the broadcast channel reaching all environments. Each environment has a unique channel, which is separate from a system's own stored channels. A system can communicate over both its own channels as well as the environment's channel. By separating these two, a system automatically stops listening to the local channel when its location is updated.

Our system will have universal attributes. Each system has the same attributes in its configuration, possibly with different values. This way modeling systems allows us to always access values on request without the need to know the local variable name. The environments and systems will both have a unique identifiable attribute, this way certain actions are required to be deterministic rather than arbitrary.

3.2 Context

To support our explanations in the next section, we will briefly describe the context of our systems and their environments. In Figure 3.1 we see a map with numbered sections. These sections we will call environments from now on. Each environment has a unique number by which it can be identified. We also see systems, with a unique letter, exist inside the space.

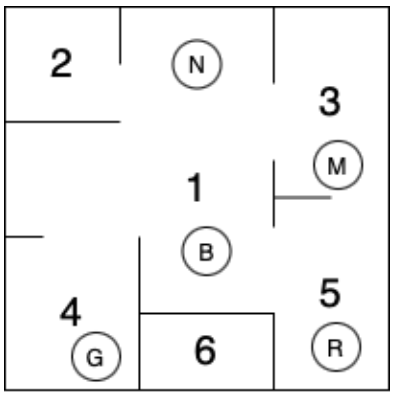

Figure 3.1: Example of space layout with systems.

We can see that some environments have one or multiple systems inside of them, but some have none. This shows that an environment is still included in our context, even if no system is actively aware of said environment. Currently, we do not know anything about the specifications of the environments of the systems, besides their existence and identifier. How these different components can be specified will be explained in the next sections. We will refer back to the example Figure 3.1.

3.3 The syntax

Now that we have clarified some design choices and given a context, it is time to introduce our developed language, the LaC-Calculus. The first step to implementing physical space into multi-process systems is having a syntax to express our systems and environments. We will explain the syntax with the support of a step-by-step example for clarification.

Environment	$E ::=$	$[S]^{\mathsf{RC},\ell}_{c,n} \quad \mid \quad E_1 \oplus E_2$
Systems	$S ::=$	$\Gamma : P \quad \mid \quad S_1 \mid\mid S_2$
Configuration	$\Gamma ::=$	$\langle \gamma : \mathscr{A} \to \mathscr{V}, \quad \mathsf{LS}, \quad \ell \rangle$
Processes	$P ::=$	$\Pi!(\tilde{e})^e.U \quad \mid \quad \Pi?(\tilde{x})^e.U \quad \mid \quad \Pi\mathsf{G}(\tilde{x})^e.U \quad \mid$
		$\mathsf{S}(\tilde{e})^e.U \quad \mid \quad \mathsf{MOV@}\Pi.U \quad \mid \quad \langle \Pi \rangle P \quad \mid$
		$P_1 + P_2 \quad \mid \quad K(x_1, ..., x_n) \quad \mid \quad 0$
Updates	$U ::=$	$P \quad \mid \quad [a := e]U \quad \mid \quad \mathtt{OP}(\mathsf{LS})U \quad \mid \quad [\ell \mapsto \ell']U$
Expressions	$e ::=$	$v \quad \mid \quad x \quad \mid \quad a \quad \mid \quad ch \quad \mid \quad e_1 \otimes e_2$

Table 3.1: The syntax of LaC-calculus.

At the top level, we can see the environment, which is the physical space in which systems exist. An environment has a set of reachability connections, RC. This is a set of mapped values $\ell \mapsto \mathbb{N}$, for every environment it maps its unique identifier to an integer value. This integer value is equal to the number of environments in between itself and the corresponding environment. This allows systems to go to a destination without being aware of the whole map, which will be shown in an example later on. For Figure 3.1 we have for example $\mathsf{RC} = \{1 \mapsto 0, 2 \mapsto 1, 3 \mapsto 1, 4 \mapsto 1, 5 \mapsto 1, 6 \mapsto \infty\}$ and $\mathsf{RC} = \{1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 1, 4 \mapsto 2, 5 \mapsto 0, 6 \mapsto \infty\}$ for the environments with number 1 and 5 respectively.
Next, we have the unique local channel identifier ℓ. This attribute serves a double purpose. It is both a unique value by which an environment can be identified and also the local channel (which is unique by default). As every system inside an environment is automatically listening to all local channels, only a single local channel is necessary as multicast is blocking. In Figure 3.1 we can set this value to the unique number on the map for each environment. This identifier has to be unique, as otherwise the reachability connection RC could be non-deterministic, resulting in a system moving to a different space with the same name or local channels being used non-locally.
Environment capacity c is a value set in the specification and cannot change. It represents the maximum number of systems an environment can accommodate at a time. If we take environment number 4 in Figure 3.1, and set its capacity to 1, assuming G is a single system, no other system can enter until G leaves.
Lastly, the number of systems in a room is represented by n. As mentioned for environment number 4 in Figure 3.1, this would be 1. For environment number 1 and 2, this would be 2 and 0 respectively, assuming N and B are single systems.
An environment is either a single environment or a composition of two environments. A composition means that two environments exist in the same context, but does not determine their relation. Rather, this is done by the environments' specifications explained before. As we are dealing with discrete space, environments will always exist as initially specified, with only the systems inside of them, and the number n, changing with time. Relating it to Figure 3.1 the composition would look like $E_1 \oplus E_2 \oplus E_3 \oplus E_4 \oplus E_5 \oplus E_6$ for every E_i with $i = \ell$.

Now that we have defined the top level, we can move on to the level below. This is the system-level. A system, S, is either a process with a configuration, which we will get to next, or a parallel composition of two systems, $S_1 \mid\mid S_2$. In Figure 3.1 these systems are represented by the nodes with a unique letter inside of them. If we take environment 4, we would have $[\Gamma_G : G]^{\mathsf{RC}_4,4}_{c_4,n_4}$. Environment 1 would look like $[\Gamma_N : N \mid\mid \Gamma_B : B]^{\mathsf{RC}_1,1}_{c_1,n_1}$.

The configuration of a process, Γ, consists of three things. The first one is γ, with the form $\mathscr{A} \to \mathscr{V}$. It is a set of attributes mapped to a set of values of any format. As mentioned earlier the set of attributes is the same for every configuration.
Next in the configuration we have LS, a set of channels the process is listening to. It is important to note that LS in the configuration does not include the local channel ℓ, as explained in Section 3.1.
Lastly, we have the location ℓ, which is the current location a system exists in. This location has to have the value of one of the unique identifiers in the environment composition. The process is aware of its current location, however it does not know anything else about the physical space it exists in. ℓ can only be used for the local channel or to request movements on the environment level. In Figure 3.1 M would for example have $\ell = 3$ and R would have $\ell = 5$ in its current configuration.

The next parts in our syntax are the lowest level, the process-level. The process-level has a more complex explanation. For this reason, we will start by giving a scenario. To focus on the new aspects compared to the aforementioned R-CHECK, the running example will cover unicast and movement. In the next chapter, we will give more examples for different actions and also provide an applicable real-life scenario that can be modeled using LaC-Calculus.

Example 3.3.1 *[Finished job (step 1/5)] For our example, we will take a simple setup to make it clear how the unicast communication works. As mentioned in Section 3.1, movements depend on the environments involved. For this reason, we will extend the example to include movement when we cover the environment semantics.*

A team of three agents has completed their job and gathered the necessary information. The team manager just needs to get the information from the two other agents to be able to report the overall results. It requests the two other agents to report back if they are done, and after reporting back they can move back to the hallway to wait for their next assignment. The setting (or the parallel composition of systems) S is as follows: $S \triangleq \Gamma_M : M \mid\mid \Gamma_Q : Q \mid\mid \Gamma_R : R$. *Here we have M, the team manager, which gathers the results. We also have Q and R, two agents with the same specification, apart from their configuration which will answer the manager's request before moving back to the hallway.*

The next question is, how do we define the processes that make up these systems, so we can model the example. As can be seen in Table 3.1, a process can take 9 different forms. Before we go over the actual process forms, 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 configuration, and can include the basic logical boolean operators. A predicate in the syntax has two different functions depending on the process. One is a predicate which will be concretized and sent along with a message to filter receivers. The other is a predicate to determine whether a process can participate in said action, we will specify which kind of predicate we are dealing with per process.
Next, we have an expression e. It can be a value v, a placeholder x, an attribute a or a channel ch. It can also be an operation over multiple expressions $e_1 \otimes e_2$, like a summation, boolean operator, etc.
In our process definition we can see $\tilde{}$, this represents a sequence of one or more expressions, allowing the passing of multiple values at a time.
Now that we have this information, we can explain Π under the configuration. When a predicate is sent along with a message we have $\{\Pi\}_\gamma = \pi$. The process sending concretizes the predicate and only receiving processes satisfying it can receive. We will also see two different evaluations $\gamma \models \{\Pi\}_\gamma$ (or π), which takes the current configuration and determines if the predicate holds in that configuration. The other no-

tation 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 placeholder in the sequences.

A system can update its configuration after executing a process, expressed by $.U$. Which U is applicable depends on both the specification and the process after which it occurs. The process-specific updates can be seen in 3.2, which will be explained more in-depth after introducing the semantics.
The first possible form of U is P. This is the next process in the system and suggests that the system is done updating its configuration.
Next, we have $[a := e]U$, the update of attributes in γ. This update changes attribute values based on a provided expression. An attribute update can rely on new information, but can also be specified in a hardcoded way.
The third form U can take is $\mathtt{OP}(\mathsf{LS})U$, updating its set of listening channels. This operator over LS either removes or adds channels, depending on the specification given. As we are dealing with a set this could for example be $\{ch\} \cup \mathsf{LS}$ or $\mathsf{LS} \backslash \{ch\}$. The broadcast channel should never be used in this function as it should always be in LS.
The last form of U is the location update, $[\ell \mapsto \ell']U$. When a process changes its environment, the configuration has to update to the new environment. This update should not be applied otherwise, this would be considered a specification error and should be checked for.
From these definitions of U, we get the update function $\{\!|\Gamma : P|\!\}$. The update function is of the following form:

$$\{\!|\Gamma : P|\!\} = \begin{cases} \{\!|\langle \gamma[a \mapsto [\![e]\!]_\gamma], \mathsf{LS}, \ell\rangle : U|\!\}, & \Gamma : P = \langle \gamma, \mathsf{LS}, \ell\rangle : [a := e]U \\ \{\!|\langle \gamma, \mathtt{OP}(\mathsf{LS}), \ell\rangle : U|\!\}, & \Gamma : P = \langle \gamma, \mathsf{LS}, \ell\rangle : \mathtt{OP}(\mathsf{LS})U \\ \{\!|\langle \gamma, \mathsf{LS}, \ell'\rangle : U|\!\}, & \Gamma : P = \langle \gamma, \mathsf{LS}, \ell\rangle : [\ell \mapsto \ell']U \\ \langle \gamma, \mathsf{LS}, \ell\rangle : P & \Gamma : P = \langle \gamma, \mathsf{LS}, \ell\rangle : P \end{cases}$$

This brings us to the actual processes. We will go over them in the same order as in Table 3.1. This will also be the same order in which they appear in the semantics.
First, we have $\Pi!(\tilde{e})^e.U$, the sending process. Here Π is not an internal predicate but rather a predicate sent along with the message. The sending process will concretize Π under the configuration as $\{\Pi\}_\gamma = \pi$ before sending the message. π only depends on the current values in the receiving process. Examples of these could be attributes like, their role, battery-level, or anything else that can determine if the process should receive the message. The sending process will evaluate the expression $\tilde{e}$ to $\tilde{v}$, the value(s) to send along. It also evaluates the expression ^{e} to a channel ch. The message is sent on this channel, which could be broadcast or multicast. 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 configuration according to the specification.
Now for the other side of this interaction, the receiving process $\Pi?(\tilde{x})^e.U$. Here Π is different from π, which the process will receive along with the message. This predicate is of the earlier explained form $\{\Pi[\tilde{v}/\tilde{x}]\}_\gamma$. Both Π and π will have to be satisfied in the configuration 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, which these placeholders will reserve. The placeholders can then be used for updates, meaning the predicates are essential to make sure only the correct systems receive the message. This however, is about the specification, not the syntax itself, and differs per setting. For this process ^{e} also gets evaluated to a channel and this channel has to match the channel on which the message is sent. After receiving the values, the system will perform the updates as specified, possibly with the received values.
The send and receive actions can be done on either broadcast or multicast channels. 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 multicast. Only processes with $ch \in \mathsf{LS}$ will receive this message, but this can still be multiple processes.
For unicast we first 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. It is again sent along with the request to filter out which processes can supply this information. It has a sequence of placeholders, $\tilde{x}$, for the information it is expecting to receive. Once again ^{e} is evaluated to a channel ch on which the request is sent. After receiving the requested information, it updates, possibly with the received values.
The get request from the previous process is answered by a supply process, $\mathsf{S}(\tilde{e})^{e}.U$. As you can see it does not have a guard or predicate. The reason for this is that it has not received any values and the system requesting the information has already been determined. The configuration of the system still has to satisfy π. The process evaluates $\tilde{e}$ and sends them back over the same channel, which can be evaluated from e and should be the same as ch over which the request arrived. It then updates its configuration according to the specification.
Up next is the moving process $\mathsf{MOV}@\Pi U$. As the name suggests this is regarding the mobility of a system. Internally this process confirms the system is ready to move, whether this move can be made will be dependent on the environments involved in the move. The sending process will concretize Π under its configuration and the receiving environment will have to satisfy this predicate. There is no direct destination environment here. The predicate sent along will determine where the move will be made to, albeit this could be non-deterministic. By specifying the moving process without a concrete destination, but rather a discrete representation of its destination, the modeling of movements will be less tedious. An example of this will be shown in the next chapter.
The other process forms are extensions of the already explained ones. This means that on their own they cannot perform an action, but rather the processes P in their syntax consist of other processes that can do this. In the semantics in Table 3.3 this will be clarified further.
First, we have a guarded process $\langle\Pi\rangle P$. This process has an internal guard and it can only continue with the process P if Π is satisfied under its own configuration. An example of such a guard could be waiting until the battery-level is sufficiently charged. It could also be used to put conditions on which action to perform next, which brings us to the next process.
The summation notation $P_1 + P_2$. This notation states that the system will either continue with the action in P_1 or P_2, but not both. The process which does not get executed will be dropped and the system will continue with the executed process. This choice can be non-deterministic depending on the specification. However, as mentioned before, combining this with the guarded process can allow us to influence which process the system will continue.
Up next is the definition process, $K(x_1,...,x_n)$. This is a parameterized process with defined placeholders. We require $x_i \neq x_j \text{iff} i \neq j$ for all $1 \leq i, j \leq n$ to be able to use the placeholders deterministically. This process allows us to use values previously received without storing them in the configuration. In the semantics, it can be seen how these placeholders are used in a process.
Last 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 it means we can terminate a process after it has done its part. The process being empty does not mean that the configuration is empty, however, we will not be able to access this configuration anymore as a system cannot continue from an empty process.

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

Example 3.3.2 *[Finished job (step 2/5)] Here are the specifications of the processes in Example 3.3.1.*

$$
\begin{array}{ll}
M \triangleq & (status = \text{"}Ready\text{"})G(x)^{id}.[result := result + x].M \\
Q \triangleq & S(result)^{manager}.H \\
R \triangleq & S(result)^{manager}.H
\end{array}
$$

We can see that we have some attributes, which we will explain in the order they appear in: status is the system status, which is a string and can be "Available", "Busy" or "Ready"; id is the unique identifier of a system; result is an integer value of the achieved results; manager is the unique id of the manager a system is currently performing under. Process H in Q and R is the moving back to the hallway, which will be specified in a later example.

3.4 The semantics

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

$$\lambda ::= \pi!(\tilde{v})^{ch} \quad | \quad \pi?(\tilde{v})^{ch} \quad | \quad \pi\mathsf{G}(\tilde{v})^{ch} \quad | \quad \pi\mathsf{S}(\tilde{v})^{ch} \quad | \quad \pi@(\Gamma : P, \ell)$$

In the order above we have the send, receive, get, supply and move transition labels. 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.

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

$$\texttt{Nsnd}\ \frac{ch \notin \mathsf{LS} \cup \{\ell\} \ \vee \ ch = *}{\langle \gamma, \mathsf{LS}, \ell\rangle : \Pi!(\tilde{e})^{e'}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell\rangle : \Pi!(\tilde{e})^{e'}.U}$$

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

$$\texttt{Nrec}\ \frac{\begin{array}{c} ch \notin \mathsf{LS} \cup \{\ell\} \ \vee \\ \big(ch = * \ \wedge \ (\gamma \not\models \{\Pi[\tilde{v}/\tilde{x}]\} \ \vee \ \gamma \not\models \pi)\big) \end{array}}{\langle \gamma, \mathsf{LS}, \ell\rangle : \Pi?(\tilde{x})^{e'}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell\rangle : \Pi?(\tilde{x})^{e'}.U}$$

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

$$\texttt{Nget}\ \frac{ch \notin \mathsf{LS} \cup \{\ell\} \ \vee \ ch = *}{\langle \gamma, \mathsf{LS}, \ell\rangle : \Pi\mathsf{G}(\tilde{x})^{e}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell\rangle : \Pi\mathsf{G}(\tilde{x})^{e}.U}$$

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

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

$$\texttt{Mov}\ \frac{\{\Pi\}_\gamma = \pi \quad \Gamma : P \triangleq \langle \gamma, \mathsf{LS}, \ell\rangle : \mathsf{MOV@}\Pi.U}{\langle \gamma, \mathsf{LS}, \ell\rangle : \mathsf{MOV@}\Pi.U \xmapsto{\pi@(\Gamma:P,\ell')} \{\!|\langle \gamma, \mathsf{LS}, \ell\rangle : [\ell \mapsto \ell']U|\!\}}$$

$$\texttt{Nmov}\ \frac{ch \notin \mathsf{LS} \cup \{\ell\} \ \vee \ ch = *}{\langle \gamma, \mathsf{LS}, \ell\rangle : \mathsf{MOV@}\Pi.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell\rangle : \mathsf{MOV@}\Pi.U}$$

Table 3.2: The process-level semantics 1/2.

