INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Spatial Types for Concurrency

A Spatial Logic to Specify and Verify Distributed Systems

Tiago Lança Matos Sucena de Carvalho

Dissertação para obtenção do Grau de Mestre em

Matemática e Aplicações

Júri

Presidente: Prof. Doutora Maria Cristina Sales Viana Serodio Sernadas
Orientador: Prof. Doutor António Maria Alarcão Ravara
Vogais: Prof. Doutor Luís Caires

Outubro 2007

TIPOS ESPACIAIS PARA CONCORRÊNCIA
Lógica Espacial para Especificação e Verificação de Sistemas Distribuídos

NOME: Tiago Lança Matos Sucena de Carvalho

MESTRADO EM: Matemática e Aplicações

ORIENTADOR: Professor Doutor António Maria Alarcão Ravara

RESUMO:
Na ciência da computação, especificar e verificar propriedades comportamentais é considerado um problema clássico. Recentemente, tem surgido um interesse em propriedades espaciais de processos em sistemas distribuídos. Como tal, têm sido propostas lógicas modais cujos sistemas de prova são, em geral, indecidíveis. Uma excepção é o provador automático proposto por Luís Caires [Cai03].

Tradicionalmente, os sistemas de tipos são usados para garantir ausência de erros em linguagens de programação por serem decidíveis e de complexidade reduzida, mas recentemente, também têm sido usados para garantir propriedades espaciais de processos.

O objectivo desta tese é definir uma lógica decidível (sintaxe, semântica e sistema dedutivo) que permita especificar e verificar estaticamente propriedades, nomeadamente invariantes, não só espaciais como comportamentais, em sistemas distribuídos implementados através de processos concorrentes.

Neste trabalho, começa-se por escolher uma linguagem simples de processos e uma linguagem expressiva de fórmulas. A primeira é o fragmento livre de escolha não-determinística do CCS de Milner enquanto que a segunda baseia-se na Lógica Espacial de Caíres e Cardelli [CC03] e na Lógica de Processos de Milner [Mil89]. Adoptando a abordagem "proposições como tipos", estabelece-se uma semântica denotacional, baseada em relações estruturais e de transição, e uma noção de subtipagem, interpretada como implicação lógica. De seguida, define-se como sistema dedutivo um sistema de tipos e prova-se alguns resultados que o caracterizam, nomeadamente um resultado de consistência fraca e um resultado de completude. Por último, desenvolve-se uma aplicação que consiste em derivar uma asserção no sistema e que permite observar o seu potencial.

PALAVRAS-CHAVE: Álgebra de Processos; Concorrente; Comportamental; Lógica Espacial; Tipo; Sistema de tipos.

SPATIAL TYPES FOR CONCURRENCY
A Spatial Logic to Specify and Verify Distributed Systems

ABSTRACT:
The problem of specifying and verifying properties is considered to be a classical problem in computer science. Recently, there has been a growing interest in spatial properties of processes in distributed systems. So, several modal logics have been proposed such that the proof systems are, in general, undecidable. An exception is the model-checker of Luís Caires [Cai03].

Traditionally, type systems are used to guarantee the absence of errors in programming languages due to being decidable and to their low complexity. But recently, they have also been used to ensure spatial properties of processes.

The aim of this thesis is to define a decidable logic (syntax, semantics and deductive system), which allows both spatial and behavioural properties of concurrent processes to be specified and verified, namely invariants.

This work first considers a simple language of processes and an expressive language of formulas. The former is the nondeterministic choice free fragment of a process algebra – Milner's CCS – while the latter is based on the Spatial Logic of Caíres and Cardelli [CC03] and on the Process Logic of Milner [Mil89]. Adopting a "propositions as types" approach, where types are formulas, denotational semantics are established through certain structural and transition relations, and subtyping is interpreted as a certain logical entailment. Furthermore, a type system is defined as a deductive system and some results are proved about it, namely weak consistency and completeness. Finally, a complete application of the system is developed, thus expressing its potential.

KEYWORDS: Behavioural; Concurrency; Process Algebra; Spatial Logic; Type; Type System.

Contents

List of Tables

Preface

The following task was proposed to me by my supervisor António Ravara: "to develop a decidable proof system to statically verify and specify spatial properties of distributed systems". The outcome is the thesis presented here. Eventually a scientific paper will be written and submitted to a journal of mathematics, as was originally intended.

This work was supported by the Space-Time-Types Project[1] and by SQIG[2] at IT[3] (former CLC[4]).

Acknowledgements

I would first like to thank my supervisor António Ravara for his guidance, motivation and constant availability to discuss work. I would also like to thank Luís Caires, for the critical observations and scientific advise given in due time and to Luís Cruz-Filipe, for the revision work and the precious suggestions. Furthermore, a special thanks to my family and friends for their constant support, encouragement and curiosity.

[1] The Space-Time-Types Project, POSC/EIA/55582/2004
[2] Security and Quantum Information Group
[3] Instituto de Telecomunicações
[4] Center for Logic and Computation

Chapter 1

Introduction

1.1 Background

Over the last thirty years, we have observed an impressive spread of new technology. Much of this new technology involves *communicating* or *interacting*. Communication plays a central role within *concurrent systems*, which are composed of several parts that interact. Early on, it was perceived that mathematical models should be developed to describe this type of systems. Around 1980, several proposals were conceived to cope with this need, among these: CSP, the *Communicating Sequential Processes* of Hoare [Hoa85], and CCS, the *Calculus of Communicating Systems* of Milner [Mil89]. These two models, though independently developed, have much in common and are widely known as *process algebras*.

Sometime after, Milner, Parrow and Walker developed a more expressive special kind of process algebra which allows data to be passed through channels – *the π-calculus: a calculus of mobile processes* [PW92]. Commonly known as the π-calculus, it was rapidly established as the new paradigm to model interaction. The special feature of this calculus is its ability to model *mobility*, being also able to represent the systems and data structures that CCS does.

In parallel, a considerable amount of work has been dedicated into the specification and verification of behavioural and temporal properties of concurrent systems, which is considered to be a classical problem in computer science.

Recently, a shift of focus has occurred from the rigid concurrent systems towards distributed systems, followed by a growing interest in spatial properties of distributed systems. For this purpose, several modal and temporal logics have been introduced, such as spatial logics.

Spatial logics support the specification not only of behavioural properties but also of spatial properties of *distributed systems* in a fairly integrated way. Essentially, spatial logics are modal logics that interpret each world as a structured space composed of a certain kind of resources [OP99]. Spatial logics have been used in the definition of several core languages, calculi and data models [LG00, IO01, CC03, GG03]. In [CC03] and in [LG00], spatial logics were developed for process calculi and for the *Ambient Calculus*, respectively. Furthermore, proof systems for verifying spatial properties have already been defined, for instance, in [CC03]; but, in general, these are undecidable. An exception is the model-checker of Luís Caíres [Cai03], which allows the user to automatically verify behavioural and spatial properties of distributed systems expressed in the π-calculus. The algorithm implemented was proved correct for all processes, and complete for a certain class of bounded processes.

Traditionally, type systems were used to guarantee *safety* in programming languages, *i.e.* the absence of errors. Some examples are the λ-calculus [Han94, Pie02], ML [Pie02], and other functional languages. Regarding concurrent processes, several type systems have been proposed involving a notion of behavioural typing [SW01, IK04, RR01, RV00]. Lately, type systems have also been proposed to ensure non-behavioural properties of processes, such as, properties related to the use of resources and the spatial distribution of capabilities. In particular, in [GG03], spatial types were already used to type programs in a functional language for semistructured data.

1.2 Aims

Behavioural properties of processes like *race-freedom* and *deadlock-freedom* are of great importance. But several other interesting properties are inherently spatial, for instance *connectivity*, stating that there exists an access route between two different sites, *unique handling*, stating that there is at most one server process listening on a given channel, or *resource availability*, stating that a bound exists on the number of channels that can be allocated at a given location.

The main purpose of this thesis is to develop a general notion of typing based on spatial logic for concurrent processes, expressive enough to capture both behavioural and spatial properties. Furthermore, we aim a decidable proof system capable of assigning an interesting class of spatial/behavioural types to an interesting class of processes.

Consider a simple process model - a nondeterministic choice free fragment of Milner's CCS [Mil89]. As formulas, consider an expressive set of connectives inspired in the spatial logic of [CC03] and in the modal connectives of Milner's $\mathcal{PL}$ (Process Logic defined in [Mil89]). We adopt a "propositions as types" approach, such that types are formulas, subtyping is interpreted as logical entailment and the type system is seen as a deductive system. Types are partial specifications of the spatial/behavioural properties of processes. We prove that the type system is consistent and complete with respect to a certain semantic notion.