In Section 3.3 we have discussed the majority of the building blocks of the semantics. We will briefly go over the semantics in Table 3.2 to provide clarification. First, we will explain the left-hand side of Table 3.2, which are the transitions with label λ. After that, we will go over the complementary discard transitions of the processes with the transition label $\widetilde{\pi?(\tilde{v})^{ch}}$.

We will first look at `Snd`, the sending transition. When a process wants to send a message, it means it is ready to. It only has to evaluate and concretize the expressions and predicate to send along. This transition can be performed over broadcast and multicast. Which type of communication it concerns depends on the evaluation of e' to ch. If $ch = *$ it is a broadcast message, otherwise it is a multicast message. After it has sent the message it can update according to the specification.
The processes which can receive the message from the previous rule are the ones that satisfy `Rec` for the message. For this to be possible e' has to evaluate to the same ch as the sent message and this channel has to be either inside the system's LS or be the local channel. It also has to satisfy both predicates as explained in Section 3.3. If all of these premises are satisfied, the process can receive the message and update its system's configuration taking into account the received message.
The rule `Get` has similar premises as `Snd`. When it is ready to request information, it will wait to do so. The difference in the premises is the evaluation of values to send along is missing. This is because it has placeholders for the values it is expecting to receive. Just like in `Rec` it will then proceed to update its configuration taking into account the received values.
The process that can supply the previous request follow `Sup`. Just like in `Rec`, it has to satisfy the predicate sent along and it has to be listening to the channel the request came over. The channel over which the values are then sent also has to be the same. The values are evaluated as explained in Section 3.3.
Last in the left column we have `Mov`. As explained in Section 3.1 movement depends on the environments involved. Just like `Snd` and `Get` it will wait until it is allowed to perform this transition, but internally it is ready. The predicate sent along with the move has to be satisfied by the receiving environment. To clarify only a single system, and not a parallel composition of systems, can move, the label also includes the single system and its configuration. As the environment the system is moving to depends on the predicate, the destination ℓ' in the transition label is determined at top level.

For the discard transitions we can cover `Nsnd`, `Nget`, `Nsup` and `Nmov` at one. They all have the same premise $ch \notin \mathsf{LS} \cup \{\ell\} \vee ch = *$. As explained in Section 3.1, a process can discard an incoming message if it is not listening to the channel it is sent along, including the local channel and a broadcast message can always be discarded.
For the rule `Nrec` the premise is a bit more complicated. When a system is not listening to a channel it can discard as it won't receive the message. If it is sent over the broadcast channel, it should only discard if either of the predicates are not satisfied. If they are satisfied, `Rec` can be applied. In case ch and e' do not match, but the other predicates do, we have two separate cases. For multicast, this can mean that the process which is expected to receive, is still waiting to receive an other message over a different channel. For broadcast, and possibly also for multicast, this means that there is a mistake in the specification. This could be in the predicate, in configuration, or in the expected channel. This is something R-CHECK would find when model checking.

$$\texttt{Grd}\ \frac{\gamma \models \Pi \quad \langle \gamma, \mathsf{LS}, \ell \rangle : P \overset{\lambda}{\mapsto} \langle \gamma', \mathsf{LS}', \ell' \rangle : P'}{\langle \gamma, \mathsf{LS}, \ell \rangle : \langle \Pi \rangle P \overset{\lambda}{\mapsto} \langle \gamma', \mathsf{LS}', \ell' \rangle : P'} \qquad \texttt{Blk}\ \frac{ch \notin \mathsf{LS} \cup \{\ell\} \ \vee \ ch = * \quad \gamma \not\models \Pi}{\langle \gamma, \mathsf{LS}, \ell \rangle : \langle \Pi \rangle P \overset{\widetilde{\pi?(\tilde{v})^{ch}}}{\longmapsto} \langle \gamma, \mathsf{LS}, \ell \rangle : \langle \Pi \rangle P}$$

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

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

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

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

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

Table 3.3: The process-level semantics 2/2.

In Table 3.3 we can see that all, but `Nnul` depends on other transitions. We have explained what the notations stand for in Section 3.3, so we will keep this brief.

The guard works as explained previously, if a process can make the transition and the predicate of the guard holds, we can apply `Grd`. If a discard transition can be made by the process and the predicate holds, we can still also perform the discard transition by the rule `Stc`. In case the predicate does not hold, the process P cannot be accessed. Whether it can discard then depends on which channel it is sent over, just like for `Nsnd`, `Nget`, `Nsup` and `Nmov`. If it is a broadcast message or the system is not listening to the channel, `Blk` can be applied.
For the summation notation, if P_1 can transition, `Lor` can be applied. If P_2 can be applied, `Ror` can be applied. In case both are possible, a non-deterministic choice can be made and either of the rules can be applied. In case neither P_1 nor P_2 can transition, but they can discard, the discard transition can be made by applying `Nor`.
For `Def` and `Ndef` the transition will be made according to the possibility of $P[\tilde{v'}/\tilde{x}]$ with $K(\tilde{x}) \triangleq P$. If this can make a transition, `Def` can be applied, if it can discard `Ndef` can be applied.
Last we have the 0 process. As mentioned in 3.3, this process should not block other systems, but cannot transition. Therefore it has no premises and can always discard.

Example 3.4.1 *[Finished job (step 3/5)] We can now look at the process-level transitions from our example. The configurations are as follows:*
$\Gamma_M = \langle \gamma : id = m1, status = \text{"}Busy\text{"}, result = 3, manager = \bot, \mathsf{LS} = \{*, m1\}, \ell = 2 \rangle$
$\Gamma_Q = \langle \gamma : id = w1, status = \text{"}Ready\text{"}, result = 21, manager = m1, \mathsf{LS} = \{*, m1\}, \ell = 2 \rangle$
$\Gamma_R = \langle \gamma : id = w2, status = \text{"}Ready\text{"}, result = 19, manager = m1, \mathsf{LS} = \{*, m1\}, \ell = 2 \rangle$
We get the following process-level transitions for each of the systems, assuming Q supplies:
$\Gamma_M : M \xmapsto{(status=\text{"}Ready\text{"})\mathsf{G}(21)^{m1}} \{\!|\Gamma_D : [result := result + x]M[21/x]|\!\}$ *by* `Get`
$\Gamma_Q : Q \xmapsto{(status=\text{"}Ready\text{"})\mathsf{S}(21)^{m1}} \{\!|\Gamma_Q : H|\!\}$ *by* `Sup`
Because we are dealing with unicast, R does not respond. Only the receive transition can be discarded, so there is no transition to discard.

For the system-level we introduce one more transition label, the hidden transition τ. On a system-level this label is exposed when a unicast between two systems occurs. For the possible transition labels we then have:

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

We can see in Table 3.4 that on a system-level, the transitions are denoted by $\rightarrow$. The discard transition for a system is exposed like a normal receive label, meaning on a system-level we cannot immediately tell which systems receive a message and which do not.
We can now define the following system-level semantics:

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

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

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

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

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

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

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

$$\texttt{Lmov}\ \frac{S_1 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1'}{S_1 \mid\mid S_2 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1' \mid\mid S_2} \qquad \texttt{Rmov}\ \frac{S_2 \xrightarrow{\pi@(\Gamma:P,\ell)} S_2'}{S_1 \mid\mid S_2 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1 \mid\mid S_2'}$$

Table 3.4: The system-level semantics.

To explain the system-level semantics we will go row by row starting with `Sys` and `Nsys`. These are the rules connecting the process-level and system-level. They are performed on a single system as opposed to the other rules which are applied to a parallel composition of systems.
We know on broadcast and multicast channels multiple systems can receive the sent message therefore we have the rule `Prec`. Combined with `Nsys`, the receive transition can also be the discard transition, but we cannot see that on a system-level.
To show when a message is sent, another system will always receive or discard it, we have the rules `Lpar` and `Rpar`. As they are the only rules with a sending transition, they also show only a single message can be sent at a time. In addition, it also shows that on a system-level a sent message and the message being received by other systems happens in a single transition.
Both the combinations of `Lget` with `Rget` and `Lsup` with `Rsup` show only a single system can sent a get

request at a time and only a single system can supply a get request at a time.
The rules `Luni` and `Runi` show that when two systems in a parallel composition perform a unicast transition, this transition is hidden by the τ transition label. This way other systems will not be aware of the transition, and no system will have to perform a discard transition, which would not be possible on a get transition label.
For `Ltau` and `Rtau`, two rules which also state a unicast communication, we know the system performing this transition has to be a parallel composition of systems. The reason for this is that we do not have a process-level τ transition and therefore `Sys` or `Nsys` cannot be applied directly to said system.
Last we have `Lmov` and `Rmov`. These rules also show only a single system can move at a time. By having $\Gamma : P$ in the label, we know that even if the system transitioning is a parallel composition, only a single system in that composition can move.

The reason we have a left and a right side rule for almost each system-level transition on parallel composition, is to provide commutativity. This will also be proven with a lemma later on. We will assume that for an uneven number of systems, the left-hand side will have one more system in its parallel composition unless stated otherwise with parentheses.

Example 3.4.2 *[Finished job (step 4/5)] We can now look at the system-level transition from our example. If we take* $S_1 \triangleq \Gamma_M : M$, $S_2 \triangleq \Gamma_Q : Q$ *and* $S_3 \triangleq \Gamma_R : R$ *we get the following transition:*
$S_1 \mid\mid S_2 \mid\mid S_3 \xrightarrow{\tau} \Gamma'_M : M \mid\mid \Gamma'_Q : H \mid\mid S_3$ *by* `Ltau` *and* `Luni`

As we can see this is a lot clearer of an overview. But we do not know what M and Q communicated with each other, nor do we know how the configurations updated. This all depends on the specification, so if the specification is done correctly, we do not need this clutter of information.

Now that we have the systems inside the environment, we can add the reasoning about physical space. The top level of our calculus, the environment-level, will use a different transition arrow again, $\Rightarrow$. We will divide the semantics for the environment-level into two. The environment dependent transitions, which includes only the movement. And the communication semantics, which are only dependent on the systems inside of the environments.

Just like before we will use the transition label $\pi@(\Gamma : P, \ell)$ to model movements. Here π is the predicate set by the moving process, as described earlier in this section. This predicate has to be satisfied by RC. Before moving a sanity check is done on the environment to make sure the number of agents will not become negative and the capacity of the destination is checked to make sure there is space. This leaves us with the following environment-level semantics for movement:

$$\texttt{Out}\ \frac{S \xrightarrow{\pi@(\Gamma:P,\ell')} S' \quad \mathsf{RC}(\ell') = 1 \quad n > 0 \quad S \triangleq \Gamma : P \,||\, S''}{[S]^{\mathsf{RC},\ell}_{c,n} \xRightarrow{\pi@(\Gamma:P,\ell')} [S'']^{\mathsf{RC},\ell}_{c,n-1}}$$

$$\texttt{In}\ \frac{\Gamma : P \xrightarrow{\pi@(\Gamma:P,\ell)} \Gamma' : P' \quad \mathsf{RC} \models \pi \quad c > n}{[S]^{\mathsf{RC},\ell}_{c,n} \xRightarrow{\pi@(\Gamma:P,\ell)} [S \,||\, \Gamma' : P']^{\mathsf{RC},\ell}_{c,n+1}}$$

$$\texttt{Lex}\ \frac{E_1 \xRightarrow{\pi@(\Gamma:P,\ell)} E_1'}{E_1 \oplus E_2 \xRightarrow{\pi@(\Gamma:P,\ell)} E_1' \oplus E_2}$$

$$\texttt{Rex}\ \frac{E_2 \xRightarrow{\pi@(\Gamma:P,\ell)} E_2'}{E_1 \oplus E_2 \xRightarrow{\pi@(\Gamma:P,\ell)} E_1 \oplus E_2'}$$

$$\texttt{Cex}\ \frac{E_1 \xRightarrow{\pi@(\Gamma:P,\ell)} E_1' \quad E_2 \xRightarrow{\pi@(\Gamma:P,\ell)} E_2'}{E_1 \oplus E_2 \xRightarrow{\tau} E_1' \oplus E_2'}$$

Table 3.5: The environment-level semantics 1/2.

The first two rules `Out` and `In` are the base cases. Here a single environment loses or gains a single system, denoted by $\Gamma : P$. The environment that the system leaves checks the destination requested for its reachability connection RC. If the destination is externally connected, $\mathsf{RC}(\ell') = 1$, the move could be made if all other premises hold. A sanity check is also performed to ensure the environment is not left with a negative number of systems. The receiving environment checks if the predicate holds under its RC and checks if the room's capacity has not been reached yet.
Just like in the system-level semantics, we have the rules `Lex` and `Rex` to ensure only a single environment can be left at a time and only a single environment can be entered at a time. If both environments are in the environment composition, the move transition is hidden. We use the same label as for the hidden unicast transition, τ. This can be seen in the rule `Cex`.

Now for the communication on an environment-level we introduce a new set of transition labels, which is a subset of α. The only difference is that the movement label is missing, we will call this β:

$$\beta ::= \pi!(\tilde{v})^{ch} \quad | \quad \pi?(\tilde{v})^{ch} \quad | \quad \pi\mathsf{G}(\tilde{v})^{ch} \quad | \quad \pi\mathsf{S}(\tilde{v})^{ch} \quad | \quad \tau$$

We then get the following semantics:

$$\texttt{Com}\ \frac{S \xrightarrow{\beta} S'}{[S]^{\mathsf{RC},\ell}_{c,n} \overset{\beta}{\Longrightarrow} [S']^{\mathsf{RC},\ell}_{c,n}}$$

$$\texttt{Eprec}\ \frac{E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1' \quad E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'}{E_1 \oplus E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1' \oplus E_2'}$$

$$\texttt{Elpar}\ \frac{E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \quad E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'}{E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'} \qquad \texttt{Erpar}\ \frac{E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1' \quad E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'}{E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'}$$

$$\texttt{Elget}\ \frac{E_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1'}{E_1 \oplus E_2 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1' \oplus E_2} \qquad \texttt{Erget}\ \frac{E_2 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_2'}{E_1 \oplus E_2 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1 \oplus E_2'}$$

$$\texttt{Elsup}\ \frac{E_1 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_1'}{E_1 \oplus E_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_1' \oplus E_2} \qquad \texttt{Ersup}\ \frac{E_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_2'}{E_1 \oplus E_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_1 \oplus E_2'}$$

$$\texttt{Eluni}\ \frac{E_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1' \quad E_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_2'}{E_1 \oplus E_2 \overset{\tau}{\Rightarrow} E_1' \oplus E_2'} \qquad \texttt{Eruni}\ \frac{E_1 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_1' \quad E_2 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_2'}{E_1 \oplus E_2 \overset{\tau}{\Rightarrow} E_1' \oplus E_2'}$$

$$\texttt{Eltau}\ \frac{E_1 \overset{\tau}{\Rightarrow} E_1'}{E_1 \oplus E_2 \overset{\tau}{\Rightarrow} E_1' \oplus E_2} \qquad \texttt{Ertau}\ \frac{E_2 \overset{\tau}{\Rightarrow} E_2'}{E_1 \oplus E_2 \overset{\tau}{\Rightarrow} E_1 \oplus E_2'}$$

Table 3.6: The environment-level semantics 2/2.

The environment-level rules are similar to the system-level rules. The base case `Com` shows the communication transitions that can happen on systems inside an environment.
The other composition rules, `Eprec`, `Elpar`, `Erpar`, `Elget`, `Erget`, `Elsup`, `Ersup`,`Eluni` an `Eruni`, are similar to the system-level rules, but on an environment-level. This way the constraints on communications between systems are kept, which we will confirm in the lemmas. It also means that inter-environment communication can happen over shared channels.
The only difference is in `Eltau` and `Ertau`. Here the *tau* transition in the premise can also be `Cex`. But by excluding the movement transition label from β, a movement can still not happen inside an environment.

Now that we have the environment rules down we can finish the running example, by showing a system moving back to the hallway.