A motivating example: *Looper*

The purpose of the application described bellow is to exhibit an example of spatial invariants that are interesting to specify in the system. Furthermore, it acts as a requirement imposed on the system, meaning that it will often be used to motivate decisions.

Consider the following process

$$Looper \stackrel{\text{def}}{=} (\nu a)(a \mid (\mathbf{rec}X.\bar{a}.(a \mid a \mid X))).$$

Looper is a process which can only perform internal transitions and for each transition it creates a new copy of a private name a. The important feature of this process is that it has always more than one thread running in parallel, *i.e.* it is not a sequential process, and that is the property we want to capture.

More precisely, we want to infer in the system the following typing judgement

$$\Gamma \vdash Looper : \Box(\bar{\mathbf{0}} \mid \bar{\mathbf{0}})$$

where type $\Box(\bar{\mathbf{0}} \mid \bar{\mathbf{0}})$ specifies that *Looper* will always have at least two separate threads running in parallel.

This example will be further developed in chapter 5.

1.3 Contributions

The main contribution of this thesis is the definition of a logic, where:

- the chosen syntax of formulas is inspired in both spatial logic of [CC03] and process logic of [Mil89], and provides expressiveness without sacrificing decidability;
- the semantics, of a denotational kind, interpret spatial connectives mainly through structural congruence and behavioural connectives through a labelled transition relation;
- the deductive system, which is a type system, decomposed in three parts:
  - a structural part containing structural inference rules;
  - a subsumption rule which introduces an axiomatic subtyping relation into the system; and
  - an infinitary conjunction rule.

1.4 Outline of the thesis

This thesis is organized in four parts briefly described as such:

- Chapter 2 introduces a calculus of processes based upon Milner's CCS [Mil89];
- Chapter 3 develops a type system presenting several inference rules that compose it, namely a subsumption rule that introduces subtyping in the system;
- Chapter 4 consists of a series of results which describe properties of the system, namely consistency and completeness results;
- Chapter 5 presents a derivation of a complete example of application of the type system.
- Chapter 6 describes the achievements of this work and the future possible developments.

Chapter 2

Calculus of Processes

The purpose of this chapter is to introduce the calculus of processes considered in the remaining chapters. This calculus is, in fact, a fragment of Milner's CCS [Mil89]. In section 2.1 the syntax of the calculus is presented and the operators are briefly explained. In section 2.2 the operational semantics is introduced, thus the operators are rendered meaning.

2.1 Syntax of processes

Consider a countable set of names, $\mathcal{A}$, ranged over by $a, b, c, ...$ and let $\overline{\mathcal{A}}$ denote the set of co-names, ranged over by $\bar{a}, \bar{b}, \bar{c}, ...$ Set $\mathcal{L} \triangleq \mathcal{A} \cup \overline{\mathcal{A}}$, where $\mathcal{L}$ is the set of labels, and let $Act \triangleq \mathcal{L} \cup \{\tau\}$ where τ is the silent or invisible action. As in CCS, we say that $\bar{a}$ is the *complement* of a and we extend complementation to all actions by defining $\bar{\bar{a}} = a$ and $\bar{\tau} = \tau$. Further, consider a countable set of *process variables*, $\mathcal{X}$, disjoint from the previous sets.

The following definition sets a language of processes.

Definition 2.1.1. (Syntax of processes) *The following grammar defines the set $\mathcal{P}$ of* processes.

Labels	α	$::=$	$a \mid \overline{a}$	
Actions	π	$::=$	$\alpha \mid \tau$	
Processes	P	$::=$	$\mathbf{0}$	*inaction*
		$\mid$	$\pi.P$	*prefix*
		$\mid$	$P \mid P$	*parallel composition*
		$\mid$	$(\nu a)P$	*restriction*
		$\mid$	$(\mathbf{rec}X.P)$	*recursion*
		$\mid$	X	*process variable*

The processes produced by this grammar constitute a fragment of CCS. Summations are left out for reasons of simplicity which will become more apparent further ahead. Thus we have the inactive process $\mathbf{0}$, action prefixing $\pi.P$ for sequential behaviour, parallel composition $P \mid Q$ to express concurrency, restricted names $(\nu a)P$ and recursion $(\mathbf{rec}X.P)$. Prefixed operations ($\alpha.$, (νa) and $\mathbf{rec}X.$) bind more tightly than composition.

Also, it is often convenient to omit '.$\mathbf{0}$'; for example we write $a.b$ for $a.b.\mathbf{0}$.

In name restriction $(\nu a)P$ and in recursion $(\textbf{rec}X.P)$, the distinguished occurrences of the name a and the variable X are binding, both with scope P.

Recursive types are supposed to describe regular trees: a process tree should be "read" as the infinite unfolding of a recursive process. But there are processes which can not be reasonably interpreted as representations of regular trees; for example, the unfolding of the process $P = (\textbf{rec}X.X)$ gives P again, which can not be "read" as a tree. Therefore, to achieve this tight correspondence we must restrict the use of recursive processes. With this purpose consider first the following definition.

Definition 2.1.2. (Subexpression of a Process) *Given a process P, we denote by $sub(P)$ the set inductively defined as follows:*

$$sub(\mathbf{0}) \triangleq \{\mathbf{0}\}$$

$$sub((\pi.P')) \triangleq \{(\pi.P')\} \cup sub(P')$$

$$sub((P_1 \mid P_2)) \triangleq \{(P_1 \mid P_2)\} \cup sub(P_1) \cup sub(P_2)$$

$$sub(((\nu a)P')) \triangleq \{((\nu a)P')\} \cup sub(P')$$

$$sub((\textbf{rec}X.P')) \triangleq \{(\textbf{rec}X.P')\} \cup sub(P')$$

$$sub(X) \triangleq \{X\}$$

The next definition, adapted from [Pie02], induces the intended restriction.

Definition 2.1.3. (Contractive Process) *A process P is* contractive *if, for any subexpression of P of the form $(\textbf{rec}X.(\textbf{rec}X_1....(\textbf{rec}X_n.Q)))$, the body Q is not X.*

In the following we will restrict our language to contractive processes.

2.2 Operational semantics

Following the work of Milner [Mil89, Mil99], this section defines the operational semantics of the process calculus via two binary relations: a static one - the *structural congruence relation* - and a dynamic one - a *labelled transition relation*.

2.2.1 Names, variables and substitutions in processes

We first define the notions of *free* and *bound* names and variables, along with the notions of *substitution* of names and variables in processes, which are used in defining the structural congruence and the labelled transition system relations.

Definition 2.2.1. (Free and bound names in processes) *For any process P, the set of* free *names of P, written $fn(P)$, and the set of* bound *names of P, written $bn(P)$, are inductively defined as follows:*

$$
\begin{array}{ll}
fn(\mathbf{0}) \triangleq \emptyset & bn(\mathbf{0}) \triangleq \emptyset \\
fn(a.P) \triangleq fn(P) \cup \{a\} & bn(\pi.P') \triangleq bn(P') \\
fn(\bar{a}.P) \triangleq fn(P) \cup \{a\} & bn(P_1 \mid P_2) \triangleq bn(P_1) \cup bn(P_2) \\
fn(\tau.P) \triangleq fn(P) & bn((\nu a)P') \triangleq \{a\} \cup bn(P') \\
fn(P \mid Q) \triangleq fn(P) \cup fn(Q) & bn((\textbf{\textit{rec}}X.P')) \triangleq bn(P') \\
fn((\nu a)P) \triangleq fn(P) \setminus \{a\} & bn(X) \triangleq \emptyset \\
fn((\textbf{\textit{rec}}X.P)) \triangleq fn(P) & \\
fn(X) \triangleq \emptyset &
\end{array}
$$

We say that a name is fresh in a process if it is neither bound nor free in that process. The set of names appearing in a process P, written $n(P)$, is the union of the free and bound names of P.

Definition 2.2.2. (Free and bound variables in processes) *For any process P, the set of* free *variables of P, written $fv(P)$, and the set of bound variables of P, written $bv(P)$, are inductively defined as follows:*

$$
\begin{array}{ll}
fv(\mathbf{0}) \triangleq \emptyset & bv(\mathbf{0}) \triangleq \emptyset \\
fv(\pi.P') \triangleq fv(P') & bv(\pi.P') \triangleq bv(P') \\
fv(P_1 \mid P_2) \triangleq fv(P_1) \cup fv(Q_2) & bv(P_1 \mid P_2) \triangleq bv(P_1) \cup bv(P_2) \\
fv((\nu a)P') \triangleq fv(P') & bv((\nu a)P') \triangleq bv(P') \\
fv((\textbf{\textit{rec}}X.P')) \triangleq fv(P') \setminus \{X\} & bv((\textbf{\textit{rec}}X.P')) \triangleq bv(P') \cup \{X\} \\
fv(X) \triangleq \{X\} & bv(X) \triangleq \emptyset
\end{array}
$$

We say that a variable is fresh in a process if it is neither bound nor free in that process. The set of variables appearing in a process P, written $v(P)$, is the union of the free and bound variables of P.

In the following we will need to substitute names for names and variables for processes in processes. Therefore, we write $P\{a/b\}$ and $P\{Q/X\}$ for the operation which replaces all occurrences of b and X for a and Q in P, respectively. Moreover, when we write $P\{a/b\}\{c/d\}$, we mean $(P\{a/b\})\{c/d\}$, *i.e.* the substitutions are sequential, not simultaneous. The next two definitions make it precise.

Definition 2.2.3. (Substitution of names) *Given a process P and names a and b, we denote by $P\{a/b\}$ the process inductively defined as follows:*

$\mathbf{0}\{a/b\} \triangleq \mathbf{0}$

$(b.P')\{a/b\} \triangleq a.(P'\{a/b\})$

$(\bar{b}.P')\{a/b\} \triangleq \bar{a}.(P'\{a/b\})$

$(\pi.P')\{a/b\} \triangleq \pi.(P'\{a/b\})$, *if* $\pi \neq b, \bar{b}$

$(P_1 \mid P_2)\{a/b\} \triangleq (P_1\{a/b\}) \mid (P_2\{a/b\})$

$((\nu a)P')\{a/b\} \triangleq (\nu c)(P'\{c/a\}\{a/b\})$, *where c is fresh in P'*

$((\nu b)P')\{a/b\} \triangleq (\nu a)(P'\{a/b\})$, *where* $a \notin fn(P')$

$((\nu c)P')\{a/b\} \triangleq (\nu c)(P'\{a/b\})$, *where* $c \neq a, b$

$(\textbf{\textit{rec}}X.P')\{a/b\} \triangleq (\textbf{\textit{rec}}X.(P'\{a/b\}))$

$X\{a/b\} \triangleq X$

Definition 2.2.4. (Substitution of variables) *Given processes P and Q, we denote by $P\{Q/X\}$ the process inductively defined as follows:*

$\mathbf{0}\{Q/X\} \triangleq \mathbf{0}$

$(\pi.P')\{Q/X\} \triangleq \pi.(P'\{Q/X\})$

$(P_1 \mid P_2)\{Q/X\} \triangleq (P_1\{Q/X\}) \mid (P_2\{Q/X\})$

$((\nu a)P')\{Q/X\} \triangleq (\nu a)(P'\{Q/X\})$

$(\textbf{\textit{rec}}Y.P')\{Y/X\} \triangleq (\textbf{\textit{rec}}Z.(P'\{Z/Y\}\{Y/X\}))$, *where Z is fresh in P'*

$(\textbf{\textit{rec}}X.P')\{Y/X\} \triangleq (\textbf{\textit{rec}}Y.(P'\{Y/X\}))$, *if* $Y \notin fv(P')$

$(\textbf{\textit{rec}}Y.P')\{Q/X\} \triangleq (\textbf{\textit{rec}}Y.(P'\{Q/X\}))$, *where* $Q \neq Y$

$Y\{Q/X\} \triangleq Y$

$X\{Q/X\} \triangleq Q$

2.2.2 Structural congruence on processes

Within a process, changing a bound name or variable into fresh ones must not alter it in terms of behaviour. Thus, in a similar fashion as in [Mil99], we relate processes which differ only in that a replacement of this kind has occurred. This is called $alpha$-*conversion* and it is formally defined next.

Definition 2.2.5. (α-conversion) *α-conversion, $\equiv_\alpha$, is the least congruence on processes such that*

$(\nu a)P \equiv_\alpha (\nu b)(P\{b/a\})$, *where b is fresh in P* *(Alpha Res)*

$(\textbf{\textit{rec}}X.P) \equiv_\alpha (\textbf{\textit{rec}}Y.(P\{Y/X\}))$, *where Y is fresh in P* *(Alpha Rec)*

The next definition is largely inspired on [Mil99].

Definition 2.2.6. (Structural congruence) *Structural congruence, $\equiv$, is the least congruence on processes such that*

1. *If* $P \equiv_\alpha Q$, *then* $P \equiv Q$
2. $P \mid \mathbf{0} \equiv P$
3. $P \mid Q \equiv Q \mid P$
4. $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$

5. $(\nu a)\mathbf{0} \equiv \mathbf{0}$

6. $(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$

7. $(\nu a)(P \mid Q) \equiv ((\nu a)P \mid Q)$, *if* $a \notin fn(Q)$

Processes which are α-convertible are also structurally congruent; laws (2)-(4) show that the set of processes equipped with the parallel composition forms an abelian monoid with inaction as its neutral element; restricting a name in an inactive process or changing the order of restriction of the names is irrelevant; the last law states that parts of a process not containing a name a can be included in the scope of a (νa) or not, with no difference.

The following lemma is presented without a proof as it is a straightforward property of structural congruence. A proof can easily be found in any book (*cf.* [Mil89], [Mil99]).

Lemma 2.2.7. *If* $P \equiv Q$, *then* $fn(P) = fn(Q)$.

2.2.3 Labelled transition system

To express the behaviour of processes we include a *labelled transition system* that contains one or more rules for each combinator of the calculus.

First, we give a standard general definition of labelled transition system.

Definition 2.2.8. (Labelled Transition System) *A Labelled Transition System over a set of actions* $\mathcal{A}$ *is a pair* $(\mathcal{Q}, \mathcal{T})$ *consisting of*

- *a set* $\mathcal{Q}$ *of states;*
- *a ternary relation* $\mathcal{T} \subseteq (\mathcal{Q} \times \mathcal{A} \times \mathcal{Q})$, *known as a* transition relation.

Moreover, if $(q, \alpha, q') \in \mathcal{T}$ *we write* $q \xrightarrow{\alpha} q'$, *and we call* q *the* source *and* q' *the target of the transition. If* $q \xrightarrow{\pi} q'$, *then* q' *is a* π-derivative *of* q.

Now, consider the following labelled transition system for the calculus. From this point further we will often refer to it simply as the LTS.

Definition 2.2.9. (Labelled Transition System for concurrent processes) *The Labelled Transition System* $(\mathcal{P}, \mathcal{T})$ *of concurrent processes over Act has* $\mathcal{P}$ *as its states and its transitions* $\mathcal{T}$ *are exactly those which can be inferred by the rules listed on Table 2.1.*

The only operator of the calculus which does not have a rule associated with it is inaction $\mathbf{0}$. This results from the fact that the inactive process can not perform any transition. In all rules a transition of a composite process can be inferred from transitions of its components. Remember that α can be of the form a or $\bar{a}$, but not τ.

This set of rules is in fact an adaptation of the labelled transition system for concurrent processes of [Mil89]. In addition it includes rule SC which appears in the reduction relation for the π-calculus of [SW01]. It allows restructuring both before and after a transition and the use of the rule after a transition ensures that the transition relation is closed under structural congruence, in the sense that if $P \xrightarrow{\pi} Q$ and $R \equiv Q$, then $P \xrightarrow{\pi} R$.

$$\frac{}{\pi.P \xrightarrow{\pi} P}\ \text{Act} \qquad \frac{P \xrightarrow{\pi} P'}{P \mid Q \xrightarrow{\pi} P' \mid Q}\ \text{Par1}$$

$$\frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\bar{\alpha}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}\ \text{Par2}$$

$$\frac{P \xrightarrow{\pi} P'}{(\nu a)P \xrightarrow{\pi} (\nu a)P'}, \pi, \bar{\pi} \neq a \quad \text{Res} \qquad \frac{P\{(\mathbf{rec}X.P)/X\} \xrightarrow{\pi} P'}{(\mathbf{rec}X.P) \xrightarrow{\pi} P'}\ \text{Rec}$$

$$\frac{P \equiv P' \xrightarrow{\pi} Q' \equiv Q}{P \xrightarrow{\pi} Q}\ \text{SC}$$

Table 2.1: The Labelled Transition System.

Chapter 3

Type System

This chapter presents the logic in the following way. Section 3.1 introduces the syntax of formulas. Section 3.2 defines the semantics of formulas, where, in subsection 3.3.2, subtyping relation and subtyping derivation are introduced. Finally, section 3.3 presents the deductive system, *i.e.* the type system.

3.1 Syntax of types

The type language is largely inspired in the *Spatial Logic* of [CC03] and in the *Process Logic* described in [Mil89]. Some type operators are closely related with spatial logic operators and the others, the modal and propositional ones, are taken from Milner's process logic. The former aspire to express spatial properties of processes while the latter are meant to express behavioural properties of processes.

Consider sets $\mathcal{A}$, $\overline{\mathcal{A}}$, $\mathcal{L}$ and *Act* as defined in 2.1. In addition consider a countable set of type variables, $\mathcal{Z}$, disjoint from these.