Example 3.4.3 *[Finished job (step 5/5)] We will first look at the environment-level transition from our example. If we take* $S_1 \triangleq \Gamma_M : M$, $S_2 \triangleq \Gamma_Q : Q$ *and* $S_3 \triangleq \Gamma_R : R$, *which we know are in the same environment* $\ell = 2$ *we get the following transition:*
$[S_1 \mid\mid S_2 \mid\mid S_3]^{\mathsf{RC},2}_{c,n} \overset{\tau}{\Rightarrow} [\Gamma'_M : M \mid\mid \Gamma'_Q : H \mid\mid S_3]^{\mathsf{RC},2}_{c,n}$ *by* `Com`

We will first define the moving process $H \triangleq \mathsf{MOV@}(\mathsf{RC}_1(1) = 0).I$. *On a process-level this will give the transition:*
$\Gamma'_Q : \mathsf{MOV@}(\mathsf{RC}_1(1) = 0).I \xmapsto{\pi@(\Gamma'_Q:H,1)} \{\!|\Gamma'_Q : [\ell \mapsto 1]I|\!\}$ *by* `Mov`
On a system-level this will look like:
$\Gamma'_Q : H \xrightarrow{\mathsf{MOV@}(\mathsf{RC}_1(1)=0)(\Gamma'_Q:H,1)} \Gamma''_Q : I$ *by* `Sys`

We will specify the two environments as follows $E_1 \triangleq [\quad]^{\mathsf{RC}_1,1}_{5,0}$ *and* $E_2 \triangleq [\Gamma'_M : M \mid\mid \Gamma'_Q : H \mid\mid S_3]^{\mathsf{RC}_2,2}_{4,3}$. *We have* $\mathsf{RC}_1 = \{1 \mapsto 0, 2 \mapsto 1\}$ *and* $\mathsf{RC}_2 = \{1 \mapsto 1, 2 \mapsto 0\}$
We will get the following environment-level transition:
$E_1 \oplus E_2 \overset{\tau}{\Rightarrow} [\Gamma''_Q : I]^{\mathsf{RC}_1,1}_{5,1} \oplus [\Gamma'_M : M \mid\mid S_3]^{\mathsf{RC}_2,2}_{4,2}$ *by* `Cex`

3.5 Properties of semantics

We will first state some essential lemmas, to prove certain behavioral properties on a system-level. Next we will state some lemmas on an environment-level. The proofs of these lemmas can be found in the appendix.

Lemma 3.5.1 ***Lemma 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. $S_1 \mid\mid \Gamma : 0 \equiv S_1$

Lemma 3.5.2 ***Lemma 2*** *is regarding the non-blocking behavior of broadcasted messages. Given 2 systems* S_1 *and* S_2 *prove:*
$ch = * \wedge S_1 \xrightarrow{\pi!(\tilde{v})^{ch}} S'_1 \implies S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S'_1 \mid\mid S'_2$ *for any* S_2.

Lemma 3.5.3 ***Lemma 3*** *shows the blocking of multicast channels. Given 2 systems* S_1 *and* S_2 *prove:*
$ch \neq * \wedge S_1 \xrightarrow{\pi!(\tilde{v})^{ch}} S'_1 \implies S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S'_1 \mid\mid S'_2$ *for any* S_2 *such that:*

1. $ch \in \mathsf{LS} \cup \{\ell\} \wedge S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S'_2$
   $\vee$
2. $ch \notin \mathsf{LS} \cup \{\ell\}$

Lemma 3.5.4 ***Lemma 4*** *is about the blocking of unicast, to show it should never go unanswered. Given 2 systems S_1 and S_2 prove:*
$S_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_1' \implies S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$ *for any S_2 such that $S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2'$.*

Lemma 3.5.5 ***Lemma 5***

1. $E_1 \oplus E_2 \equiv E_2 \oplus E_1$
2. $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$

Note that we are missing the third point that we have in Lemma 3.5.1 because an empty environment can still be entered after, unlike how an empty process cannot proceed.

Lemma 3.5.6 ***Lemma 6*** *is about the blocking of moving in an environment composition. It shows that when a system leaves an environment, it always enters an other externally connected environment. It also shows a movement is allowed by an environment, a process can only request it. Given 2 environments E_1 and E_2 prove:*
For $E_1 = [S_1]^{\mathsf{RC}_1,\ell_1}_{c_1,n_1}$ and $E_2 = [S_2]^{\mathsf{RC}_2,\ell_2}_{c_2,n_2}$:
$E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1' \implies E_1 \oplus E_2 \xRightarrow{\tau} E_1' \oplus E_2'$ *for any E_2 such that: $\ell = \ell_2,\ n_1 > 0,\ c_2 > n_2,\ \mathsf{RC} \models \pi$ and $\mathsf{RC}_1(\ell) = 1 \wedge E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2'$*

Lemma 3.5.7 ***Lemma 7*** *is regarding the non-blocking behavior of broadcasted messages. Given 2 environments E_1 and E_2 prove:*
$ch = * \wedge E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \implies E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \mid\mid E_2'$ *for any E_2.*

Lemma 3.5.8 ***Lemma 8*** *shows the blocking of multicast channels. Given 2 environments E_1 and E_2 prove:*
$ch \neq * \wedge E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \implies E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$ *for any E_2 such that for all S_i in E_2:*

1. $ch \in \mathsf{LS}_i \cup \{\ell\}_i \wedge E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$
   $\vee$
2. $ch \notin \mathsf{LS}_i \cup \{\ell\}_i$

Lemma 3.5.9 ***Lemma 9*** *is about the blocking of unicast, to show it should never go unanswered. Given 2 environments E_1 and E_2 prove:*
$E_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_1' \implies E_1 \oplus E_2 \xRightarrow{\tau} E_1' \mid\mid E_2'$ *for any E_2 such that $E_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_2'$.*

4

Case studies

To show the extent of application of our developed system, we will give some smaller examples and combine these into one bigger example. Beginning with a process level, we will look at a case for each of the following behaviors: broadcast, multicast, unicast, move. For the last case, it will be mostly defining the process and system, as the moving happens on an environment level.

4.1 Process and System Level

To keep the examples clear, we will use at most three processes per example, with a single system level transition. Later on, for the full application, some of these can be combined and expanded to get an idea of the application of our calculus.

4.1.1 Broadcast

For this example we will take a broadcasting process B, a receiving process R and a process D, which shows the non-blocking behavior of broadcasting. The broadcaster wants all non-busy systems (denoted by status $\mapsto 0$), to terminate if they have reached the capacity. The specifications are then as follows:
$B \triangleq (status = 0)!(\text{"}Terminate\text{"}, cap)^*.0$
$R \triangleq ((x = \text{"}Terminate\text{"}) \wedge (y \geq cap))?(x,y)^*.0$
$D \triangleq ((x = \text{"}Terminate\text{"}) \wedge (y \geq cap))?(x,y)^*.0$

We know all configurations share the same attributes, so the configuration is as follows:
$\Gamma = \langle \gamma : \{status, cap\}, \mathsf{LS} = \{*\}, \ell = 5 \rangle$ for all three processes with the following attribute values:
$\gamma_B(status) = \gamma_R(status) = \gamma_D(status) = 0$,
$\gamma_B(cap) = \gamma_R(cap) = 3$ and $\gamma_D(cap) = 2$.

On a process level we then get the following transition for B by applying the rule `Snd`:
$\Gamma_B : B \xmapsto{(status=0)!(\text{"}Terminate\text{"},3)^*} \{\!|\Gamma_B : 0|\!\}$
For R we get the following transition by applying the rule `Rec`:
$\Gamma_R : R \xmapsto{(status=0)?(\text{"}Terminate\text{"},3)^*} \{\!|\Gamma_R : 0[\text{"}Terminate\text{"}/x, 3/y]|\!\}$
And for R we get the following transition by applying the rule `Nrec`:

$\Gamma_D : D \xmapsto{(status=0)?(\widetilde{"Terminate"},3)^*} \Gamma_D : D$

If we put these systems together in a parallel composition, with $S_1 \triangleq \Gamma_B : B$, $S_2 \triangleq \Gamma_R : R$, $S_3 \triangleq \Gamma_D : D$, we get $S_1 \mid\mid S_2 \mid\mid S_3$. Assuming for uneven numbers we take one extra system on the left hand side, we get the following transition by applying the rule `Lpar` twice:
$S_1 \mid\mid S_2 \mid\mid S_3 \xrightarrow{(status=0)!("Terminate",3)^*} \Gamma'_B : 0 \mid\mid \Gamma'_R : 0 \mid\mid S_3$

4.1.2 Multicast

For the multicast example we will take similar processes to the broadcasting example: A sending process M, a receiving process R, and a process D. In this case D will not receive the message from M, however as multicast is blocking, the reason will be that D is not listening to the multicast channel as it is part of a different team. The scenario we will portray is one of a team of processes, which have completed a task and are ready to go back to idle. We will denote an idle process by I, but for this example the specification of I is not important. The specifications are then as follows:
$M \triangleq (status = 2)!("Complete", 0)^{id}.[status := 0]I$
$R \triangleq (x = "Complete")?(x,y)^{manager}.[status := y]I$
$D \triangleq (x = "Complete")?(x,y)^{manager}.[status := y]I$

We know all configurations share the same attributes, so the configuration is as follows:
$\Gamma = \langle \gamma : \{id, status, manager\}, \mathsf{LS}, \ell = 5 \rangle$ for all three processes with the following attribute values:
$\gamma_M(id) = 1$, $\gamma_R(id) = 2$, $\gamma_D(id) = 3$,
$\gamma_M(status) = \gamma_R(status) = \gamma_D(status) = 2$,
$\gamma_M(manager) = \perp$, $\gamma_R(manager) = 1$ and $\gamma_D(manager) = 4$.
For the set of listening channels we get $\mathsf{LS}_M = \{*, 1\}$, $\mathsf{LS}_R = \{*, 1\}$ and $\mathsf{LS}_D = \{*, 4\}$.

On a process level we then get the following transition for M by applying the rule `Snd`:
$\Gamma_M : M \xrightarrow{(status=2)!("Complete",0)^1} \{\!|\Gamma_M : [status := 0]I|\!\}$
For R we get the following transition by applying the rule `Rec`:
$\Gamma_R : R \xmapsto{(status=2)?("Complete",0)^1} \{\!|\Gamma_R : [status := y]I["Complete"/x, 0/y]|\!\}$
And for R we get the following transition by applying the rule `Nrec`:
$\Gamma_D : D \xmapsto{(status=2)?(\widetilde{"Complete"},0)^1} \Gamma_D : D$

If we put these systems together in a parallel composition, with $S_1 \triangleq \Gamma_M : M$, $S_2 \triangleq \Gamma_R : R$, $S_3 \triangleq \Gamma_D : D$, we get $S_1 \mid\mid S_2 \mid\mid S_3$. Assuming for uneven numbers we take one extra system on the left hand side, we get the following transition by applying the rule `Lpar` twice:
$S_1 \mid\mid S_2 \mid\mid S_3 \xrightarrow{(status=2)!("Complete",0)^1} \Gamma'_M : I \mid\mid \Gamma'_R : I \mid\mid S_3$

4.1.3 Unicast

For the unicast example we will take three processes again: A requesting process F, a supplying process O, and a process D. In this case D will not supply the get request from F, but it would be able to. This is to show that only a single process supplies. The scenario we will portray is one of a team of processes, which have completed a task and are ready to supply their results and go back to idle. We will denote an idle process by I again. The specifications are then as follows:
$F \triangleq (status = 2)\mathsf{G}(x)^{id}.[profit := profit + x]F$
$O \triangleq \mathsf{S}(profit)^{manager}.[status := 0]I$
$D \triangleq \mathsf{S}(profit)^{manager}.[status := 0]I$

We know all configurations share the same attributes, so the configuration is as follows:
$\Gamma = \langle \gamma : \{id, status, manager, profit\}, \mathsf{LS} = \{*, 1\}, \ell = 5\rangle$ for all three processes with following attribute values:
$\gamma_F(id) = 1$, $\gamma_O(id) = 2$, $\gamma_D(id) = 3$,
$\gamma_F(status) = 1$, $\gamma_O(status) = \gamma_D(status) = 2$,
$\gamma_M(manager) = \perp$, $\gamma_R(manager) = \gamma_D(manager) = 1$ and
$\gamma_F(profit) = 0$, $\gamma_O(profit) = 5$ and $\gamma_D(profit) = 3$.

On a process level we then get the following transition for F by applying the rule `Get`:
$\Gamma_F : F \xmapsto{(status=2)\mathsf{G}(5)^1} \{\!|\Gamma_F : [profit := profit + x]F|\!\}$
For O we get the following transition by applying the rule `Sup`:
$\Gamma_O : O \xmapsto{(status=2)\mathsf{S}(5)^1} \{\!|\Gamma_O : [status := o]I|\!\}$
And for R we do not get a transition as only 1 process can supply at a time.

If we put these systems together in a parallel composition, with $S_1 \triangleq \Gamma_F : F$, $S_2 \triangleq \Gamma_O : O$, $S_3 \triangleq \Gamma_D : D$, we get $S_1 \mid\mid S_2 \mid\mid S_3$. Assuming for uneven numbers we take one extra system on the left hand side, we get the following transition by applying the rule `Ltau` and `Luni`:
$S_1 \mid\mid S_2 \mid\mid S_3 \xrightarrow{\tau} \Gamma'_F : F \mid\mid \Gamma'_R : I \mid\mid S_3$

4.2 Environment Level

In this section we will show an example of a system moving between two externally connected environments. The example will not cover communication, but we will implement this in the larger example at the end.

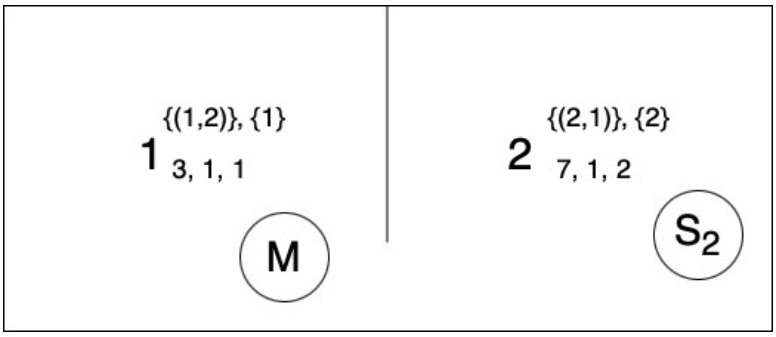

Figure 4.1: Example of space layout with configurationd and systems portrayed.

As only a single system can move at a time, we only have one process to define, let's call this process M and an undefined system S_2 in the waiting room. To keep this example simple, the environment composition will consist of two environments. The process is done performing its job and wants to move back to the waiting room to wait for its next assignment as an idle process. The specification is then as follows:
$M \triangleq \mathsf{MOV}@(\mathsf{RC}(loc) < \ell.\mathsf{RC}(loc)).M + \langle \ell = loc \rangle I$

The configuration of M is specified as: $\Gamma_M = \langle \gamma : \{id, loc\}, \mathsf{LS} = \{*\}, \ell = 1 \rangle$, with $\gamma_M(id) = 5$ and $\gamma_M(loc) = 2$.

On a process level we then get the following transition for M by applying the rule `Mov`:
$\Gamma_M : M \xrightarrow{(\mathsf{RC}(2)<1)@(\Gamma_M:M,2)} \{\!|\Gamma_M : [\ell \mapsto 2].M|\!\}$

If we take $S_1 \triangleq \Gamma_M : M$, on a system level we then get the following transition for M by applying the rule `Sys`:
$S_1 \xrightarrow{(\mathsf{RC}(2)<1)@(\Gamma_M:M,2)} \Gamma'_M : M$

This brings us to the specifications of our environments, which can be seen in 4.1. We will write them down formally for clarification:
$E_1 \triangleq [\Gamma_M : M]^{\mathsf{RC}_1,\ell_1}_{c_1,n_1}$
$E_2 \triangleq [S_2]^{\mathsf{RC}_2,\ell_2}_{c_2,n_2}$

The configurations of the environments can also be deducted from 4.1 and are:
$\mathsf{RC}_1 = \{1 \mapsto 0, 2 \mapsto 1\}$, $\mathsf{RC}_2 = \{1 \mapsto 1, 2 \mapsto 0\}$,
$\ell_1 = 1$ and $\ell_2 = 2$.
$c_1 = 3$, $c_2 = 7$,
$n_1 = 1$, $n_2 = 1$,

On an environment level we get the following transitions for E_1 by applying the rule `Out`:
$[S_1]^{\mathsf{RC}_1,1}_{3,1} \xrightarrow{(\mathsf{RC}(2)<1)@(\Gamma:M,2)} [\quad]^{\mathsf{RC}_1,1}_{3,1-1}$
And for E_2 we get the following transition by applying `In`:
$[S_2]^{\mathsf{RC}_2,2}_{7,1} \xrightarrow{(\mathsf{RC}(2)<1)@(\Gamma:M,2)} [S_2 \mid\mid \Gamma'_M : I]^{\mathsf{RC}_1,2}_{7,1+1}$

If we put the two environments together in a composition as $E_1 \oplus E_2$, we can apply `Cex` to get the

following transition:
$E_1 \oplus E_2 \stackrel{\tau}{\Rightarrow} E_1' \oplus E_2'$

4.3 Case

To back up the relevance of the results, a more well rounded case will be presented. We will take the same space layout as in Figure 3.1. To keep the case uncluttered, only three systems inside the space will be present. These systems are called A, st_1 and st_2. Their current locations can be seen in 4.2. Here both st are a stationary system, presented by a dot, and A is a mobile system, presented by an open node with the system inside. The setting will highlight how even without being aware of the layout, a system can reach destinations in an efficient way using the reachability connections stored in the environment. It will also show how we can model the case when a broadcast message only expects a single response, but possibly multiple systems can answer. The processes of the system and its transitions will be described step-by-step as to not overcrowd the specifications.