Definition 3.1.1. (Syntax of types) *Let I be a nonempty countable index set. The following grammar defines the set $\mathcal{T}$ of* types.

$$
\begin{array}{rcll}
A & ::= & \mathbf{0} & \textit{inaction} \\
 & | & \pi.A & \textit{prefix} \\
 & | & A \mid A & \textit{parallel composition} \\
 & | & a\textcircled{R}A & \textit{restriction} \\
 & | & (\textbf{\textit{rec}}\mathcal{Z}.A) & \textit{recursion} \\
 & | & \mathcal{Z} & \textit{type variable} \\
 & | & \bar{\mathbf{0}} & \textit{active} \\
 & | & [\pi]\,A & \textit{if } \pi \\
 & | & \langle\pi\rangle\,A & \textit{may } \pi \\
 & | & \bigwedge_{i\in I} A_i & \textit{conjunction} \\
 & | & \bigvee_{i\in I} A_i & \textit{disjunction}
\end{array}
$$

Note as a first observation, that this grammar does not contain any form of negation, though it is possible to define; [CC03], is an example of a closely related logic that uses negation. But, instead of using negation to define other connectives, all dual operators are taken as primitives in the language; this is meant this way

for reasons of decidability.

Every operator in the calculus of processes has a counterpart among types: inaction, prefix, parallel composition and recursion combinators are also defined within types. Furthermore, restriction of names in types is also defined, though with a different notation taken from [CC03] – $a\circledR A$. We call these type combinators the *structural operators*. We have the active type $\bar{\mathbf{0}}$, the dual of $\mathbf{0}$; two modalities: $[\pi]\,A$ to express the type of a process after performing any transition and $\langle\pi\rangle\,A$ to express the possibility of transition by a process and its type after that transition; conjunction and disjunction to enable the definition of more expressive modalities.

All these operators will be rendered meaning when we define the *satisfaction* relation below.

Before moving on we must restrict the language of types. As in section 2.1, for recursive types to be interpreted as regular trees a restriction must be assumed. Accordingly, consider the next two definitions.

Definition 3.1.2. (Subexpression of a Type) *Given a type A, we denote by $sub(A)$ the set inductively defined as follows:*

$sub(\mathbf{0}) \triangleq \{\mathbf{0}\}$

$sub(\bar{\mathbf{0}}) \triangleq \{\bar{\mathbf{0}}\}$

$sub(\pi.A') \triangleq \{\pi.A'\} \cup sub(A')$

$sub(A_1 \mid A_2) \triangleq \{A_1 \mid A_2\} \cup sub(A_1) \cup sub(A_2)$

$sub(\pi\circledR A') \triangleq \{\pi\circledR A'\} \cup sub(A')$

$sub((\textbf{\textit{rec}}\,Z.A')) \triangleq \{(\textbf{\textit{rec}}\,Z.A')\} \cup sub(A')$

$sub(Z) \triangleq \{Z\}$

$sub([\pi]\,A') \triangleq \{[\pi]\,A'\} \cup sub(A')$

$sub(\langle\pi\rangle\,A') \triangleq \{\langle\pi\rangle\,A'\} \cup sub(A')$

$sub(\bigwedge_{i\in I} A_i) \triangleq \{\bigwedge_{i\in I} A_i\} \cup \bigcup_{i\in I} sub(A_i)$

$sub(\bigvee_{i\in I} A_i) \triangleq \{\bigvee_{i\in I} A_i\} \cup \bigcup_{i\in I} sub(A_i)$

Definition 3.1.3. (Contractive type) *A type A is* contractive *if, for any subexpression of A of the form $(\textbf{\textit{rec}}\,Z.(\textbf{\textit{rec}}\,Z_1....(\textbf{\textit{rec}}\,Z_n.B)))$, the body B is not Z.*

In the following we will restrict our language to contractive types.

3.2 Semantics

3.2.1 Structural congruence on types

In an analogous way as in for processes we also need to define the notions of *free* and *bound* names and variables, along with the notions of *substitution* of names and variables in types.

Definition 3.2.1. (Free and bound names in types) *For any type A, the set of* free *names of A, written $fn(A)$, and the set of* bound *names of A, written $bn(A)$, are inductively defined as follows:*

$$
\begin{array}{ll}
fn(\mathbf{0}) \triangleq \emptyset & bn(\mathbf{0}) \triangleq \emptyset \\
fn(\bar{\mathbf{0}}) \triangleq \emptyset & bn(\bar{\mathbf{0}}) \triangleq \emptyset \\
fn(\pi.A') \triangleq \{\pi\} \cup fn(A') & bn(\pi.A') \triangleq bn(A') \\
fn(A_1 \mid A_2) \triangleq fn(A_1) \cup fn(A_2) & bn(A_1 \mid A_2) \triangleq bn(A_1) \cup bn(A_2) \\
fn(a\circledR A') \triangleq fn(A') \setminus \{a\} & bn(a\circledR A') \triangleq bn(A') \cup \{a\} \\
fn((\textbf{\textit{rec}}\mathcal{Z}.A')) \triangleq fn(A') & bn((\textbf{\textit{rec}}\mathcal{Z}.A')) \triangleq bn(A') \\
fn(\mathcal{Z}) \triangleq \emptyset & bn(\mathcal{Z}) \triangleq \emptyset \\
fn(\langle\pi\rangle A') \triangleq \{\pi\} \cup fn(A') & bn(\langle\pi\rangle A') \triangleq bn(A') \\
fn([\pi] A') \triangleq \{\pi\} \cup fn(A') & bn([\pi] A') \triangleq bn(A') \\
fn(\bigwedge_{i\in I} A_i) \triangleq \bigcup_{i\in I} fn(A_i) & bn(\bigwedge_{i\in I} A_i) \triangleq \bigcup_{i\in I} bn(A_i) \\
fn(\bigvee_{i\in I} A_i) \triangleq \bigcup_{i\in I} fn(A_i) & bn(\bigvee_{i\in I} A_i) \triangleq \bigcup_{i\in I} bn(A_i)
\end{array}
$$

We say that a name is fresh in a type if it is neither bound nor free in that type. The set of names appearing in a type A, written $nt(A)$, is the union of the free and bound names of A.

Definition 3.2.2. (Free and bound variables in Types) *For any type A, the set of* free *variables of A, written $fv(A)$, and the set of* bound *variables of A, written $bv(A)$, are inductively defined as follows:*

$$
\begin{array}{ll}
fv(\mathbf{0}) \triangleq \emptyset & bv(\mathbf{0}) \triangleq \emptyset \\
fv(\bar{\mathbf{0}}) \triangleq \emptyset & bv(\bar{\mathbf{0}}) \triangleq \emptyset \\
fv(\pi.A') \triangleq fv(A') & bv(\pi.A') \triangleq bv(A') \\
fv(A_1 \mid A_2) \triangleq fv(A_1) \cup fv(A_2) & bv(A_1 \mid A_2) \triangleq bv(A_1) \cup bv(A_2) \\
fv(\pi\circledR A') \triangleq fv(A') & bv(\pi\circledR A') \triangleq bv(A') \\
fv((\textbf{\textit{rec}}\mathcal{Z}.A')) \triangleq fv(A') \setminus \{\mathcal{Z}\} & bv((\textbf{\textit{rec}}\mathcal{Z}.A')) \triangleq bv(A') \cup \{\mathcal{Z}\} \\
fv(\mathcal{Z}) \triangleq \{\mathcal{Z}\} & bv(\mathcal{Z}) \triangleq \emptyset \\
fv(\langle\pi\rangle A') \triangleq fv(A') & bv(\langle\pi\rangle A') \triangleq bv(A') \\
fv([\pi] A') \triangleq fv(A') & bv([\pi] A') \triangleq bv(A') \\
fv(\bigwedge_{i\in I} A_i) \triangleq \bigcup_{i\in I} fv(A_i) & bv(\bigwedge_{i\in I} A_i) \triangleq \bigcup_{i\in I} bv(A_i) \\
fv(\bigvee_{i\in I} A_i) \triangleq \bigcup_{i\in I} fv(A_i) & bv(\bigvee_{i\in I} A_i) \triangleq \bigcup_{i\in I} bv(A_i)
\end{array}
$$

We say that a variable is fresh in a type if it is neither bound nor free in that type. The set of variables appearing in a type A, written $vt(A)$, is the union of the free and bound variables of A.

The next two definitions extend the operation, introduced in section 2.2, of substituting names for names and variables for variables, to types. Therefore, we write $A\{a/b\}$ and $A\{B/\mathcal{Z}\}$ for the operation which replaces all occurrences of b and $\mathcal{Z}$ for a and B in A, respectively.

Definition 3.2.3. (Substitution of names in types) *Given a type A and names a and b, we denote by $A\{a/b\}$ the type inductively defined as follows:*

$$\mathbf{0}\{a/b\} \triangleq \mathbf{0}$$

$$\bar{\mathbf{0}}\{a/b\} \triangleq \bar{\mathbf{0}}$$

$$(b.A')\{a/b\} \triangleq a.(A'\{a/b\})$$

$(\bar{b}.A')\{a/b\} \triangleq \bar{a}.(A'\{a/b\})$

$(\pi.A')\{a/b\} \triangleq \pi.(A'\{a/b\})$, *if* $\pi \neq b, \bar{b}$

$(A_1 \mid A_2)\{a/b\} \triangleq (A_1\{a/b\}) \mid (A_2\{a/b\})$

$(a\circledR A')\{a/b\} \triangleq c\circledR(A'\{c/a\}\{a/b\})$, *where* $c \notin fn(A')$

$b\circledR A')\{a/b\} \triangleq b\circledR(A'\{a/b\})$, *where* $b \notin fn(A')$

$(c\circledR A')\{a/b\} \triangleq c\circledR(A'\{a/b\})$, *where* $c \neq a, b$

$(\mathbf{rec}\mathcal{Z}.A')\{a/b\} \triangleq (\mathbf{rec}\mathcal{Z}.(A'\{a/b\}))$

$\mathcal{Z}\{a/b\} \triangleq \mathcal{Z}$

$([b]\,A')\{a/b\} \triangleq [a]\,(A'\{a/b\})$

$([\bar{b}]\,A')\{a/b\} \triangleq [\bar{a}]\,(A'\{a/b\})$

$([\pi]\,A')\{a/b\} \triangleq [\pi]\,(A'\{a/b\})$, *if* $\pi \neq b, \bar{b}$

$(\langle b\rangle\,A')\{a/b\} \triangleq \langle a\rangle\,(A'\{a/b\})$

$(\langle \bar{b}\rangle\,A')\{a/b\} \triangleq \langle \bar{a}\rangle\,(A'\{a/b\})$

$(\langle \pi\rangle\,A')\{a/b\} \triangleq \langle \pi\rangle\,(A'\{a/b\})$, *if* $\pi \neq b, \bar{b}$

$(\bigwedge_{i\in I} A_i)\{a/b\} \triangleq \bigwedge_{i\in I}(A_i\{a/b\})$

$(\bigvee_{i\in I} A_i)\{a/b\} \triangleq \bigvee_{i\in I}(A_i\{a/b\})$

Definition 3.2.4. (Substitution of variables in types) *Given types* A *and* B*, we denote by* $A\{B/\mathcal{Z}\}$ *the type inductively defined as follows:*

$\mathbf{0}\{B/\mathcal{Z}\} \triangleq \mathbf{0}$

$\bar{\mathbf{0}}\{B/\mathcal{Z}\} \triangleq \bar{\mathbf{0}}$

$\pi.A'\{B/\mathcal{Z}\} \triangleq \pi.(A'\{B/\mathcal{Z}\})$

$(A_1 \mid A_2)\{B/\mathcal{Z}\} \triangleq (A_1\{B/\mathcal{Z}\}) \mid (A_2\{B/\mathcal{Z}\})$

$(a\circledR A')\{B/\mathcal{Z}\} \triangleq a\circledR(A'\{B/\mathcal{Z}\})$

$(\mathbf{rec}\mathcal{W}.A')\{\mathcal{W}/\mathcal{Z}\} \triangleq (\mathbf{rec}\mathcal{X}.(A'\{\mathcal{X}/\mathcal{W}\}\{\mathcal{W}/\mathcal{Z}\}))$, *where* $\mathcal{X}$ *is fresh in* A'

$(\mathbf{rec}\mathcal{Z}.A')\{\mathcal{W}/\mathcal{Z}\} \triangleq (\mathbf{rec}\mathcal{W}.(A'\{\mathcal{W}/\mathcal{Z}\}))$, *where* $\mathcal{W} \notin fv(A')$

$(\mathbf{rec}\mathcal{W}.A')\{B/\mathcal{Z}\} \triangleq (\mathbf{rec}\mathcal{W}.(A'\{B/\mathcal{Z}\}))$, *where* $B \neq \mathcal{W}$

$\mathcal{W}\{B/\mathcal{Z}\} \triangleq \mathcal{W}$

$\mathcal{Z}\{B/\mathcal{Z}\} \triangleq B$

$[\pi]\, A'\{B/\mathcal{Z}\} \triangleq [\pi]\, (A'\{B/\mathcal{Z}\})$

$\langle\pi\rangle\, A'\{B/\mathcal{Z}\} \triangleq \langle\pi\rangle\, (A'\{B/\mathcal{Z}\})$

$(\bigwedge_{i\in I} A_i)\{B/\mathcal{Z}\} \triangleq \bigwedge_{i\in I} (A_i\{B/\mathcal{Z}\})$

$(\bigvee_{i\in I} A_i)\{B/\mathcal{Z}\} \triangleq \bigvee_{i\in I} (A_i\{B/\mathcal{Z}\})$

As the structural type operators are the same as the calculus operators it is natural to define α-conversion and structural congruence over types as well. This is done in a similar way as in section 2.2.2.

Definition 3.2.5. **(α-conversion on types)** *α-conversion on types, $\equiv^t_\alpha$, is the least congruence on types such that*

$a\circledR A \equiv^t_\alpha b\circledR(A\{b/a\})$, *where b is fresh in A* *(t-Alpha Res)*

$(\mathbf{rec}\mathcal{Z}.A) \equiv^t_\alpha (\mathbf{rec}\mathcal{W}.(A\{\mathcal{W}/\mathcal{Z}\}))$, *where $\mathcal{W}$ is fresh in A* *(t-Alpha Rec)*

Definition 3.2.6. **(Structural congruence on types)** *Types-structural congruence, $\equiv^t$, is the least congruence on types such that*

1. *If $A \equiv^t_\alpha B$, then $A \equiv^t B$*
2. $A \mid \mathbf{0} \equiv^t A$.
3. $A \mid B \equiv^t B \mid A$.
4. $A \mid (B \mid C) \equiv^t (A \mid B) \mid C$.
5. $a\circledR\mathbf{0} \equiv^t \mathbf{0}$.
6. $a\circledR b\circledR A \equiv^t b\circledR a\circledR A$.
7. $a\circledR(A \mid B) \equiv^t (a\circledR A \mid B)$, *if $a \notin fn(B)$*.

So, types which are α-convertible are also structuraly congruent; laws (2)-(4) show that the set of types equipped with the parallel composition forms an abelian monoid with inaction as its neutral element; restricting a name in an inactive type or changing the order of restriction of the names is irrelevant; the last law states that parts of a type not containing a name a can be included in the scope of a $a\circledR$ or not, with no difference.

3.2.2 Interpretation of types

After having formally defined a language of formulas (types), we introduce below a new semantics using the notions of structural congruence and of labelled transition.

First, we give the following three definitions about processes, taken from [Mil99].

Definition 3.2.7. (Experiment relations) *The relations $\Rightarrow$ and $\stackrel{s}{\Rightarrow}$, for any $s \in Act^*$, are defined as follows:*

1. *$P \Rightarrow Q$ means that there is a sequence of zero or more τ transitions $P \stackrel{\tau}{\rightarrow} \ldots \stackrel{\tau}{\rightarrow} Q$. Formally, $\Rightarrow \stackrel{\text{def}}{=} (\stackrel{\tau}{\rightarrow})^*$, is the transitive reflexive closure of $\stackrel{\tau}{\rightarrow}$.*
2. *Let $s = \pi_1 \ldots \pi_n$. Then $P \stackrel{s}{\Rightarrow} Q$ means $P \Rightarrow \stackrel{\pi_1}{\longrightarrow} P_1 \cdots \Rightarrow \stackrel{\pi_n}{\longrightarrow} P_n \Rightarrow Q$. Formally, $\stackrel{s}{\Rightarrow} \stackrel{\text{def}}{=} \Rightarrow \stackrel{\pi_1}{\longrightarrow} \Rightarrow \cdots \Rightarrow \stackrel{\pi_n}{\longrightarrow} \Rightarrow$.*

For $s \in Act*$, if $P \stackrel{s}{\Rightarrow} P'$, then P' is an s-descendant *of P.*

Definition 3.2.8. (Weak Simulation) *Let $\mathcal{S}$ be a binary relation over $\mathcal{P}$. Then $\mathcal{S}$ is said to be a* weak simulation *if, whenever $P\mathcal{S}Q$,*

if $P \stackrel{e}{\Rightarrow} P'$ then there exists $Q' \in \mathcal{P}$ such that $Q \stackrel{e}{\Rightarrow} Q'$ and $P'\mathcal{S}Q'$.