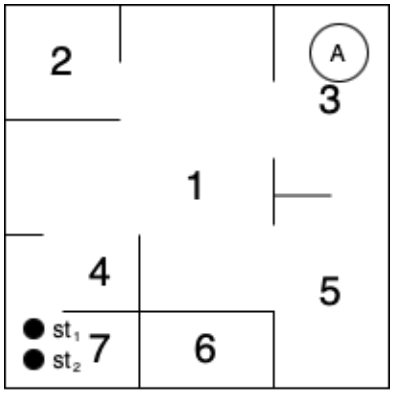

Figure 4.2: Systems inside the space.

The scenario is as follows: System A has a low battery level and needs a charging station. In its current environment, no charging station is present. Rather than exploring environments and checking whether there is a station available, A broadcasts a message to inquire a charging station's location. The charging stations st received this message and one of them answers to let A know where it is located. This is done over A's private link which is sent along, and in this case is the same as its unique identifier. When A receives this information it starts heading to the environment where the charging station is located.

The processes of A and st are defined like this:

$$
\begin{aligned}
A \triangleq \quad & \langle batlvl < 30 \rangle \Big((role = \text{"}station\text{"})!(\text{"}Charge\text{"}, id)^*.A \\
& + (x = \text{"}Ack\text{"})?(x,y)^{id}.[loc := y]M \Big) \\
st_1 \triangleq st_2 \triangleq \quad & (x = \text{"}Charge\text{"})?(x,y)^*.[lnk := y]W \\
W \triangleq \quad & \Big((true)!(\text{"}Ack\text{"}, \ell)^{lnk}.[lnk := \bot]st_1 \\
& + (x = \text{"}Ack\text{"})?(x,y)^{lnk}.[lnk := \bot]st_1 \Big)
\end{aligned}
$$

The reason we have a summation for A, of which the send action moves back to A, is so that the system can send a request until it is answered and it will not stop looking for an available charging station until it has one. When it has found one, it continues with M, which is when the systems start to move. This process will be defined together with the environment later.
For the station processes $st_1 \triangleq st_2$, we also have a summation notation. As all available charging stations receive the request for their location, they will all try and send it to A over its private multicast channel. If we recall, multicast is blocking, so the summation makes sure that any station that does not respond first, does not try to respond later.
The configurations of the systems are:
$\Gamma_A = \langle \{id \mapsto A, role \mapsto \text{"}worker\text{"}, batlvl \mapsto 25, loc \mapsto 3, lnk \mapsto \bot\}, \{*, A, lnk\}, 3 \rangle$
$\Gamma_{st_1} = \langle \{id \mapsto st1, role \mapsto \text{"}station\text{"}, batlvl \mapsto 75, loc \mapsto \bot, lnk \mapsto \bot\}, \{*, st1, lnk\}, 7 \rangle$
$\Gamma_{st_2} = \langle \{id \mapsto st2, role \mapsto \text{"}station\text{"}, batlvl \mapsto 85, loc \mapsto \bot, lnk \mapsto \bot\}, \{*, st2, lnk\}, 7 \rangle$

The process level transitions are then in the following order:
$\Gamma_A : A \xmapsto{(role=\text{"}station\text{"})!(\text{"}Charge\text{"},A)^*} \Gamma_A : A$ by `Snd`

$\Gamma_{st_1} : st_1 \xmapsto{(role=\text{"}station\text{"})?(\text{"}Charge\text{"},A)^*} \Gamma'_{st_1} : W[\text{"}Charge\text{"}/x, A/y]$ by `Rec`

$\Gamma_{st_2} : st_2 \xmapsto{(role=\text{"}station\text{"})?(\text{"}Charge\text{"},A)^*} \Gamma'_{st_2} : W[\text{"}Charge\text{"}/x, A/y]$ by `Rec`

$\Gamma'_{st_1} : W \xmapsto{(true)!(\text{"}Ack\text{"},7)^A} \Gamma_{st_1} : st_1[\text{"}Charge\text{"}/x, A/y]$ by `Snd`

$\Gamma'_{st_2} : W \xmapsto{(true)?(\text{"}Ack\text{"},7)^A} \Gamma_{st_2} : st_2[\text{"}Ack\text{"}/x, 7/y]$ by `Rec`

$\Gamma_A : A \xmapsto{(true)?(\text{"}Ack\text{"},7)^A} \Gamma'_A : M[\text{"}Ack\text{"}/x, 7/y]$ by `Rec`

Assuming $S_1 \triangleq \Gamma_A : A$, $S_2 \triangleq \Gamma_{st_1} : st_1$ and $S_3 \triangleq \Gamma_{st_2} : st_2$ with $S_1 \parallel S_2 \parallel S_3$. On a system level this gives the following two transitions:
$S_1 \parallel S_2 \parallel S_3 \xrightarrow{(role=\text{"}station\text{"})!(\text{"}Charge\text{"},A)^*} S'_1 \parallel S'_2 \parallel S'_3$ by `Lpar` twice

$S'_1 \parallel S'_2 \parallel S'_3 \xrightarrow{(true)!(\text{"}Ack\text{"},7)^A} S''_1 \parallel S''_2 \parallel S''_3$ by `Lpar` and `Rpar`

With $S'_1 \triangleq \Gamma_A : A$, $S'_2 \triangleq \Gamma'_{st_1} : W[\text{"}Charge\text{"}/x, A/y]$, $S'_3 \triangleq \Gamma'_{st_2} : W[\text{"}Charge\text{"}/x, A/y]$,
$S''_1 \triangleq \Gamma'_A : M[\text{"}Ack\text{"}/x, 7/y]$, $S''_2 \triangleq \Gamma_{st_1} : st_1[\text{"}Charge\text{"}/x, A/y]$ and $S''_3 \triangleq \Gamma_{st_2} : st_2[\text{"}Ack\text{"}/x, 7/y]$

This brings us to the environment level. First environment 3 and 7 will be defined so we can provide

the communication transitions:

$$E_3 \triangleq \ [S_1]^{\mathsf{RC}_3,3}_{3,1}$$
$$E_7 \triangleq \ [S_2 \mid\mid S_3]^{\mathsf{RC}_7,7}_{4,2}$$

$\mathsf{RC}_3 = \{1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 0, 4 \mapsto 2, 5 \mapsto 1, 6 \mapsto \infty, 7 \mapsto 3\}$
$\mathsf{RC}_7 = \{1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 3, 4 \mapsto 1, 5 \mapsto 3, 6 \mapsto \infty, 7 \mapsto 0\}$

We get the following transitions:
$[S_1]^{\mathsf{RC}_3,3}_{3,1} \oplus [S_2 \mid\mid S_3]^{\mathsf{RC}_7,7}_{4,2} \xrightarrow{(role=\text{"}station\text{"})!(\text{"}Charge\text{"},A)^*} [S'_1]^{\mathsf{RC}_3,3}_{3,1} \oplus [S'_2 \mid\mid S'_3]^{\mathsf{RC}_7,7}_{4,2}$ by `Elpar`

$[S'_1]^{\mathsf{RC}_3,3}_{3,1} \oplus [S'_2 \mid\mid S'_3]^{\mathsf{RC}_7,7}_{4,2} \xrightarrow{(true)!(\text{"}Ack\text{"},7)^A} [S''_1]^{\mathsf{RC}_3,3}_{3,1} \oplus [S''_2 \mid\mid S''_3]^{\mathsf{RC}_7,7}_{4,2}$ by `Erpar`

We can see in Figure 4.2 the fastest way for system A to go would be through environment 1 and 4. This means we will have to specify this as well. We are dealing with 6 environment composition, $E_1 \oplus E_2 \oplus E_3 \oplus E_4 \oplus E_5 \oplus E_6 \oplus E_7$. For the other environments, we will only provide $\mathsf{RC}(7)$, as this will be essential to decide how the system moves based on the predicate.

$$E_1 \triangleq \ [\ \ ]^{\mathsf{RC}_1,1}_{8,0}$$
$$E_4 \triangleq \ [\ \ ]^{\mathsf{RC}_4,4}_{3,0}$$

$\mathsf{RC}_1 = \{1 \mapsto 0, 2 \mapsto 1, 3 \mapsto 1, 4 \mapsto 1, 5 \mapsto 1, 6 \mapsto \infty, 7 \mapsto 2\}$.
$\mathsf{RC}_2(7) = \mathsf{RC}_5(7) = 3$ and $\mathsf{RC}_6(7) = \infty$

No we can provide the specification of M:

$$M \triangleq \ \mathsf{MOV}@(\ell.\mathsf{RC}(loc) > \mathsf{RC}(loc)).M \\ +\langle \ell = loc \rangle C$$

Here the summation includes the movement and the charging process, which will stay undefined. If an environment satisfies the predicate but is not externally connected, the environment will block this option based on the premises in `Out`. This leaves only the moves to the externally connected environments. The predicate sent along, shows that only environments with a reachability connection closer than the system's current environment can receive the system, meaning system A will always move closer.
Now we get the final transitions to complete the case:

$[\ \ ]^{\mathsf{RC}_1,1}_{8,0} \oplus [\Gamma'_A : M]^{\mathsf{RC}_3,3}_{3,1} \overset{\tau}{\Rightarrow} [\Gamma''_A : M]^{\mathsf{RC}_1,1}_{8,1} \oplus [\ \ ]^{\mathsf{RC}_3,3}_{3,0}$ by `Cex`

$[\Gamma''_A : M]^{\mathsf{RC}_1,1}_{8,1} \oplus [\ \ ]^{\mathsf{RC}_4,4}_{3,0} \overset{\tau}{\Rightarrow} [\ \ ]^{\mathsf{RC}_1,1}_{8,0} \oplus [\Gamma'''_A : M]^{\mathsf{RC}_4,4}_{3,1}$ by `Cex`

$[\Gamma'''_A : M]^{\mathsf{RC}_4,4}_{3,1} \oplus [S''_2 \mid\mid S''_3]^{\mathsf{RC}_7,7}_{4,2} \overset{\tau}{\Rightarrow} [\ \ ]^{\mathsf{RC}_4,4}_{3,0} \oplus [S''_2 \mid\mid S''_3 \mid\mid \Gamma''''_A : M]^{\mathsf{RC}_7,7}_{4,3}$ by `Cex`

Note that the environments which are not in the composition in the transitions still exist. We omitted

them to keep it clear. The transition label is hidden, but we have a completed move. In the specification of M, we see that the predicate $(\ell.\mathsf{RC}(loc) > \mathsf{RC}(loc))$ will de concretized to $(3 > \mathsf{RC}(7))$ the first transition, $(2 > \mathsf{RC}(7))$ the second transition, and finally $(1 > \mathsf{RC}(7))$ the last transition. It shows that it can only move to an environment which is closer to the destination than the current environment. This is also the reason we did not specify E_2, E_5 and E_6 further. We can see in the provided values, that this predicate does not hold for them.
The predicate for the move action would then become $(0 > \mathsf{RC}(7))$, which will never hold as this is only the case when the system is in the environment itself. The guard before the charging process C, $\langle \ell = loc \rangle$, then holds, meaning it can continue with that process.
What this case study has shown is a concrete example of how the LaC-calculus extends the functionality of R-CHECK. The scenario shows how systems can communicate over broadcast and multicast. It also shows how a system can make moves within physical space, based on its configuration, even without having direct knowledge of the other environments. In Section 4.1.3, we can see how the extension of unicast works. These are useful functionalities to model, as it enables the systems inside a physical space, to organize themselves.

5

Conclusion

In this chapter, we will summarize the results presented in this report, mention which improvements have been made, and finally discuss possible future work.

5.1 Summary

Starting from R-CHECK, we provided the LaC-calculus. The calculus provides ways to model interactions between multiple systems. It provides communication modelling possibilities for broadcast, multicast and unicast messages. In addition, it provides a way to model movements of systems. These actions are modelled by a system's configurations and the processes it consists of.

The physical space in which these systems exist, called environments, can be modelled with their own configuration. These configurations influence the systems inside of it, by providing a local communication channel and restricting movements based on possibilities.

We proved some important properties of our semantics, like the blocking of multicast, unicast and movement and the non-blocking of broadcast.

We provided multiple examples to show how the LaC-calculus and its semantics can be applied, including an extended real-life example to highlight its functionality.

5.2 Future work

For future work, the physical space can be made continuous. Currently only the systems inside the environments can move, but for other applications, such as self-driving cars, moving environments could be used. This would require reasoning about how the reachability connections change as the environments move. In addition, the movement of environments should be split from the communication of the systems inside. The reason for this is that systems inside of a moving environment, like in the example of a self-driving car, should not halt their processes during the movements of environments.

The current way of modeling can result in a whole system getting blocked if an agent stops working. Depending on the modeled system, this might be unwanted. For future work, possible protocols to deal with this problem could be explored. This will help to come closer to real-life scenarios.

The LaC-calculus has not been implemented into R-CHECK yet, which is also something to be done

in the future. This way, the models made with this calculus could be checked for their correctness. In addition, by model testing potential problems could arise. This could require some changes to be made to the calculus. It would also make it possible to use the LaC-calculus in applications, as the specifications of the models can be confirmed to be correct.

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.

[22] K. P. Seng, L. M. Ang, and E. Ngharamike, "Artificial intelligence internet of things: A new paradigm of distributed sensor networks," *International Journal of Distributed Sensor Networks*, vol. 18, no. 3, p. 15 501 477 211 062 835, 2022.

[23] A. Bucchiarone, M. D'Angelo, D. Pianini, *et al.*, "On the social implications of collective adaptive systems," *IEEE Technology and Society Magazine*, vol. 39, no. 3, pp. 36–46, 2020.

A

Appendix 1

A.1 Lemma 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 prove this lemma by case analysis on α. We will have 6 cases with multiple subcases each. To prove 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: $\pi!(\tilde{v})^{ch}$, $\pi?(\tilde{v})^{ch}$, $\pi\mathsf{G}(\tilde{v})^{ch}$, $\pi\mathsf{S}(\tilde{v})^{ch}$, τ and $\pi@(\Gamma : P, \ell)$
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 $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$.
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 $S_2 = \langle \gamma, \mathsf{LS}, \ell \rangle : P$ and $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$. 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 $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$ for $S_2 = \langle \gamma, \mathsf{LS}, \ell \rangle : P$ 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 $S_1 = \langle \gamma, \mathsf{LS}, \ell \rangle : P$ and $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$. 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 $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$ with $S_1 = \langle \gamma, \mathsf{LS}, \ell \rangle : P$ 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 $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$. 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 $S_2 = \langle \gamma, \mathsf{LS}, \ell \rangle : P$, we get $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$. 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 $S_1 = \langle \gamma, \mathsf{LS}, \ell \rangle : P$, we get $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$. 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 $S_1 = \langle \gamma, \mathsf{LS}, \ell \rangle : P_1$ and $S_2 = \langle \gamma, \mathsf{LS}, \ell \rangle : P_2$, we get $\langle \gamma, \mathsf{LS}, \ell \rangle : P_1 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_1$ and $\langle \gamma, \mathsf{LS}, \ell \rangle : P_2 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_2$. 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 $S_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_1'$, resulting in $S_2 \mid\mid S_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_2 \mid\mid S_1'$, 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 $S_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_2'$, resulting in $S_1 \mid\mid S_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_1 \mid\mid S_2'$. By `Lget` on $S_2 \mid\mid S_1$, we also get $S_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_2'$, resulting in $S_2 \mid\mid S_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_2' \mid\mid S_1$, 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 $\alpha = \pi \mathsf{S}(\tilde{v})^{ch}$.
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 $S_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_1'$, resulting in $S_1 \mid\mid S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_1' \mid\mid S_2$. By `Rsup` on $S_2 \mid\mid S_1$, we also get $S_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_1'$, resulting in $S_2 \mid\mid S_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2 \mid\mid S_1'$, 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 $S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2'$, resulting in $S_1 \mid\mid S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_1 \mid\mid S_2'$. By `Lsup` on $S_2 \mid\mid S_1$, we also get $S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2'$, resulting in $S_2 \mid\mid S_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2' \mid\mid S_1$, 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 $S_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2'$ resulting in $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$. By `Runi`, we also get $S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2'$ and $S_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_1'$ 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 $S_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_2'$ resulting in $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$. By `Luni`, we also get $S_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_2'$ and $S_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_1'$ 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.