We say that Q weakly simulates *P, written $P <: Q$, if there exists a weak simulation $\mathcal{S}$ such that $P\mathcal{S}Q$.*

Definition 3.2.9. (Weak Bisimulation) *A binary relation $\mathcal{S}$ over $\mathcal{P}$ is said to be a* weak bisimulation *if both $\mathcal{S}$ and its converse are weak simulations. We say that P and Q are* weakly bisimilar*, written $P \approx Q$, if there exists a weak bisimulation $\mathcal{S}$ such that $P\mathcal{S}Q$.*

A reason not to consider summations in the language of types becomes apparent after the next result.

Proposition 3.2.10. (Preservation) *Simulation is preserved by all process operators.*

In [Mil99] it was proved that bisimulation is preserved by all operators except summations. Therefore, the work is considerably simplified. In particular, it enables the previous helpful result.

A simple to prove result relating structural congruence and bisimulation is presented below. It is usually known as the "Harmony Lemma" and a proof of it can be found in [Mil89].

Lemma 3.2.11. (Harmony Lemma) *Structural congruence is a bisimulation.*

The following direct corollary of this result will be useful further ahead.

Corollary 3.2.12. (SC is a Simulation) *If $P \equiv Q$, then $P <: Q$.*

Proof. By Proposition 3.2.11, if $P \equiv Q$, then $P \approx Q$. Then, there exists a weak bisimulation $\mathcal{S}$ such that $P\mathcal{S}Q$. Moreover, by definition of weak bisimulation, $\mathcal{S}$ and its converse are weak simulations. Therefore, $P <: Q$. □

The semantics of types is defined by assigning to each type A a set of processes, namely the set of processes which satisfy the property denoted by A. With this purpose, we introduce a binary relation between processes and types, named *satisfaction*, which induces the mentioned set. But before formally defining satisfaction we need to mention the notion of *environment*, denoted by Γ, needed to cater for free variables within the body of recursion, which is an injective function mapping process variables into type variables. *Satisfaction*, denoted by $\models_\Gamma$, is formalized in the following definition.

Definition 3.2.13. (Interpretation of Types) *The satisfaction relation, $\models_\Gamma$, is inductively defined over types by the following rules:*

Voi $P \models_\Gamma \mathbf{0}$, *if* $P \stackrel{\pi}{\not\longrightarrow}$, *for every* π;

Non $P \models_\Gamma \bar{\mathbf{0}}$, *if* $P \stackrel{\pi}{\longrightarrow} P'$, *for some* π *and* P';

Var $P \models_\Gamma \mathcal{Z}$, *if* P *is a variable* X *and* $X : \mathcal{Z}$ *is in* Γ;

Par $P \models_\Gamma A \mid B$, *if* $P \equiv Q \mid R, Q \models_\Gamma A$ *and* $R \models_\Gamma B$, *for some* Q *and* R;

Pre $P \models_\Gamma \pi.A$, *if for some* P' $P \stackrel{\pi}{\longrightarrow} P'$ *and* $P' \models_\Gamma A$, *and, for every* π' *and* P'', *if* $P \stackrel{\pi'}{\longrightarrow} P''$ *then* π' *is* π *and,* P'' *and* P' *are weakly bisimilar*;

New $P \models_\Gamma a\circledR A$, *if* $P \equiv (\nu a)Q$ *and* $Q \models_\Gamma A$, *for some* Q;

Rec $P \models_\Gamma (\mathbf{rec}\mathcal{Z}.A)$, *if* $P \equiv (\mathbf{rec}X.P')$ *and* $P' \models_{\Gamma,\{X:\mathcal{Z}\}} A'$, *for some* X *and* P';

Dia $P \models_\Gamma \langle\pi\rangle A$, *if* $P \stackrel{\pi}{\longrightarrow} P'$ *and* $P' \models_\Gamma A$, *for some* P';

Box $P \models_\Gamma [\pi] A$, *if* $P \models_\Gamma \pi.A$ *or* $P \stackrel{\pi}{\not\longrightarrow}$;

Con $P \models_\Gamma \bigwedge_{i\in I} A_i$, *if* $P \models_\Gamma A_i$ *for all* $i \in I$;

Dis $P \models_\Gamma \bigvee_{i\in I} A_i$, *if* $P \models_\Gamma A_i$ *for some* $i \in I$.

Notation 1. *To simplify, we write* $P \models A$ *instead of* $P \models_\emptyset A$.

We now provide a brief explanation of the relation defined above.

- $\mathbf{0}$ is satisfied by every process that can not perform any transition, for instance $\mathbf{0}$ and $(\nu a)a.\mathbf{0}$;
- $\bar{\mathbf{0}}$, the dual of $\mathbf{0}$, is satisfied by every process that can perform an action;
- a type variable $\mathcal{Z}$ is exactly satisfied by the process variable given by the environment Γ;
- $A \mid B$ is satisfied by every process that can be spatially decomposed in two parts, provided each part satisfies A and B, respectively;
- $\pi.A$ is satisfied by every process which has a π-derivative P' that satisfies A and such that every π-derivative of it is bisimilar to P';
- $a\circledR A$ is satisfied by every process which can be transformed (using structural congruence) into a process restricted by a and that the scope of the restriction satisfies A;
- $(\mathbf{rec}\mathcal{Z}.A)$ is satisfied by every process which can be transformed into a process recursion, the body of the recursion satisfies A and that the variable of the recursion satisfies $\mathcal{Z}$;
- $\langle\pi\rangle A$ is satisfied by every process having a π-derivative that satisfies A;
- $[\pi] A$ is satisfied by every process P such that either every π-derivative of P satisfies A or P does not have any π-derivative;

- $P \models_\Gamma \bigwedge_{i \in I} A_i$ is satisfied by every process that satisfies A_i for every $i \in I$;
- $P \models_\Gamma \bigvee_{i \in I} A_i$ is satisfied by every process that satisfies A_i for some $i \in I$;

The use of both structural congruence and labelled transition in the definition of satisfaction enables the expressiveness to talk about spatial properties and behavioural properties.

Let us now introduce some derived forms for the system. These will enable other connectives to be expressed, which subsequently will make its power more apparent. First, it is more convenient to subscribe a conjunction or a disjunction by a condition different from $i \in I$. Thus, it becomes easier and more intuitive to define the following logical operators - modal operators.

Definition 3.2.14. (Abbreviations)

$$\begin{array}{lcll} \langle t \rangle\, A & \stackrel{\text{abv}}{=} & \langle \pi_1 \dots \pi_n \rangle\, A & \stackrel{\text{abv}}{=} \quad \langle \pi_1 \rangle \dots \langle \pi_n \rangle\, A, \textit{ where } n \geq 1 \textit{ and } t = \pi_1 \dots \pi_n; \\ [t]\, A & \stackrel{\text{abv}}{=} & [\pi_1 \dots \pi_n]\, A & \stackrel{\text{abv}}{=} \quad [\pi_1] \dots [\pi_n]\, A, \textit{ where } n \geq 1 \textit{ and } t = \pi_1 \dots \pi_n; \\ \Diamond A & \stackrel{\text{abv}}{=} & \bigvee_{t \in Act^*} \langle t \rangle\, A; & \\ \Box A & \stackrel{\text{abv}}{=} & \bigwedge_{t \in Act^*} [t]\, A; & \\ !A & \stackrel{\text{abv}}{=} & (\textbf{\textit{rec}} Z.A \mid Z). & \end{array}$$

We also provide a brief explanation of these operators.

- $\langle t \rangle\, A$, read as *possible t-A*, is satisfied by every process which has a t-descendant that satisfies A;
- $[t]\, A$, read as *necessarily t-A*, is satisfied by every process such that every t-descendant satisfies A;
- $\Diamond A$, read as *some time* A, is satisfied by every process which has a t-descendant that satisfies A, for some $t \in Act*$;
- $\Box A$, read as *always* A, is satisfied by every process such that every t-descendant satisfies A, for every $t \in Act*$;
- $!A$, read as *iterate* A, is satisfied by every process recursion with $A \mid Z$ in the body of the recursion.

3.3 Deductive system

3.3.1 Structural type rules

We now introduce an important part of our type system: the structural type rules.

To cater for free variables (within the body of recursion, etc.) we need an *environment* Γ which is an injective function mapping process variables into type variables, as already stated. Hence, *judgements* are of the form $\Gamma \vdash P : A$, where P is a process, A is a type and Γ has the form $X_1 : Z_1, \dots, X_n : Z_n$. Moreover, $\Gamma, X : Z$ denotes the disjoint union of Γ and $\{X : Z\}$, where $Z \notin \text{codom}(\Gamma)$.

The inference rules and axioms composing the structural type system are those listed on table 3.1. It has exactly one rule for each connective of the process calculus; these rules build a correspondence between the calculus operators and the structural type operators. Rule **T-Voi** states that the process **0** has type **0**; rule **T-Var** uses the environment $\Gamma, X : Z$ to determine the type of the process variable X; in rules **T-Par**, **T-Pre** and **T-New** a type of a composite process can be inferred from its components with the same environment;

$$\frac{}{\Gamma \vdash \mathbf{0} : \mathbf{0}}\ \text{T-Voi} \qquad \frac{}{\Gamma, X : \mathcal{Z} \vdash X : \mathcal{Z}}\ \text{T-Var}$$

$$\frac{\Gamma \vdash P : A \quad \Gamma \vdash Q : B}{\Gamma \vdash P \mid Q : A \mid B}\ \text{T-Par}$$

$$\frac{\Gamma \vdash P : A}{\Gamma \vdash \pi.P : \pi.A}\ \text{T-Pre} \qquad \frac{\Gamma \vdash P : A}{\Gamma \vdash (\nu a)P : a\circledR A}\ \text{T-New}$$

$$\frac{\Gamma, X : \mathcal{Z} \vdash P : B}{\Gamma \vdash (\mathbf{rec} X.P) : (\mathbf{rec}\mathcal{Z}.B)}\ \text{T-Rec}$$

Table 3.1: The Structural Type System.

rule **T-Rec** states that recursive processes have recursive types which can be inferred from the type of the body of the recursion, extending the environment to cope with the variables that become free.

Definition 3.3.1. (Structural derivation) *The* Structural Type System *consists of all the structural rules. Therefore,* valid structural type judgements *are those which can be inferred from these rules and a* structural typing derivation *is a proof of the validity of a certain structural type judgement.*

Notation 2. *To simplify, we write* $\vdash P : A$ *instead of* $\emptyset \vdash P : A$.

3.3.2 Subtyping

The structural typing rules introduced are insufficient to infer interesting properties about processes. This section introduces a binary relation over types which is central to the type system. Amidst the type system, this relation brings inference power and expressiveness. Commonly known as subtyping, it is introduced into the the system through a *subsumption* rule Subsumbtion functions like a bridge between typing and subtyping relations.

The subsumption rule

Weak simulation relates processes according to their behaviour. As a type denotes a set of processes, it is natural to define a relation over types based on weak simulation.

Definition 3.3.2. (Subtyping) Subtyping, $<:$*, is the smallest binary relation over types such that, for every types* A *and* B, $A <: B$ *if, for every process* P *and environment* Γ *such that* $P \models_\Gamma A$*, there exists a process* Q *such that* $P <: Q$ *and* $Q \models_\Gamma B$.
We say that A *is a* subtype *of* B *if* $A <: B$.

A simpler intuition is to read $A <: B$ as logical entailment to a certain extent, that is, "if a process P has type A, then it is weakly simulated by a process Q with type B".

Like weak simulation, subtyping is also reflexive and transitive. In other words, subtyping has a preorder structure.

Proposition 3.3.3. (Preorder) *The subtyping relation is a preorder. That is:*

1. $A <: A$;
2. $A <: B$ *and* $B <: C \Rightarrow A <: B$.

Proof. For part 1., just note that the identity relation is a weak simulation. Therefore, for any P such that $P \models_\Gamma A$, P is weakly simulated by P itself; so the subtyping relation is reflexive.
For part 2., assume $P \models_\Gamma A$. Then, as $A <: B$, there exists Q such that $P <: Q$ and $Q \models_\Gamma B$. Subsequently, as $B <: C$, there exists R such that $Q <: R$ and $R \models_\Gamma C$. It is easy to see that if $P <: Q$ and $Q <: R$, then $P <: R$ and this suffices to conclude that the subtyping relation is transitive. □

Type equivalence is defined as the symmetrical closure of subtyping.

Definition 3.3.4. (Type Equivalence) *For every types* A *and* B, $A <:> B$, *if* $A <: B$ *and* $B <: A$.

A simple observation is that type equivalence is an equivalence relation.

Proposition 3.3.5. (Equivalence) $<:>$ *is an equivalence relation.*

Proof. Reflexivity and transitivity are inferred directly from proposition 3.3.3 and symmetry comes from definition 3.3.4. □

The following rule defines the bridge between typing and subtyping.

Definition 3.3.6. (Subsumption rule)

$$\frac{\Gamma \vdash P : A \quad A <: B}{\Gamma \vdash P : B}\text{, with } A \not<:> \mathbf{0} \text{ or } B \neq \bar{\mathbf{0}} \quad \textit{T-Sub}$$

This rule states that, if P has type A and A is a subtype of B, then P also has type B. For instance, it is easy to see that process $P = a.\mathbf{0}$ is weakly simulated by process $Q = a.\mathbf{0} \mid b.\mathbf{0}$. The only action that P can perform before becoming inactive is a and Q mimics this action, though not becoming inactive after that. It is straightforward to infer structurally that P has type $A = a.\mathbf{0}$ and to infer that $a.\mathbf{0}$ is subtype of $B = a.\mathbf{0} \mid b.\mathbf{0}$. Therefore, by subsumption we conclude that P has also type $a.\mathbf{0} \mid b.\mathbf{0}$. This means that type B expresses at least the properties of P.

The side-condition is included to prevent awkward results. For instance, without it, it would be possible to infer vacuously that process $\mathbf{0}$ has type $\bar{\mathbf{0}}$, *i.e.* that the inactive process has at least the property of doing something.

Subtyping rules

The subtype relation is formalized as a collection of inference rules for deriving statements of the form $A <: B$. In this subsection we introduce several subtyping rules adding to reflexivity and transitivity proved in the last subsection. The rules are presented grouped in propositions. Every inference rule is proved correct.

The following proposition says that the the set of types equipped with parallel composition constitutes a commutative monoid with type $\mathbf{0}$ as its neutral element.

Proposition 3.3.7. (Commutative monoid)

1. $A \mid \mathbf{0} <:> A$;
2. $A \mid B <:> B \mid A$;
3. $A \mid (B \mid C) <:> (A \mid B) \mid C$.

Proof.

Neutral element: First, assume that $P \models_\Gamma A \mid \mathbf{0}$. Then, there exists processes Q and R such that $P \equiv Q \mid R$, $Q \models_\Gamma A$ and $R \models_\Gamma \mathbf{0}$. Consider the binary relation $\mathcal{S} = \{(P \mid \mathbf{0}, P) : P \in \mathcal{P}\}$. Then $(Q \mid \mathbf{0}, Q) \in \mathcal{S}$ and, if $Q \mid \mathbf{0} \stackrel{e}{\Rightarrow} Q'$, then $Q' = Q'' \mid \mathbf{0}$, $Q \stackrel{e}{\Rightarrow} Q''$ and $(Q'' \mid \mathbf{0}, Q'') \in \mathcal{S}$. So $\mathcal{S}$ is a simulation and $Q \mid \mathbf{0}$ is weakly simulated by Q; hence, $A \mid \mathbf{0} <: A$.

Secondly, assume that $P \models_\Gamma A$. Of course $\mathbf{0} \models_\Gamma \mathbf{0}$ and, by definition 2.2.6, $P \equiv P \mid \mathbf{0}$. So $P \models_\Gamma A \mid \mathbf{0}$. Hence, as $P <: P$, we have that $A <: A \mid \mathbf{0}$.

Therefore $A \mid \mathbf{0} <:> A$.

Commutativity: First, assume that $P \models_\Gamma A \mid B$. Then, there exists processes Q and R such that $P \equiv Q \mid R$, $Q \models_\Gamma A$ and $R \models_\Gamma B$. By definition 2.2.6, $Q \mid R \equiv R \mid Q$. Moreover, $P \equiv R \mid Q$ and, subsequently, $P \models_\Gamma B \mid A$. Then, as $P <: P$, we have that $A \mid B <: B \mid A$.

The other direction is similar.

Therefore $A \mid B <:> B \mid A$.

Associativity: First, assume that $P \models_\Gamma A \mid (B \mid C)$. Then, there exists processes Q and R such that $P \equiv Q \mid R$, $Q \models_\Gamma A$ and $R \models_\Gamma (B \mid C)$. Again, there exists processes R_1 and R_2 such that $R \equiv R_1 \mid R_2$, $R_1 \models_\Gamma B$ and $R_2 \models_\Gamma C$. Moreover, there exists processes Q, R_1 and R_2 such that $P \equiv Q \mid (R_1 \mid R_2)$ and $Q \models_\Gamma A$, $R_1 \models_\Gamma B$ and $R_2 \models_\Gamma C$. By definition 2.2.6, $Q \mid (R_1 \mid R_2) \equiv (Q \mid R_1) \mid R_2$. Then, $P \equiv (Q \mid R_1) \mid R_2$ and, subsequently, $P \models_\Gamma (A \mid B) \mid C$. Then, as $P <: P$, we have that $A \mid (B \mid C) <: (A \mid B) \mid C$.

The other direction is similar.

Therefore $A \mid (B \mid C) <:> (A \mid B) \mid C$.