Case 6: Consider $\alpha = \pi@(\Gamma : P, \ell)$.
Subcase 6.1: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2'$. We know only one process can move at a time. This means only a single system can actually transition, and this subcase is not possible.
Subcase 6.2: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1' \mid\mid S_2$. In this case only S_1 transitions. We consider the two move transitions.
Subcase 6.2.1: Consider `Lmov`, we get $S_1 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1'$. This results in $S_1 \mid\mid S_2 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1' \mid\mid S_2$. By `Rmov` on $S_2 \mid\mid S_1$ we then get $S_1 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1'$ as well. This results in $S_2 \mid\mid S_1 \xrightarrow{\pi@(\Gamma:P,\ell)} S_2 \mid\mid S_1'$. Showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$.
Subcase 6.2.2: Consider `Rmov`, we would get S_2 with the moving 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 6.3: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2'$. In this case we have the opposite of the previous case.
Subcase 6.2.1: Consider `Lmov`, we would get S_1 with the moving transition, but by our IH, we have that S_1 does not transition. By `Nsys`, we see that it is only possible for the receiving, and thus this is not possible.
Subcase 6.2.2: Consider `Rmov`, we get $S_2 \xrightarrow{\pi@(\Gamma:P,\ell)} S_2'$. This results in $S_1 \mid\mid S_2 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1 \mid\mid S_2'$. By `Lmov` on $S_2 \mid\mid S_1$ we then get $S_2 \xrightarrow{\pi@(\Gamma:P,\ell)} S_2'$ as well. This results in $S_2 \mid\mid S_1 \xrightarrow{\pi@(\Gamma:P,\ell)} S_2' \mid\mid S_1$. Showing $S_1 \mid\mid S_2 \equiv S_2 \mid\mid S_1$.
Subcase 6.4: $S_1 \mid\mid S_2 \xrightarrow{\alpha} S_1 \mid\mid S_2$, this case is a combination of subcases 6.2.2 and 6.3.1. For a moving transition at least one system has to transition, as it cannot be discarded, hence this is not possible.
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 $\alpha = \pi\mathsf{S}(\tilde{v})^{ch}$.
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 $S_1 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S_1'$. By `Lsup` on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_1 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S_1'$. 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 $S_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S_2'$.

By Rsup and Lsup on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2'$. 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 $S_3 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_3'$. By Rsup twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_3 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_3'$. 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 $S_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2'$, 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 $S_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2'$ 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 $S_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_2'$, 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 $S_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_1'$ and $S_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_2'$ 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 $S_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_1'$ and $S_3 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_3'$, 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 $S_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_1'$ and $S_3 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_3'$ 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 $S_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_1'$ and $S_3 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_3'$, 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 $S_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_1'$ and $S_3 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_3'$ 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 $S_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_3'$, 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 $S_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_3'$ 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 $S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_3'$, 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 $S_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} S_2'$ and $S_3 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} S_3'$ 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.

Case 6: Consider $\alpha = \pi@(\Gamma : P, \ell)$.

Subcase 6.1: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3'$. We know only one system can move at a time. This means for this α only 1 system can transition and thus this subcase is not possible.

Subcase 6.2: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3$. As we know a system has to transition to move, we know S_1 has to be the moving system. The only way to get this is by applying Lmove twice. We get $S_1 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1'$. By Lmov on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_1 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1'$. 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 6.3: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3$. As we know a system has to transition to move, we know S_2

has to be the moving system. The only way to get this is by applying Lmov and Rmove. We get $S_2 \xrightarrow{\pi@(\Gamma:P,\ell)} S_2'$. By Rmov and Lmov on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_2 \xrightarrow{\pi@(\Gamma:P,\ell)} S_2'$. 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 6.4: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3'$. As we know a system has to transition to move, we know S_3 has to be the moving system. The only way to get this is by applying Rmov. We get $S_3 \xrightarrow{\pi@(\Gamma:P,\ell)} S_3'$. By Rmov twice on $S_1 \mid\mid (S_2 \mid\mid S_3)$, we also get $S_3 \xrightarrow{\pi@(\Gamma:P,\ell)} S_3'$. 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 6.5: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2') \mid\mid S_3$. We know only one system can move at a time. This means for this α only 1 system can transition and thus this subcase is not possible.
Subcase 6.6: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1' \mid\mid S_2) \mid\mid S_3'$. We know only one system can move at a time. This means for this α only 1 system can transition and thus this subcase is not possible.
Subcase 6.7: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2') \mid\mid S_3'$. We know only one system can move at a time. This means for this α only 1 system can transition and thus this subcase is not possible.
Subcase 6.8: $(S_1 \mid\mid S_2) \mid\mid S_3 \xrightarrow{\alpha} (S_1 \mid\mid S_2) \mid\mid S_3$ We know by Nsys, that only a receive transition can be discarded. This means for our move transition, at least 1 system has to transition and thus 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 $\alpha = \pi\mathsf{G}(\tilde{v})^{ch}$.

Subcase 3.1: Consider Lget, we get $S_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} S_1'$. By Sys together with our α, we can also get $S_1 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S_1'$ 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 $\alpha = \pi\mathsf{S}(\tilde{v})^{ch}$.

Subcase 4.2: Consider Lsup, we get $S_1 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S_1'$. By Sys, we can also get $S_1 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S_1'$ 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.
Case 6: Consider $\alpha = \pi@(\Gamma : P, \ell)$.
Subcase 6.2: Consider Lmov, we get $S_1 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1'$. By Sys, we can also get $S_1 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1'$ on the RHS.
Subcase 6.2: Consider Rmov, this would require the empty process to move, which we know is not possible by Nnul.
As we have covered all cases we have proven that also $S_1 \mid\mid \Gamma : 0 \equiv S_1$. $\square$

A.2 Lemma 2

Given 2 systems S_1 and S_2 proof:
Lemma 2 is regarding the non-blocking behavior of broadcasted messages.
$ch = * \wedge S_1 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \implies S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2' \quad$ 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 $S_2 = \langle \gamma, \mathsf{LS}, \ell \rangle : P$. Consider Lpar we have that we need to prove $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. By Nsys, this also includes $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$. 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 $\Gamma : 0 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : 0$. 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 $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. The premise says we can apply Nsnd if $ch = *$, which it is. We then get $\Gamma : \Pi!(\tilde{e})^{e'}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \Pi!(\tilde{e})^{e'}.U$. By Nsys 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 $ch \in \mathsf{LS}, \ell$, 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, giving us $\Gamma : \Pi?(\tilde{e})^{e'}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \Pi?(\tilde{e})^{e'}.U$. With Nsys this gives $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2$.
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, because we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$. The premise states that we can apply this rule if $ch = *$, which is the case. We get $\Gamma : \Pi\mathsf{G}(\tilde{x})^{e'}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \Pi\mathsf{G}(\tilde{x})^{e'}.U$. By `Nsys`, 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. The premise holds as we know $ch = *$, giving us $\Gamma : \mathsf{S}(\tilde{x})^{e}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \mathsf{S}(\tilde{x})^{e}.U$. By `Nsys` 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: $P = \mathsf{MOV@}\Pi.U$. Consider `Nmov`, which is the only appplicable rule for this P. The premise holds as we know $ch = *$, giving us $\Gamma : \mathsf{MOV@}\Pi.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \mathsf{MOV@}\Pi.U$. By `Nsys` 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: $P = \langle\Pi\rangle P$.

Subcase 7.1: Consider `Grd`, if $\gamma \models \Pi$ and $\langle \gamma, \mathsf{LS}, \ell\rangle : P \xmapsto{\alpha} \langle \gamma, \mathsf{LS}, \ell\rangle : P$, we can receive the message. As we know we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$, and by `Sys` this includes $\Gamma : P \xmapsto{\pi?(\tilde{v})^{ch}} \Gamma' : P'$, we know α, has to be the receiving transition, which we covered in case 3. This gives us $\Gamma : \langle\Pi\rangle P \xmapsto{\pi?(\tilde{v})^{ch}} \Gamma' : P'$. By `Sys` we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

Subcase 7.2: Consider `Blk`, in case $\gamma \not\models \pi$, we can discard as the other premise is satisfied because we have $ch = *$. This gives us $\Gamma : \langle\Pi\rangle P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \langle\Pi\rangle P$. By `Nsys` we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

Subcase 7.3: Consider `Stc`, if $\gamma \models \Pi$ and we can do $\langle \gamma, \mathsf{LS}, \ell\rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell\rangle : P$, which is covered in the previous cases as non-blocking, we can discard. This gives us $\Gamma : \langle\Pi\rangle P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \langle\Pi\rangle P$. By `Nsys` we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

We then have $\big(\gamma \models \Pi \wedge (\langle \gamma, \mathsf{LS}, \ell\rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell\rangle : P \vee \langle \gamma, \mathsf{LS}, \ell\rangle : P \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell\rangle : P)\big) \vee \gamma \not\models \Pi$, which is a tautology. As we know for each subcase we can get the required transition, we have that it is non-blocking as required.

Case 8: $P = P_1 + P_2$.

Subcase 8.1: Consider `Lor`, if $\langle \gamma, \mathsf{LS}, \ell\rangle : P_1 \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell\rangle : P_1'$, we have $\langle \gamma, \mathsf{LS}, \ell\rangle : P_1 + P_2 \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell\rangle : P_1'$. Note we use $\alpha = \pi?(\tilde{v})^{ch}$ because we know we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2$. If this is possible, by `Sys` we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

Subcase 8.2: Consider `Ror`, if $\langle \gamma, \mathsf{LS}, \ell\rangle : P_2 \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell\rangle : P_2'$, we have $\langle \gamma, \mathsf{LS}, \ell\rangle : P_1 + P_2 \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell\rangle : P_2'$. Note we use $\alpha = \pi?(\tilde{v})^{ch}$ because we know we need $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2$. If this is possible, by `Sys` we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

Subcase 8.3: Consider `Nor`, we need $\langle \gamma, \mathsf{LS}, \ell\rangle : P_1 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell\rangle : P_1$ and $\langle \gamma, \mathsf{LS}, \ell\rangle : P_2 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell\rangle : P_2$. By the previous cases, we have covered that these cases are non-blocking for broadcast. We then have $\langle \gamma, \mathsf{LS}, \ell\rangle : P_1 + P_2 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell\rangle : P_1 + P_2$, and by `Nsys` we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

We get the premises, $\langle \gamma, \mathsf{LS}, \ell \rangle : P_1 \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_1' \vee \langle \gamma, \mathsf{LS}, \ell \rangle : P_2 \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_2' \vee (\langle \gamma, \mathsf{LS}, \ell \rangle : P_1 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_1 \wedge \langle \gamma, \mathsf{LS}, \ell \rangle : P_2 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_2)$, which is a tautlogy, so it always holds. 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 9: $P = K(\tilde{x})$.

Subcase 9.1: Consider `Def`, if $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell \rangle : P'$, we can apply this rule and receive the message. By `Sys`, we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

Subcase 9.2: Consider `Ndef`, if $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$, which by our previous cases has been shown to be non-blocking for broadcast, we can discard it. By `Nsys`, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.
As these are all cases (receive or discard) and we know discarding is non-blocking, we have for all cases $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$, making it non-blocking as required.
These are all possible P so we have shown that for broadcast our base cases are non-blocking.
For the inductive step, we take $S_2 = S_i || S_j$. This inductive step follows from `Prec`, which can be applied on S_2 after `Lpar` on $S_1 || S_2$. We get $S_i \xrightarrow{\pi?(\tilde{v})^{ch}} S_i'$ and $S_j \xrightarrow{\pi?(\tilde{v})^{ch}} S_j'$. We need to be able to apply `Sys` or `Nsys` on both S_i and S_j, which has been shown in our base cases. As we have shown the base cases to be non-blocking for broadcast, the inductive step is non-blocking as well.□

A.3 Lemma 3

Lemma 3 shows the blocking of multicast 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. $ch \in \mathsf{LS} \cup \ell \wedge S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$
   $\vee$
2. $ch \notin \mathsf{LS} \cup \ell$

There are two points to this lemma. The first point shows that a system blocks a multicast channel if it is listening but not ready to receive. The second point shows that a system does not block a multicast channel if it is not listening to it.
We will use the same cases as in A.2 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 $S_2 = \langle \gamma, \mathsf{LS}, \ell \rangle : P$. 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 $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$. 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 multicast channel either. We get $\Gamma : 0 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : 0$ for both subcases. By `Nsys` this gives $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ as required.

Case 2: $P = \Pi!(\tilde{e})^{e'}.U$. Consider Nsnd, as we need to get $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$, so it is the only option.
Subcase 2.1 We have that $ch \neq *$ and $ch \in \mathsf{LS} \cup \ell$, meaning our premise does not hold. This means it cannot transition and thus is blocking.

Subcase 2.2 As we have $ch \notin \mathsf{LS} \cup \ell$, our premise holds and we get $\Gamma : \Pi!(\tilde{e})^{e'}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \Pi!(\tilde{e})^{e'}.U$. By Nsys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.
Case 3: $P = \Pi?(\tilde{e})^{e'}.U$.
Subcase 3.1:
Subcase 3.1.1: Consider Rec, we have the premises $[\![e']\!]_\gamma = ch \wedge \gamma \models \{\Pi[\tilde{v}/\tilde{x}]\} \wedge \gamma \models \pi \wedge ch \in \mathsf{LS}, \ell$. We know $ch \in \mathsf{LS}, \ell$, so we are left with $[\![e']\!]_\gamma = ch \wedge \gamma \models \{\Pi[\tilde{v}/\tilde{x}]\} \wedge \gamma \models \pi$. If these hold, we get $\langle \gamma, \mathsf{LS}, \ell \rangle : \Pi?(\tilde{x})^{e'}.U \xmapsto{\pi?(\tilde{v})^{ch}} \{\!|\langle \gamma, \mathsf{LS}, \ell \rangle : [\tilde{v}/\tilde{x}]U|\!\}$. By Sys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.
Subcase 3.1.2: Consider Nrec, as we know $ch \in \mathsf{LS}, \ell$, 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 $\langle \gamma, \mathsf{LS}, \ell \rangle : \Pi?(\tilde{x})^{e'}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : \Pi?(\tilde{x})^{e'}.U$ is never possible in this case. This means that if the premises hold, it receives, but if they do not, it blocks the multicast channel as it cannot discard.
Subcase 3.2
Subcase 3.2.1: Consider Rec, we have that $ch \notin \mathsf{LS}, \ell$, meaning this is not possible as $ch \in \mathsf{LS}, \ell$ is a premise.
Subcase 3.1.2: Consider Nrec, our premise is $ch \notin \mathsf{LS}, \ell \vee \big(ch = * \wedge (\gamma \not\models \{\Pi[\tilde{v}/\tilde{x}]\} \vee \gamma \not\models \pi)\big)$. We have that $ch \notin \mathsf{LS}, \ell$, thus we can apply this rule and get $\Gamma : \Pi?(\tilde{x})^{e'}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \Pi?(\tilde{x})^{e'}.U$. By Nsys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$. making it non-blocking if the process is not listening to the channel.

Case 4: $P = \Pi\mathsf{G}(\tilde{x})^{e'}.U$. Consider Nget, as we need to get $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$.
Subcase 4.1 We have that $ch \neq *$ and $ch \in \mathsf{LS} \cup \ell$, meaning our premise does not hold. This means it cannot transition and thus is blocking.

Subcase 4.2 As we have $ch \notin \mathsf{LS} \cup \ell$, our premise holds and we get $\Gamma : \Pi\mathsf{G}(\tilde{x})^{e'}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \Pi\mathsf{G}(\tilde{x})^{e'}.U$. By Nsys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

Case 5: $P = \mathsf{S}(\tilde{x})^{e'}.U$. Consider Nsup, as it is the only rule we can apply in this case to get $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$.
Subcase 5.1 We have that $ch \neq *$ and $ch \in \mathsf{LS} \cup \ell$, meaning our premise does not hold. This means it cannot transition and thus is blocking.

Subcase 5.2 As we have $ch \notin \mathsf{LS} \cup \ell$, our premise holds and we get $\Gamma : \mathsf{S}(\tilde{e})^{e'}.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \Pi\mathsf{G}(\tilde{e})^{e'}.U$. By Nsys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

Case 6: $P = \mathsf{MOV}@\Pi.U$. Consider Nmov, as it is the only rule we can apply in this case to get $S_2 \xrightarrow{\pi?(\tilde{v})^{ch}} S_2'$.
Subcase 5.1 We have that $ch \neq *$ and $ch \in \mathsf{LS} \cup \ell$, meaning our premise does not hold. This means it cannot transition and thus is blocking.

Subcase 5.2 As we have $ch \notin \mathsf{LS} \cup \ell$, our premise holds and we get $\Gamma : \mathsf{MOV}@\Pi.U \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \Gamma : \mathsf{MOV}@\Pi.U$. By Nsys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.
Case 7: $P = \langle \Pi \rangle P$.
Subcase 7.1: Consider Grd, with λ as the receive label. As the premises do not depend directly on the listening channels, we will cover both cases at the same time. If it can receive by $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell \rangle : P'$

and $\gamma \models \Pi$, it will receive on the multicast channel. By premise in the previous cases, meaning if $\gamma \models \Pi$ holds, it will receive. By Sys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.
Subcase 7.2: Consider Stc. **Subcase 7.2.1:** We have that $ch \neq *$ and $ch \in \mathsf{LS} \cup \ell$, meaning our premise does not hold. The other premise, $\gamma \not\models \Pi$, does not matter for this subcase, but is important for the whole case on blocking of multicast channels. We will get back to this at the end of this case.
Subcase 7.2.2: We have that $ch \notin \mathsf{LS} \cup \ell$, so if $\gamma \not\models \Pi$, we can discard and get $\gamma \models \Pi$ holds, we get $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$. By Nsys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.
Subcase 7.3: Consider Stc. As the premises do not depend directly on the listening channels, we will cover both cases at the same time. We have covered the cases for when $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$ is possible on multicast channels, only blocking when appropriate. If the premise $\gamma \models \Pi$ holds, we get $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$. By Nsys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

By these cases we have that if $\gamma \models \Pi$, and $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell \rangle : P'$ or $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$, which is a tautology, it is blocking according to our previous cases. If $\gamma \not\models \Pi$, it is only blocking when it is listening to the multicast channel, as desired. **Case 8:** $P = P_1 + P_2$. None of the cases depend on *ch* directly, meaning we can cover both cases at the same time.

Subcase 8.1: Consider Lor, if the premise $\langle \gamma, \mathsf{LS}, \ell \rangle : P_1 \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_1'$ holds, we get $\langle \gamma, \mathsf{LS}, \ell \rangle : P_1 + P_2 \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_1'$. By Sys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$

Subcase 8.2: Consider Ror, if the premise $\langle \gamma, \mathsf{LS}, \ell \rangle : P_2 \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_2'$ holds, we know it accepts and we get $\langle \gamma, \mathsf{LS}, \ell \rangle : P_1 + P_2 \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_2'$. By Sys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

Subcase 8.3: Consider Nor, if $\langle \gamma, \mathsf{LS}, \ell \rangle : P_1 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_1$ and $\langle \gamma, \mathsf{LS}, \ell \rangle : P_2 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_2$ hold we get $\langle \gamma, \mathsf{LS}, \ell \rangle : P_1 + P_2 \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P_1 + P_2$. By Nsys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.
We know by our previous cases that if it can receive, it will. In addition it can only discard, if it is not listening to the multicast channel. This means for this case it is also blocking for multicast channels when the channel is in the listening channels.
Case 9: $P = K(\tilde{x})$. Just like the previous cases none of the cases depend on *ch* directly, meaning we can cover both cases at the same time. By our case we have that $K(\tilde{x}) = P$.

Subcase 9.1: Consider Def, if $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell \rangle : P'$, we can apply this rule and receive the message by $\langle \gamma, \mathsf{LS}, \ell \rangle : K(\tilde{x}) \xmapsto{\pi?(\tilde{v})^{ch}} \langle \gamma, \mathsf{LS}, \ell \rangle : P'$. By Sys, we then get $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$.

Subcase 9.2: Consider Ndef, if $\langle \gamma, \mathsf{LS}, \ell \rangle : P \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P'$, we can discard the message by $\langle \gamma, \mathsf{LS}, \ell \rangle : K(\tilde{x}) \xmapsto{\widetilde{\pi?(\tilde{v})^{ch}}} \langle \gamma, \mathsf{LS}, \ell \rangle : P$. By the previous cases can only be done when the system is not listening to the channel.

As these are all cases for $P = K(\tilde{x})$, we then have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ if S_2 can receive or S_2 is not listening to the channel, making it blocking only if appropriate as required.

We have covered all of the subcases and we have $S_1 \mid\mid S_2 \xrightarrow{\pi!(\tilde{v})^{ch}} S_1' \mid\mid S_2'$ if it can receive or is not listening and blocking otherwise.

For the inductive step, we take $S_2 = S_i \mid\mid S_j$. This inductive step follows from `Prec`. Which can be applied on S_2 after `Lpar` on $S_1||S_2$. We get $S_i \xrightarrow{\pi?(\tilde{v})^{ch}} S_i'$ and $S_j \xrightarrow{\pi?(\tilde{v})^{ch}} S_j'$. We need to be able to apply `Sys` or `Nsys` on both S_i and S_j, which has been shown in our base cases to be blocking when $ch \neq *$ and it is in the set of listening channels. As we have shown the base cases to be blocking for multicast, the inductive step is blocking as well.□

A.4 Lemma 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 $S_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S_2'$.
We know by `Nsys`, that the supply transition $\pi\mathsf{S}(\tilde{v})^{ch}$ cannot be discarded. As a complete unicast transition, with the transition label τ, is hidden, the systems which do not supply, do not transition nor discard. We will perform induction on S_2 again. We group our S_2 into two groups, the group which cannot make the transition required. $S_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S_2'$ and the group which can make the transition.

Our base induction case will again be $S_2 = \langle \gamma, \mathsf{LS}, \ell \rangle : P$. By `Luni` this means we have to prove $S_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S_2'$ to get $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$ given $S_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} S_1'$.
Case 1: By our semantics we cannot get the transition label $\pi\mathsf{S}(\tilde{v})^{ch}$ for
$P = 0 \mid \Pi!(\tilde{e})^e.U \mid \Pi?(\tilde{x})^e.U \mid \Pi\mathsf{G}(\tilde{x})^e.U \mid \mathsf{MOV}@\Pi.U \mid \langle\Pi\rangle P \mid P_1 + P_2 \mid K(x_1,...,x_n)$
By `Sys` we cannot get $S_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S_2'$ and therefore also not $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$ for these cases. This means for this base case, it would block unicast as there is no other system supplying. If there was, then the transition is hidden and the systems would not block.
Case 2: For our other P, consisting of a single process, $P = \mathsf{S}(\tilde{e})^e.U$, we have two rules, only one with the desired transition label. Consider `Sup`, the premises are $[\![\tilde{e}]\!]_\gamma = \tilde{v} \wedge [\![e']\!]_\gamma = ch \wedge \gamma \models \pi \wedge ch \in \mathsf{LS} \cup \ell$. The first two premises are concretized values and are not conditional. The π, send by S_1, has to be satisfied. And to receive the get request, S_2 has to be listening to the channel, $ch \in \mathsf{LS} \cup \ell$. If this is the case, S_2 can supply giving us $S_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S_2'$ resulting in $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$. If the premises do not hold, S_2 cannot transition, as `Nsup` has the wrong transition label. This means it would block unicast as there is no other system supplying. If there was, then the transition would be hidden and the system does not block.
Now that we have covered the base cases, we know only a single case can respond to unicast. And if this is not possible, it is blocked. For our inductive step we take $S_2 = S_i||S_j$. By our base case we know at least one of S_i and S_j, has to be able to supply for the get request. If neither can supply, the unicast is blocked, as desired. If one of them can supply, we get $S_1 \mid\mid S_2 \xrightarrow{\tau} S_1' \mid\mid S_2'$. And if both can supply, by `Lsup` and `Rsup`, we know only one of them can supply, but it will not block.
This shows that a unicast get request should never go unanswered, making it blocking as desired. □

A.5 Lemma 5

This lemma is about the commutativity and associativity of environments.

1. $E_1 \oplus E_2 \equiv E_2 \oplus E_1$
2. $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$

We will prove this lemma on case analysis on the transition label α. **1.** The first point is proving commutativity. We can see in the syntax that the cases we have to cover are: $\pi!(\tilde{v})^{ch}$, $\pi?(\tilde{v})^{ch}$, $\pi\mathsf{G}(\tilde{v})^{ch}$, $\pi\mathsf{S}(\tilde{v})^{ch}$, τ and $\pi@(\Gamma : P, \ell)$

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

Subcase 1.1: $E_1 \oplus E_2 \overset{\alpha}{\Rightarrow} E_1' \oplus E_2'$. Assume $E_1 \neq E_1'$ and $E_2 \neq E_2'$, 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 `Elpar`, we have $E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$. Giving us $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$. By `Erpar` on $E_2 \oplus E_1$, we also get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ and $E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1'$, resulting in $E_2 \oplus E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2' \oplus E_1'$. Showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$.

Subcase 1.1.2: Consider `Erpar`, we have $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'$. Giving us $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$. By `Elpar` on $E_2 \oplus E_1$, we also get $E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'$ and $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$, resulting in $E_2 \oplus E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2' \oplus E_1'$. Showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$.

Subcase 1.2: $E_1 \oplus E_2 \overset{\alpha}{\Rightarrow} E_1' \oplus E_2$. In this case only E_1 transitions. Looking at the semantics, it does not immediately seem like such a transition exists. However, by `Com` and `Nsys` we can still apply the $S \xrightarrow{\pi?(\tilde{v})^{ch}} S'$ transition making it $\langle \gamma, \mathsf{LS}, \ell \rangle : P \overset{\widetilde{\pi?(\tilde{v})^{ch}}}{\longmapsto} \langle \gamma, \mathsf{LS}, \ell \rangle : P$ and therefore also having $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ with $E_2 \equiv E_2'$.

Subcase 1.2.1: Consider `Elpar`, we get $E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$, but for this case by `Com` and `Nsys`, we get $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2$. By `Erpar`, `Com` and `Nsys`, we then get the same transitions again. This results in $E_2 \oplus E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2 \oplus E_1'$, showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$.

Subcase 1.2.2: Consider `Erpar`, we would get E_2 with the sending transition, but by our IH, we have that E_2 does not transition. By `Com` and `Nsys`, we see that it is only possible for the receiving transition, and thus this is not possible.

Subcase 1.3: $E_1 \oplus E_2 \overset{\alpha}{\Rightarrow} E_1 \oplus E_2'$. In this case we have the opposite of the previous case.

Subcase 1.3.1: Consider `Elpar`, just like in case 1.2.2, this is not possible, because E_1 would be the sending environment, but we know E_1 does not transition.

Subcase 1.3.2: Consider `Erpar`, we would get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'$, but for this case by `Com` and `Nsys`, we get the discard transition on a process level. This results in $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1 \oplus E_2'$. By `Elpar`, `Com` and `Nsys`, we then get the same transitions. This results in $E_2 \oplus E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2 \oplus E_1'$. Showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$.

Subcase 1.4: $E_1 \oplus E_2 \overset{\alpha}{\Rightarrow} E_1 \oplus E_2$, 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 environment has to transition. This means this transition is not possible.

Case 2: Consider $\alpha = \pi?(\tilde{v})^{ch}$.
Subcase 2.1: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1' \oplus E_2'$. Assume $E_1 \neq E_1'$ and $E_2 \neq E_2'$, as those cases are covered by subcases 2.2, 2.3 and 2.4. Consider `Eprec`, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$. By `Eprec`, we also get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ and $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ for $E_2 \oplus E_1$, showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$.
Subcase 2.2: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1' \oplus E_2$. In this case only E_1 transitions. By `Com` and `Nsys` we can discard the $S \xrightarrow{\pi?(\tilde{v})^{ch}} S'$ for the system inside E_2, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2$. This results in $E_1 \oplus E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1' \oplus E_2$. By `Eprec`, `Com` and `Nsys` on E_2, we also get $E_2 \oplus E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2 \oplus E_1'$ for $E_2 \oplus E_1$, showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$.
Subcase 2.3: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1 \oplus E_2'$. In this case we have the opposite of the previous case. Consider `Eprec`, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$, by `Com` and `Nsys`, we can discard the $S \xrightarrow{\pi?(\tilde{v})^{ch}} S'$ for the system inside E_1, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$. This results in $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1 \oplus E_2'$. By `Eprec`, `Com` and `Nsys` on E_1, we also get $E_2 \oplus E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2' \oplus E_1$ for $E_2 \oplus E_1$, showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$.
Subcase 2.4: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1 \oplus E_2$. This is a combination of subcase 2.2 and 2.3. Consider `Eprec`, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$. By `Com` and `Nsys` on both, we can discard the $S \xrightarrow{\pi?(\tilde{v})^{ch}} S'$ for both the system inside E_1 and E_2, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2$. This gives us $E_1 \oplus E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1 \oplus E_2$. By `Eprec`, `Com` and `Nsys` on both E_2 and E_1, we also get $E_2 \oplus E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2 \oplus E_1$ for $E_2 \oplus E_1$, showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$.

Case 3: Consider $\alpha = \pi\mathsf{G}(\tilde{v})^{ch}$.
Subcase 3.1: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1' \oplus E_2'$. Assume $E_1 \neq E_1'$ and $E_2 \neq E_2'$, as those cases are covered by subcases 3.2, 3.3 and 3.4. The rules which have $\pi\mathsf{G}(\tilde{v})^{ch}$ exposed are `Elget` and `Erget`. Each of these only transition one of the environments, while the other stays in the same state. This means it is not possible for this case.
Subcase 3.2: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1' \oplus E_2$. In this case only E_1 transitions.

Subcase 3.2.1: Consider `Elget`, we get $E_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1'$, resulting in $E_1 \oplus E_2 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1' \oplus E_2$. By `Erget` on $E_2 \oplus E_1$, we also get $E_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1'$, resulting in $E_2 \oplus E_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_2 \oplus E_1'$, showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$ in this case.
Subcase 3.2.2: Consider `Erget`, this would require E_2 to transition, but we know it does not. So this subcase is not possible.
Subcase 3.3: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1 \oplus E_2'$. In this case we have the opposite of the previous case.
Subcase 3.2.1: Consider `Elget`, this would require E_1 to transition, but we know it does not. So this subcase is not possible.
Subcase 3.2.2: Consider `Erget`, we get $E_2 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_2'$, resulting in $E_1 \oplus E_2 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1 \oplus E_2'$. By `Elget`

on $E_2 \oplus E_1$, we also get $E_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_2'$, resulting in $E_2 \oplus E_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_2' \oplus E_1$, showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$ in this case.
Subcase 3.4: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1 \oplus E_2$. Consider `Com` and `Nsys`, together they state only the receive transition can be discarded, which is not the case, so this is not possible.

Case 4: Consider $\alpha = \pi \mathsf{S}(\tilde{v})^{ch}$.
Subcase 4.1: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1' \oplus E_2'$. Assume $E_1 \neq E_1'$ and $E_2 \neq E_2'$, as those cases are covered by subcases 4.2, 4.3 and 4.4. Just like in subcase 3.1, this case is not possible. Only 1 environment can supply at a time, meaning we cannot have two environments transition with an exposed supply transition.
Subcase 4.2: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1' \oplus E_2$. In this case only E_1 transitions.

Subcase 4.2.1: Consider `Elsup`, we get $E_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_1'$, resulting in $E_1 \oplus E_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_1' \oplus E_2$. By `Ersup` on $E_2 \oplus E_1$, we also get $E_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_1'$, resulting in $E_2 \oplus E_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_2 \oplus E_1'$, showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$ in this case.
Subcase 4.2.2: Consider `Ersup`, this would require E_2 to transition, but we know it does not. So this subcase is not possible.
Subcase 4.3: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1 \oplus E_2'$. In this case we have the opposite of the previous case.
Subcase 4.3.1: Consider `Elsup`, this would require E_2 to transition, but we know it does not. So this subcase is not possible.
Subcase 4.3.2: Consider `Ersup`, we get $E_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_2'$, resulting in $E_1 \oplus E_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_1 \oplus E_2'$. By `Elsup` on $E_2 \oplus E_1$, we also get $E_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_2'$, resulting in $E_2 \oplus E_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_2' \oplus E_1$, showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$ in this case.
Subcase 4.4: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1 \oplus E_2$. Consider `Com` and `Nsys`, together they state only the receive transition can be discarded, which is not the case, so this is not possible.

Case 5: Consider $\alpha = \tau$.
Subcase 5.1: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1' \oplus E_2'$. Assume $E_1 \neq E_1'$ and $E_2 \neq E_2'$, as those cases are covered by subcases 5.2, 5.3 and 5.4. For this case we have two transitions as subcases, the unicast τ transition and the move τ transition.
Subcase 5.1.1: Consider `Cex`, we get $E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1'$ and $E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2'$ resulting in $E_1 \oplus E_2 \stackrel{\tau}{\Rightarrow} E_1' \oplus E_2'$. By `Cex`, we also get $E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2'$ and $E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1'$ for $E_2 \oplus E_1$. Resulting in $E_2 \oplus E_1 \stackrel{\tau}{\Rightarrow} E_2' \oplus E_1'$. Therefore $E_1 \oplus E_2 \equiv E_2 \oplus E_1$ holds for this case.
subcase 5.1.2: Consider `Lcom`, we know only E_1 transitions, but as know both environments transition in this subcase, this is not possible.
subcase 5.1.3: Consider `Rcom`, we know only E_2 transitions, but as know both environments transition in this subcase, this is not possible.
Subcase 5.1.4: Consider `Eluni`, we get $E_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_2'$ resulting in $E_1 \oplus E_2 \stackrel{\tau}{\Rightarrow} E_1' \oplus E_2'$. By `Eruni`, we also get $E_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_2'$ and $E_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_1'$ for $E_2 \oplus E_1$. Resulting in $E_2 \oplus E_1 \stackrel{\tau}{\Rightarrow} E_2' \oplus E_1'$. Therefore $E_1 \oplus E_2 \equiv E_2 \oplus E_1$ holds for this case.

Subcase 5.1.5: Consider Eruni, we get $E_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_2'$ resulting in $E_1 \oplus E_2 \stackrel{\tau}{\Rightarrow} E_1' \oplus E_2'$. By Eluni, we also get $E_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_2'$ and $E_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_1'$ for $E_2 \oplus E_1$. Resulting in $E_2 \oplus E_1 \stackrel{\tau}{\Rightarrow} E_2' \oplus E_1'$. Therefore $E_1 \oplus E_2 \equiv E_2 \oplus E_1$ holds for this case.
Subcase 5.2: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1' \oplus E_2$. In this case only E_1 transitions with τ.
subcase 5.2.1: Consider Cex, we see both E_1 and E_2 should transition, as we know E_2 does not we need to look at a lower level to see if the $\pi@(\Gamma : P, \ell)$ transition can be discarded on a process level. Both Out and In have the same system level transition as a premise, the move transition. On a system level we see that by Nsys, only the receive transition can be discarded. This is not the transition we have in our premise, meaning E_2 has to transition as well. Therefore this case in not possible.
subcase 5.2.2: Consider Lcom, We get $E_1 \stackrel{\tau}{\Rightarrow} E_1'$, resulting in $E_1 \oplus E_2 \stackrel{\tau}{\Rightarrow} E_1' \oplus E_2$. By Rcom on $E_2 \oplus E_1$, we also get $E_1 \stackrel{\tau}{\Rightarrow} E_1'$. This gives us $E_2 \oplus E_1 \stackrel{\tau}{\Rightarrow} E_2 \oplus E_1'$. Therefore $E_1 \oplus E_2 \equiv E_2 \oplus E_1$ holds for this case.
subcase 5.2.3: Consider Rcom, we know only E_1 transitions, so this is not possible as E_1 does not transition in this case.
subcase 5.2.4: Consider Eluni, from subcase 5.2.1 we know on a system level only the receive transition can be discarded. By Com, we have that this is not the transition happening on a system level. So this case is not possible.
subcase 5.2.5: Consider Eruni, from subcase 5.2.1 we know on a system level only the receive transition can be discarded. By Com, we have that this is not the transition happening on a system level. So this case is not possible.
Subcase 5.3: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1 \oplus E_2'$. In this case we have the opposite of the previous case.
subcase 5.3.1: Consider Cex, for the same reasoning as in subcase 5.2.1, we cannot discard the environment transition for E_1, so this case is not possible.
subcase 5.3.2: Consider Lcom, we know only E_2 transitions, so this is not possible as E_2 does not transition in this case.
subcase 5.3.3: Consider Rcom, We get $E_2 \stackrel{\tau}{\Rightarrow} E_2'$, resulting in $E_1 \oplus E_2 \stackrel{\tau}{\Rightarrow} E_1 \oplus E_2'$. By Rcom on $E_2 \oplus E_1$, we also get $E_2 \stackrel{\tau}{\Rightarrow} E_2'$. This gives us $E_2 \oplus E_1 \stackrel{\tau}{\Rightarrow} E_2' \oplus E_1$. Therefore $E_1 \oplus E_2 \equiv E_2 \oplus E_1$ holds for this case.
subcase 5.3.4: Consider Eluni, for the same reasoning as in subcase 5.2.4, we cannot discard the environment transition for E_1, so this case is not possible.
subcase 5.3.5: Consider Eruni, for the same reasoning as in subcase 5.2.5, we cannot discard the environment transition for E_1, so this case is not possible.
Subcase 5.4: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1 \oplus E_2$. This case is not possible for a τ transition. By the previous subcases we know we cannot discard the transitions required for a *tau* transition on a process level. This means at least 1 environment has to transition, which is not the case.

Case 6: Consider $\alpha = \pi@(\Gamma : P, \ell)$.
Subcase 6.1: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1' \oplus E_2'$. Assume $E_1 \neq E_1'$ and $E_2 \neq E_2'$, as those cases are covered by subcases 6.2, 6.3 and 6.4. We can see that the only environment level transitions with $\pi@(\Gamma : P, \ell)$ exposed, have only a single environment transitioning, so this case is not possible.
Subcase 6.2: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1' \oplus E_2$. In this case only E_1 transitions. We consider the two move transitions.

Subcase 6.2.1: Consider `Lex`, we get $E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1'$. This results in $E_1 \oplus E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1' \oplus E_2$. By `Rex` on $E_2 \oplus E_1$ we then get $E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1'$ as well. This results in $E_2 \oplus E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2 \oplus E_1'$. Showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$.
Subcase 6.2.2: Consider `Rex`, we would get E_2 with the moving transition, but we have that E_2 does not transition, so this is not possible.
Subcase 6.3: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1 \oplus E_2'$. In this case we have the opposite of the previous case.
Subcase 6.2.1: Consider `Lex`, we would get E_1 with the moving transition, but we have that E_1 does not transition, so this is not possible.
Subcase 6.2.2: Consider `Rex`, we get $E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2'$. This results in $E_1 \oplus E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1 \oplus E_2'$. By `Lex` on $E_2 \oplus E_1$ we then get $E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2'$ as well. This results in $E_2 \oplus E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2' \oplus E_1$. Showing $E_1 \oplus E_2 \equiv E_2 \oplus E_1$.
Subcase 6.4: $E_1 \oplus E_2 \stackrel{\alpha}{\Rightarrow} E_1 \oplus E_2$. The two possible rules we can apply, requires an environment to transition with the rule `Out` or `In`. A premise for these rules is that on a system level the transition with label $\pi@(\Gamma:P,\ell)$ can be made. By `Nsys`, we know a move transition cannot be discarded, so at least one of the environments has to transition. This means this case is not possible.
As it holds for all cases, we have proven commutativity. □

2. The second point is proving associativity. We can see in the syntax that the cases we have to cover are: $\pi!(\tilde{v})^{ch}$, $\pi?(\tilde{v})^{ch}$, τ and $\pi@(\Gamma:P,\ell)$
Case 1: Consider $\alpha = \pi!(\tilde{v})^{ch}$.
Subcase 1.1: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3'$. Assume $E_i \neq E_i'$ as that is covered by other subcases. As we have a sending transition, we will end up with 3 subcases, namely of each environment being the sending environment, because only 1 system can send at a time, meaning only 1 environment can as well.
Subcase 1.1.1: Consider `Elpar`, we get $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$. We can now split the first transition into two subcases, one where E_1 is sending and one where E_2 is sending.
Subcase 1.1.1.1: Consider `Elpar` on $E_1 \oplus E_2$. We get $E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$, together with $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$, we then have $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} (E_1' \oplus E_2') \oplus E_3'$. By `Elpar` on $E_1 \oplus (E_2 \oplus E_3)$, we get $E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1'$ and $E_2 \oplus E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2' \oplus E_3'$. By `Eprec`, we then get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$, which are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 1.1.1.2: Now consider `Erpar` on $E_1 \oplus E_2$. We get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'$, together with $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$, we then have $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} (E_1' \oplus E_2') \oplus E_3'$. By `Erpar` on $E_1 \oplus (E_2 \oplus E_3)$, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2' \oplus E_3'$. By `Elpar`, we then get $E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$, which are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 1.1.2: Consider `Erpar`, we get $E_1 \oplus E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1' \oplus E_2'$ and $E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_3'$. By `Eprec` on $E_1 \oplus E_2$, we then get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$. By `Erpar` on $E_1 \oplus (E_2 \oplus E_3)$, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$

and $E_2 \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2' \oplus E_3'$. By `Erpar` on $E_2 \oplus E_3$, we then get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_3'$, which are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Subcase 1.2: $(E_1 \oplus E_2) \oplus E_3 \xRightarrow{\alpha} (E_1' \oplus E_2) \oplus E_3$. Here only E_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 E_1.

Subcase 1.2.1: Consider `Elpar`, we get $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$.

Subcase 1.2.1.1: Consider `Elpar` on $E_1 \oplus E_2$. We get $E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$. By `Com` and `Nsys` on $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$, we get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3$. We then have $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} (E_1' \oplus E_2) \oplus E_3$. By `Elpar` on $E_1 \oplus (E_2 \oplus E_3)$, we get $E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1'$ and $E_2 \oplus E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2' \oplus E_3'$. By `Eprec`, `Com` and `Nsys` on both transitions, we then get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3$, which are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Subcase 1.2.1.1: Consider `Erpar` on $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$. This would require E_2 to be the sending system, however we know E_2 does not transition and thus this is not possible.

Subcase 1.2.2: Consider `Erpar`. This would require E_3 to be the sending system, however we know E_3 does not transition and thus this is not possible.

Subcase 1.3: $(E_1 \oplus E_2) \oplus E_3 \xRightarrow{\alpha} (E_1 \oplus E_2') \oplus E_3$. This case is similar to the previous subcase, but this time we have E_2 which has to be the sending transition.

Subcase 1.3.1: Consider `Elpar`, we get $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$.

Subcase 1.3.1.1: Consider `Elpar` on $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$. This would require E_1 to be the sending system, however we know E_1 does not transition and thus this is not possible.

Subcase 1.3.1.2: Consider `Erpar` on $E_1 \oplus E_2$. We get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'$. By `Com` and `Nsys` on $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3$. We then have $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} (E_1 \oplus E_2') \oplus E_3$. By `Erpar` on $E_1 \oplus (E_2 \oplus E_3)$, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2' \oplus E_3'$. By `Elpar`, `Com` and `Nsys` on E_3, we then get $E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3$, which are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Subcase 1.3.2: Consider `Erpar`. This would require E_3 to be the sending system, however we know E_3 does not transition and thus this is not possible.

Subcase 1.4: $(E_1 \oplus E_2) \oplus E_3 \xRightarrow{\alpha} (E_1 \oplus E_2) \oplus E_3'$. This is once again a similar case, but this time with E_3 being the sending transition.

Subcase 1.4.1: Consider `Elpar`. This would require E_1 or E_2 to be the sending system, however we know neither transitions and thus this is not possible.

Subcase 1.4.2: Consider `Erpar`, we get $E_1 \oplus E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1' \oplus E_2'$ and $E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_3'$. Now consider `Eprec` on $E_1 \oplus E_2$, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$. By `Com` and `Nsys` on $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$,

we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2$. Resulting in $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} (E_1 \oplus E_2) \oplus E_3'$. By `Erpar` on $E_1 \oplus (E_2 \oplus E_3)$ we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2' \oplus E_3'$. Then by `Erpar` again on the latter transition we get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_3'$. By `Com` and `Nsys` on E_1 and E_2 this results in $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} (E_1 \oplus E_2) \oplus E_3'$ with the same transitions. Showing $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Subcase 1.5: $(E_1 \oplus E_2) \oplus E_3 \overset{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3$. In this case E_1 and E_2 transition, meaning either of them can be the sending system.

Subcase 1.5.1: Consider `Elpar`, we get $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$.

Subcase 1.5.1.1: Consider `Elpar` on $E_1 \oplus E_2$. We get $E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$. By `Com` and `Nsys` on $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$, we get $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3$, we then have $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} (E_1' \oplus E_2') \oplus E_3$. By `Elpar` on $E_1 \oplus (E_2 \oplus E_3)$, we get $E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1'$ and $E_2 \oplus E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2' \oplus E_3'$. By `Eprec`, we then get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$ and by `Com` and `Nsys` on the latter we get $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3$, which are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Subcase 1.5.1.2: Now consider `Erpar` on $E_1 \oplus E_2$. We get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'$. By `Com` and `Nsys` on $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$, we get $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3$, we then have $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} (E_1' \oplus E_2') \oplus E_3$. By `Erpar` on $E_1 \oplus (E_2 \oplus E_3)$, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2' \oplus E_3'$. By `Elpar`, we then get $E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$, and by `Com` and `Nsys` we have $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3$, which are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Subcase 1.5.2: Consider `Erpar`, this would require E_3 to be the sending system, but as it does not transition this is not possible.

Subcase 1.6: $(E_1 \oplus E_2) \oplus E_3 \overset{\alpha}{\Rightarrow} (E_1' \oplus E_2) \oplus E_3'$. This is similar to the previous case, but now with E_1 or E_3 being the sending transition.

Subcase 1.6.1: Consider `Elpar`, we get $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$.

Subcase 1.6.1.1: Consider `Elpar` on $E_1 \oplus E_2$. We get $E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$. By `Com` `Nsys` on $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$, we get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2$. Together with $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$ we then have $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}}$ $(E_1' \oplus E_2) \oplus E_3'$. By `Elpar` on $E_1 \oplus (E_2 \oplus E_3)$, we get $E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1'$ and $E_2 \oplus E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2' \oplus E_3'$. By `Eprec`, we then get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$ and by `Com` and `Nsys` on the latter we get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2$, which are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Subcase 1.6.1.2: Consider `Erpar` on $E_1 \oplus E_2$. This would require E_2 to be the sending transition but as it does not transition this is not possible.

Subcase 1.6.2: Consider Erpar, we get $E_1 \oplus E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1' \oplus E_2'$ and $E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_3'$. By Eprec on $E_1 \oplus E_2$, we then get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$, combined with Com and Nsys on the latter we get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2$. Resulting in $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} (E_1' \oplus E_2) \oplus E_3'$. By Erpar on $E_1 \oplus (E_2 \oplus E_3)$, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2' \oplus E_3'$. By Erpar on $E_2 \oplus E_3$, we then get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_3'$. By Com and Nsys on $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ we get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2$, which are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 1.7: $(E_1 \oplus E_2) \oplus E_3 \overset{\alpha}{\Rightarrow} (E_1 \oplus E_2') \oplus E_3'$. Once again this is a similar case to the previous subcases but with E_2 or E_3 being the sending system.
Subcase 1.7.1: Consider Elpar, we get $E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$.
Subcase 1.7.1.1: Consider Elpar on $E_1 \oplus E_2$. This would require E_1 to be the sending transition but as it does not transition this is not possible.
Subcase 1.7.1.2: Consider Erpar on $E_1 \oplus E_2$. We get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'$. By Com and Nsys on $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1$. Together with $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$ we then have $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} (E_1 \oplus E_2') \oplus E_3'$. By Erpar on $E_1 \oplus (E_2 \oplus E_3)$, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2' \oplus E_3'$. By Erpar again, we then get $E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$ and by Com and Nsys on $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1$. These are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 1.7.2: Consider Erpar, we get $E_1 \oplus E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1' \oplus E_2'$ and $E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_3'$. By Eprec on $E_1 \oplus E_2$, we then get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$, combined with Com and Nsys on the first we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1$. Resulting in $(E_1 \oplus E_2) \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} (E_1 \oplus E_2') \oplus E_3'$. By Erpar on $E_1 \oplus (E_2 \oplus E_3)$, we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \oplus E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_2' \oplus E_3'$. By Erpar on $E_2 \oplus E_3$, we then get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi!(\tilde{v})^{ch}} E_3'$. By Com and Nsys on $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ we get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1$, which are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 1.8: $(E_1 \oplus E_2) \oplus E_3 \overset{\alpha}{\Rightarrow} (E_1 \oplus E_2) \oplus E_3$. 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: $(E_1 \oplus E_2) \oplus E_3 \overset{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3'$. Assume $E_i \neq E_i'$ as that is covered by other subcases. Consider Eprec, we get $E_1 \oplus E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1' \oplus E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$. By Eprec on $E_1 \oplus E_2$ we also get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$. By Eprec twice on $E_1 \oplus (E_2 \oplus E_3)$ we also get $E_1 \xrightarrow{\pi?(\tilde{v})^{ch}} E_1'$, $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi?(\tilde{v})^{ch}} E_3'$ which are the same transitions and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for

this case.
Subcase 2.2: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2) \oplus E_3$. Here only E_1 transitions. This subcase goes the same as the previous subcase, but now we apply `Com` and `Nsys` on the transitions of E_2 and E_3 on both sides, still giving us the same transitions. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 2.3: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2') \oplus E_3$. Here only E_2 transitions. This subcase goes the same as the previous subcase, but now we apply `Com` and `Nsys` on the transitions of E_1 and E_3 on both sides, still giving us the same transitions. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 2.4: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2) \oplus E_3'$. Here only E_3 transitions. This subcase goes the same as the previous subcase, but now we apply `Com` and `Nsys` on the transitions of E_1 and E_2 on both sides, still giving us the same transitions. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 2.5: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3$. In this case E_1 and E_2 transition. We take the same steps in subcase 2.1 and then apply `Com` and `Nsys` on the transition of E_3 on both sides, giving us the same transitions on each side. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 2.6: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2) \oplus E_3'$. In this case E_1 and E_3 transition. We take the same steps in subcase 2.5 and then apply `Com` and `Nsys` on the transition of E_2 on both sides, giving us the same transitions on each side. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 2.7: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2') \oplus E_3'$. In this case E_2 and E_3 transition. We take the same steps in subcase 2.5 and then apply `Com` and `Nsys` on the transition of E_1 on both sides, giving us the same transitions on each side. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 2.8: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2) \oplus E_3$. Now none of the environments transition, meaning we apply `Com` and `Nsys` on the transitions of E_1, E_2 and E_3 on both sides. This gives us the same transitions on both sides. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Case 3: Consider $\alpha = \pi\mathsf{G}(\tilde{v})^{ch}$.
Subcase 3.1: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3'$. We know only one environment 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: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2) \oplus E_3$. As we know an evironment has to transition to send a get request, we know E_1 has to have the getting system. The only way to get this is by applying `Elget` twice. We get $E_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1'$. By `Elget` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1'$. which is the same transition and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 3.3: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2') \oplus E_3$. As we know an environment has to transition to send a get request, we know E_2 has to have the getting system. The only way to get this is by applying `Elget` and `Erget`. We get $E_2 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_2'$. By `Erget` and `Elget` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_2 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_2'$. which is the same transition and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 3.4: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2) \oplus E_3'$. As we know an environment has to transition to send a get request, we know E_3 has to have the getting system. The only way to get this is by applying `Erget`. We get $E_3 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_3'$. By `Erget` twice on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_3 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_3'$. which is the same transition and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 3.5: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3$. We know only one environment can send a get request

at a time. We also know that if an other (or the same) environment supplies, the transition is hidden, this means for this α only 1 environment can transition and thus this subcase is not possible.
Subcase 3.6: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2) \oplus E_3'$. We know only one environment can send a get request at a time. We also know that if an other (or the same) environment supplies, the transition is hidden, this means for this α only 1 environment can transition and thus this subcase is not possible.
Subcase 3.7: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2') \oplus E_3'$. We know only one environment can send a get request at a time. We also know that if an other(or the same) environment supplies, the transition is hidden, this means for this α only 1 environment can transition and thus this subcase is not possible.
Subcase 3.8: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2) \oplus E_3$. We know by `Com` and `Nsys`, that only a receive transition can be discarded. This means for our get transition, at least 1 environment has to transition and thus this subcase is not possible.

Case 4: Consider $\alpha = \pi\mathsf{S}(\tilde{v})^{ch}$.
Subcase 4.1: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3'$. This case has similar reasoning to the previous case. We know only one environment can supply at a time, thus this subcase is not possible.
Subcase 4.2: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2) \oplus E_3$. As we know an environment has to transition to supply, we know E_1 has to be the supplying environment. The only way to get this is by applying `Elsup` twice. We get $E_1 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_1'$. By `Lsup` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_1 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_1'$. This is the same transition and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 4.3: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2') \oplus E_3$. As we know an environment has to transition to supply, we know E_2 has to have the supplying system. The only way to get this is by applying `Elsup` and `Ersup`. We get $E_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_2'$. By `Ersup` and `Elsup` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_2'$. This is the same transition and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 4.4: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2) \oplus E_3'$. As we know an environment has to transition to supply, we know E_3 has to have the supplying system. The only way to get this is by applying `Ersup`. We get $E_3 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_3'$. By `Ersup` twice on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_3 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_3'$. This is the same transition and shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 4.5: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3$. We know only one environment can supply at a time, therefore this subcase is not possible.
Subcase 4.6: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2) \oplus E_3'$. We know only one system can supply at a time, therefore this subcase is not possible.
Subcase 4.7: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2') \oplus E_3'$. We know only one system can supply at a time, therefore this subcase is not possible.
Subcase 4.8: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2) \oplus E_3$ We know by `Com` and `Nsys`, that only a receive transition can be discarded. This means for our supply transition, at least one environment has to transition and thus this subcase is not possible.

Case 5: Consider $\alpha = \tau$.
Subcase 5.1: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3'$. Assume $E_i \neq E_i'$ as that is covered by other subcases. None of the environment τ transition rules allow more than two environments to transition. This makes sense

as we know a move transition can only happen between two externally connected environments and unicast communication can only happen between 2 systems, which can at most be in 2 environments, so this case is not possible.

Subcase 5.2: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2) \oplus E_3$. Here only E_1 transitions. We can ignore the rules `Cex`, `Eluni` and `Eruni` as they have 2 transitioning environments. The only way to then get E_1 to transition is by applying `Eltau` twice, giving us $E_1 \stackrel{\tau}{\Rightarrow} E_1'$. By `Eltau` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_1 \stackrel{\tau}{\Rightarrow} E_1'$ as our only transition, showing $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Subcase 5.3: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2') \oplus E_3$. Here only E_2 transitions. We can ignore the rules `Cex`, `Eluni` and `Eruni` as they have 2 transitioning environments. The only way to then get E_2 to transition is by applying `Eltau` and `Ertau`, giving us $E_2 \stackrel{\tau}{\Rightarrow} E_2'$. By `Ertau` and `Eltau` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_2 \stackrel{\tau}{\Rightarrow} E_2'$ as our only transition, showing $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Subcase 5.4: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2) \oplus E_3'$. Here only E_3 transitions. We can ignore the rules `Cex`, `Eluni` and `Eruni` as they have 2 transitioning environments. The only way to then get E_1 to transition is by applying `Ertau`, giving us $E_3 \stackrel{\tau}{\Rightarrow} E_3'$. By `Ertau` twice on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_3 \stackrel{\tau}{\Rightarrow} E_3'$ as our only transition, showing $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Subcase 5.5: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3$. In this case E_1 and E_2 transition. Consider the rules `Cex`, `Eluni`, `Eruni` and `Ertau`, all of these require E_3 to transition. Now consider `Eltau`, this gives us $E_1 \oplus E_2 \stackrel{\tau}{\Rightarrow} E_1' \oplus E_2'$.

Subcase 5.5.1: Consider `Cex`, we get $E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1'$ and $E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2'$. By `Cex` and `Lex` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1'$ and $E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2'$. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ holds for this case.

Subcase 5.5.2: Consider `Eluni`, we get $E_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_2'$. By `Eluni` and `Elsup` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_2'$. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ holds for this case.

Subcase 5.5.3: Consider `Eruni`, we get $E_1 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_2'$. By `Eruni` and `Elget` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_1 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_1'$ and $E_2 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_2'$. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ holds for this case.

Subcase 5.6: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2) \oplus E_3'$. In this case E_1 and E_3 transition. This means for the first step we do not have to consider `Eltau` and `Ertau` and they only transition one half of the composition.

Subcase 5.6.1: Consider `Cex` and we get $E_3 \xrightarrow{\pi@(\Gamma:P,\ell)} E_3'$ and the same transition label for $E_1 \oplus E_2$. This leaves us with one option, `Lex`, as E_2 does not transition, resulting in $E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1'$. By `Cex` and `Rex` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1'$ and $E_3 \xrightarrow{\pi@(\Gamma:P,\ell)} E_3'$. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ holds for this case.

Subcase 5.6.2: Consider `Eluni`, we get $E_3 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_3'$ and $E_1 \oplus E_2$ sending the get request. We know this

has to be E_1, meaning we can only apply `Elget`, resulting in $E_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_1'$. By `Eluni` and `Ersup`, we also get $E_1 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_1'$ and $E_3 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_3'$. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ holds for this case.

Subcase 5.6.3: Consider `Eruni`, we get $E_3 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_3'$ and $E_1 \oplus E_2$ supplying. We know this has to be E_1, meaning we can only apply `Elsup`, resulting in $E_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_1'$. By `Eruni` and `Ersup`, we also get $E_1 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_1'$ and $E_3 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_3'$. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ holds for this case.

Subcase 5.7: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2') \oplus E_3'$. In this case E_2 and E_3 transition. This means for the first step we do not have to consider `Eltau` and `Ertau` and they only transition one half of the composition.

Subcase 5.7.1: Consider `Cex` and we get $E_3 \xrightarrow{\pi @ (\Gamma : P, \ell)} E_3'$ and the same transition label for $E_1 \oplus E_2$. This leaves us with one option, `Rex`, as E_1 does not transition, resulting in $E_2 \xrightarrow{\pi @ (\Gamma : P, \ell)} E_2'$. By `Ertau` and `Cex` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_2 \xrightarrow{\pi @ (\Gamma : P, \ell)} E_2'$ and $E_3 \xrightarrow{\pi @ (\Gamma : P, \ell)} E_3'$. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ holds for this case.

Subcase 5.7.2: Consider `Eluni`, we get $E_3 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_3'$ and $E_1 \oplus E_2$ sending the get request. We know this has to be E_2, meaning we can only apply `Erget`, resulting in $E_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_2'$. By `Ertau` and `Eluni`, we also get $E_2 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_3'$. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ holds for this case.

Subcase 5.7.3: Consider `Eruni`, we get $E_3 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_3'$ and $E_1 \oplus E_2$ supplying. We know this has to be E_2, meaning we can only apply `Ersup`, resulting in $E_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_2'$. By `Ertau` and `Eruni`, we also get $E_2 \xrightarrow{\pi \mathsf{S}(\tilde{v})^{ch}} E_2'$ and $E_3 \xrightarrow{\pi \mathsf{G}(\tilde{v})^{ch}} E_3'$. This shows $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ holds for this case.

Subcase 5.8: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2) \oplus E_3$. Now none of the environments transition. By `Com` and `Nsys`, we know only the receive transition can be discarded. A τ cannot be discarded and none of the τ transitions is made up of a receive transition (as it only includes unicast and movement), so this is not possible.

Case 6: Consider $\alpha = \pi @ (\Gamma : P, \ell)$.

Subcase 6.1: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3'$. Assume $E_i \neq E_i'$ as that is covered by other subcases. None of the exposed move transition rules allow more than one environments to transition. This makes sense as we know a move transition can only happen between two externally connected environments, in which case it is a hidden transition, so this case is not possible.

Subcase 6.2: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2) \oplus E_3$. Here only E_1 transitions.

Subcase 6.2.1: Consider `Lex`, we get $E_1 \oplus E_2$ with the move transition and E_3 does not transition.

Subcase 6.2.1.1: By `Lex` on $E_1 \oplus E_2$ we get $E_1 \xrightarrow{\pi @ (\Gamma : P, \ell)} E_1'$ and E_2 without a transition. By `Lex` on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_1 \xrightarrow{\pi @ (\Gamma : P, \ell)} E_1'$ as our only transition, showing $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.

Subcase 6.2.1.2: By `Rex` on $E_1 \oplus E_2$ we would get that E_1 does not transition, which is not the case in this

subcase so this does not hold.
Subcase 6.2.2: Consider Rex, this would mean E_1 does not transition, which does not hold for this subcase so this is not possible.
Subcase 6.3: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2') \oplus E_3$. Here only E_2 transitions.
Subcase 6.3.1: Consider Lex, we get $E_1 \oplus E_2$ with the move transition and E_3 does not transition.
Subcase 6.3.1.1: By Lex on $E_1 \oplus E_2$ we would get that E_2 does not transition, which is not the case in this subcase so this does not hold.
Subcase 6.3.1.2: By Rex on $E_1 \oplus E_2$ we get $E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2'$ and E_1 without a transition. By Rex twice on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2'$ as our only transition, showing $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 6.3.2: Consider Rex, this would mean E_2 does not transition, which does not hold for this subcase so this is not possible.
Subcase 6.4: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2) \oplus E_3'$. Here only E_3 transitions.
Subcase 6.4.1: Consider Lex, we would get that E_2 does not transition, which is not the case in this subcase so this does not hold. **Subcase 6.4.2:** Consider Rex, we get $E_3 \xrightarrow{\pi@(\Gamma:P,\ell)} E_3'$ and the two other environments not transitioning. By Rex twice on $E_1 \oplus (E_2 \oplus E_3)$, we also get $E_3 \xrightarrow{\pi@(\Gamma:P,\ell)} E_3'$ as our only transition, showing $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$ for this case.
Subcase 6.5: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2') \oplus E_3$. In this case E_1 and E_2 transition. Like in subcase 6.1, we know for an exposed move transition only a single environment can transition, so this case is not possible.
Subcase 6.6: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1' \oplus E_2) \oplus E_3'$. In this case E_1 and E_3 transition. Like in subcase 6.1, we know for an exposed move transition only a single environment can transition, so this case is not possible.
Subcase 6.7: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2') \oplus E_3'$. In this case E_2 and E_3 transition. Like in subcase 6.1, we know for an exposed move transition only a single environment can transition, so this case is not possible.
Subcase 6.8: $(E_1 \oplus E_2) \oplus E_3 \stackrel{\alpha}{\Rightarrow} (E_1 \oplus E_2) \oplus E_3$. Now none of the environments transition. A move transition comes from Out or In, a premise of this is that on a system level the move transition can be made. By Nsys we know only the receive transition can be discarded so at least one environment has to transition, so this case is no possible.
As it holds for all cases, we have proven associativity. □

A.6 Lemma 6

This lemma is about the blocking of moving in an environment composition. It shows that when a system leaves an environment, it always enters an other externally connected environment. It also shows a movement is allowed by an environment, a process can only request it.
For $E_1 = [S_1]^{\mathsf{RC}_1,\ell}_{c_1,n_1}$ and $E_2 = [S_2]^{\mathsf{RC}_2,\ell}_{c_2,n_2}$:
$E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1' \implies E_1 \oplus E_2 \stackrel{\tau}{\Rightarrow} E_1' \oplus E_2'$ for any E_2 such that: $\ell = \ell_2,\ n_1 > 0,\ c_2 > n_2,\ \mathsf{RC}_2 \models \pi$ and $\mathsf{RC}_1(\ell) = 1 \wedge E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2'$

Consider Cex, we have that $E_1 \xrightarrow{\pi@(\Gamma:P,\ell)} E_1'$ and $E_2 \xrightarrow{\pi@(\Gamma:P,\ell)} E_2'$ to get $E_1 \oplus E_2 \xRightarrow{\tau} E_1' \oplus E_2$. We know $\ell = \ell_2$, which by In means that E_2 is the receiving environment.
Consider Out, we know $\mathsf{RC}_1(\ell_2) = 1$ and $n_1 > 0$. This leaves us with $S_1 \xrightarrow{\pi@(\Gamma:P,\ell)} S_1'$.
Consider the definition of $S_1 \triangleq \Gamma : P \mid\mid S_1''$ as in Out. Now consider Lmov, we get $\Gamma : P \xrightarrow{\pi@(\Gamma:P,\ell)} \Gamma' : P'$. By Sys and Mov, we get $\Gamma : P \xmapsto{\pi@(\Gamma:P,\ell)} \Gamma' : P'$. We see that the premises on this transition are not conditional. Now consider In, we know $\mathsf{RC} \models \pi$ and $c_2 > n_2$. We have also just shown that if E_1 can do Out, we get $\Gamma : P \xrightarrow{\pi@(\Gamma:P,\ell)} \Gamma' : P'$. meaning the move transition can be completed and will become a hidden transition by Cex.
It shows that the only premises are based on the environments. Now consider Com and Nsys. We can only discard the receive transition. This means that if E_2 is unable to receive the system, we do not get the hidden transition. As the other environments and systems cannot discard this transition, it is blocking.□

A.7 Lemma 7

This lemma is regarding the non-blocking behavior of broadcasted messages. Given 2 environments E_1 and E_2 prove:
$ch = * \wedge E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \implies E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \mid\mid E_2'$ for any E_2.

We can only apply Elpar to $E_1 \oplus E_2$. We get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$.

For our base case we take $E_2 = [S_2]^{\mathsf{RC}_2,\ell}_{c_2,n_2}$. By Com we get that it depends on $S \xrightarrow{\pi?(\tilde{v})^{ch}} S'$, which we have proven to be non-blocking for broadcast in Lemma A.2. This means for the base case on an environment level it is also non-blocking for broadcast.

For our inductive step we take $E_2 = E_i \oplus E_j$. The only applicable rule on $E_i \oplus E_j \xrightarrow{\pi?(\tilde{v})^{ch}} E_i' \oplus E_j'$ is Eprec. This gives us $E_i \xrightarrow{\pi?(\tilde{v})^{ch}} E_i'$ and $E_j \xrightarrow{\pi?(\tilde{v})^{ch}} E_j'$. By our base case we have shown that this transition is non-blocking for broadcast, meaning it also holds for our inductive step.
As we have proven that broadcast is non-blocking for all E_2, we have proven broadcast is also non-blocking on an environment level.□

A.8 Lemma 8

This lemma shows the blocking of multicast channels. Given 2 environments E_1 and E_2 prove:
$ch \neq * \wedge E_1 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \implies E_1 \oplus E_2 \xrightarrow{\pi!(\tilde{v})^{ch}} E_1' \oplus E_2'$ for any E_2 such that for all S_i in E_2:

1. $ch \in \mathsf{LS}_i \cup \ell_i \wedge E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$
   $\vee$
2. $ch \notin \mathsf{LS}_i \cup \ell_i$

We can only apply Elpar to $E_1 \oplus E_2$. We get $E_2 \xrightarrow{\pi?(\tilde{v})^{ch}} E_2'$.

For our base case we take $E_2 = [S_2]^{\mathsf{RC}_2,\ell}_{c_2,n_2}$. By Com we get that it depends on $S \xrightarrow{\pi?(\tilde{v})^{ch}} S'$, which we have proven to be blocking for multicast in Lemma A.3. This means for the base case on an environment level it is also blocking for multicast.

For our inductive step we take $E_2 = E_i \oplus E_j$. The only applicable rule on $E_i \oplus E_j \xrightarrow{\pi?(\tilde{v})^{ch}} E_i' \oplus E_j'$ is Eprec. This gives us $E_i \xrightarrow{\pi?(\tilde{v})^{ch}} E_i'$ and $E_j \xrightarrow{\pi?(\tilde{v})^{ch}} E_j'$. By our base case we have shown that this transition is blocking for multicast, meaning it also holds for our inductive step.
As we have proven that multicast is blocking for all E_2, we have proven multicast is also blocking on an environment level.□

A.9 Lemma 9

This lemma is about the blocking of unicast, to show it should never go unanswered. Given 2 environments E_1 and E_2 prove:
$E_1 \xrightarrow{\pi\mathsf{G}(\tilde{v})^{ch}} E_1' \implies E_1 \oplus E_2 \stackrel{\tau}{\Rightarrow} E_1' \,||\, E_2'$ for any E_2 such that $E_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_2'$.

We can only apply Eluni to $E_1 \oplus E_2$. We get $E_2 \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_2'$.

For our base case we take $E_2 = [S_2]^{\mathsf{RC}_2,\ell}_{c_2,n_2}$. By Com we get that it depends on $S \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} S'$, which we have proven to be blocking for unicast in Lemma A.4. This means for the base case on an environment level it is also blocking for unicast.

For our inductive step we take $E_2 = E_i \oplus E_j$. The only applicable rules on $E_i \oplus E_j \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_i' \oplus E_j'$ are Elsup and Ersup. This gives us $E_i \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_i'$ or $E_j \xrightarrow{\pi\mathsf{S}(\tilde{v})^{ch}} E_j'$. By our base case we have shown that either of these transitions has to happen or otherwise unicast is blocking, meaning it also holds for our inductive step.
As we have proven that unicast is blocking for all E_2, we have proven unicast is also blocking on an environment level.□