□

The following proposition is a rather important preservation result.

Proposition 3.3.8. (Preservation) *Subtyping is preserved by all type operators except conjunction.*

Proof.

1. Suppose $A <: B$ and let $P \models_\Gamma \pi.A$. Then, there exists P'such that $P \xrightarrow{\pi} P'$, $P' \models_\Gamma A$ and, for every π' and P'', $P \xrightarrow{\pi'} P''$ implies that $\pi' = \pi$ and $P'' \approx P'$. Subsequently, by hypothesis, there exists Q' such that $P' <: Q'$ and $Q' \models_\Gamma B$. Let Q be $\pi.Q'$. Then, $Q \models_\Gamma \pi.B$ and it remains to be shown that $P <: Q$. For this purpose assume that $P \xrightarrow{\pi'} P''$. Then, as already stated, $\pi' = \pi$ and $P'' \approx P'$, and, in particular, $P'' <: P'$. On the other hand, $Q \xrightarrow{\pi} Q'$ and $P' <: Q'$. Then, by transitivity, $P'' <: Q'$ and we conclude that there exists a simulation $\mathcal{S}$ such that $P\mathcal{S}Q$. Therefore, $P <: Q$.

2. Suppose $A <: B$ and let $P \models_\Gamma A \mid C$. Then, there exists P_1 and P_2 such that $P \equiv P_1 \mid P_2$, $P_1 \models_\Gamma A$ and $P_2 \models_\Gamma C$. Subsequently, by hypothesis, there exists Q_1 such that $P_1 <: Q_1$ and $Q_1 \models_\Gamma B$. Then, $Q_1 \mid P_2 \models_\Gamma B \mid C$. By corollary 3.2.12, $P <: P_1 \mid P_2$ and, by proposition 3.2.10, $P_1 \mid P_2 <: Q_1 \mid P_2$; thus, by transitivity, $P <: Q_1 \mid P_2$.

3. Suppose $A <: B$ and let $P \models_\Gamma a®A$. Then, there exists P_1 such that $P \equiv (\nu a)P_1$ and $P_1 \models_\Gamma A$. Subsequently, by hypothesis, there exists Q_1 such that $P_1 <: Q_1$ and $Q_1 \models_\Gamma B$. Then we have that $(\nu a)Q_1 \models_\Gamma a®B$. By corollary 3.2.12, $P <: (\nu a)P_1$ and, by proposition 3.2.10, we get $(\nu a)P_1 <: (\nu a)Q_1$; thus, by transitivity, $P <: (\nu a)Q_1$.

4. Suppose $A <: B$ and let $P \models_\Gamma (\mathbf{rec}Z.A)$. Then, there exists X and P_1 such that $P \equiv (\mathbf{rec}X.P_1)$ and $P_1 \models_{\Gamma,\{X:Z\}} A$. Subsequently, by hypothesis, there exists Q_1 such that $P_1 <: Q_1$ and $Q_1 \models_{\Gamma,\{X:Z\}} B$. Then, $(\mathbf{rec}X.Q_1) \models_\Gamma (\mathbf{rec}Z.B)$. By corollary 3.2.12, $P <: (\mathbf{rec}X.P_1)$ and, by proposition 3.2.10, $(\mathbf{rec}X.P_1) <: (\mathbf{rec}X.Q_1)$; thus, by transitivity, $P <: (\mathbf{rec}X.Q_1)$.

5. Suppose $A <: B$ and let $P \models_\Gamma [\pi] A$. Then, either $P \xrightarrow{\pi} \not\rightarrow$ for every π or $P \models_\Gamma \pi.A$. So, we must consider both cases. In the first, it is easy to find a process Q and a weak simulation $\mathcal{S}$ such that $P\mathcal{S}Q$ and $Q \models_\Gamma [\pi] B$. In fact, every process Q vacuously weakly simulates P because $P \xrightarrow{\pi} \not\rightarrow$ for every π. In the second case, using *Part (1.):*, one can conclude that there exists Q such that $P <: Q$ and $Q \models_\Gamma [\pi] B$. Subsequently, $Q \models_\Gamma \pi.B$.

   In both cases $\pi.A <: \pi.B$.

6. Suppose $A <: B$ and let $P \models_\Gamma \langle\pi\rangle A$. Then, there exists a process P' such that $P \xrightarrow{\pi} P'$ and $P' \models_\Gamma A$. By hypothesis, there exists Q' such that $P' <: Q'$ and $Q' \models_\Gamma B$. Let Q be the process obtained from P by substituting the occurrence of P' by Q'. First, it is true that $Q \xrightarrow{\pi} Q'$ because so did P. Then, $Q \models_\Gamma \langle\pi\rangle B$. Secondly, $P <: Q$ by construction and using the fact that $P' <: Q'$.

7. Suppose $A <: B$ and let $P \models_\Gamma A \vee C$. Then, $P \models_\Gamma A$ or $P \models_\Gamma C$. By hypothesis, there exists Q such that $P <: Q$ and $Q \models_\Gamma B$. Hence $Q \models_\Gamma B \vee C$. It is easy to generalize this case to the index set I; just consider C to be $\bigvee_{i\in I} C_i$

□

It is still an open result to determine whether conjunction preserves subtyping or not.

The proof of the next result is given on chapter 4.3.

Proposition 3.3.9. *If $A \equiv^t B$, then $A <: B$.*

The next, is an auxiliary result. It is used in the proof of the following proposition.

Lemma 3.3.10. *Let A be an arbitrary type and $s = \pi_1 \dots \pi_n$ an arbitrary sequence of actions. If $A <: [\pi] A$, for every π, and $P \models_\Gamma A$, then exactly one of the following assertions is true:*

1. *There exists a sequence of processes $P_1, \dots, P_n, P'_0, \dots, P'_n$ such that:*
   - *(a)* $P'_0 = P$;
   - *(b)* $P'_k <: P_{k+1}$, $0 \leq k \leq n$;
   - *(c)* $P_k \xrightarrow{\pi_k} P'_k$, $1 \leq k \leq n-1$;
   - *(d)* $P_k \models_\Gamma \pi_k.A$, $1 \leq k \leq n-1$;
   - *(e)* $P_n \models_\Gamma [\pi_n] A$.
2. *There exists a sequence of processes $P_1, \dots, P_j, P'_0, \dots, P'_j$, with $j \leq n-1$, such that:*
   - *(a)* $P'_0 = P$;
   - *(b)* $P'_k <: P_{k+1}$, $0 \leq k \leq j-1$;
   - *(c)* $P_k \xrightarrow{\pi_k} P'_k$, $1 \leq k \leq n-1$;
   - *(d)* $P_k \models_\Gamma \pi_k.A$, $1 \leq k \leq n-1$;
   - *(e)* $P_j \not\xrightarrow{\pi_j}$.

Proof. By induction on the size of the action sequence.

n=1 If $A <: [\pi] A$, $\forall \pi$, and $P \models_\Gamma A$, then, by definition of Subtyping (3.3.2), there exists a process P_1 such $P <: P_1$ and $P_1 \models_\Gamma [\pi] A$. Let $P = P_0$. Thus, conditions *(a)-(e)* from the first assertion are immediately satisfied.

n=m+1 By induction hypothesis the lemma is valid for $n = m$ and we have two cases to consider:

- Suppose that it is the first assertion that is true for m. Then, through *(e)*, there exists a process P_m such that $P_m \models_\Gamma [\pi_m] A$, i.e., either $P_m \models_\Gamma \pi_m.A$ or $P_m \not\xrightarrow{\pi_m}$. If the latter is true, then the second assertion of the lemma is true. If the former is true, then there exists a process P'_m such that $P_m \xrightarrow{\pi_m} P'_m$ and $P'_m \models_\Gamma A$. Hence, by hypothesis, there exists a process P_{m+1} such that $P_m <: P_{m+1}$ and $P_{m+1} \models_\Gamma [\pi_{m+1}] A$. Therefore conditions *(a)-(e)* are verified; that is, the first assertion is true for $m+1$;
- Suppose that it is the second assertion that is true. Then, as $j \leq m-1 \leq m = n-1$, the same sequence suffices to verify the second assertion for $m+1$.

□

The following proposition gathers several crucial subtyping rules needed for completeness and to derive the motivating example.

Proposition 3.3.11.

1. $A <: !A \mid A$;
2. $A \mid !A <:> !A$;
3. $\pi.A <: \langle\pi\rangle A$;
4. *If* $A <: \langle\pi\rangle B$, *then* $A \mid C <: \langle\pi\rangle (B \mid C)$;
5. *If* $A_1 <: \langle\alpha\rangle B_1$ *and* $A_2 <: \langle\bar{\alpha}\rangle B_2$, *then* $A_1 \mid A_2 <: \langle\tau\rangle (B_1 \mid B_2)$;
6. $a\text{®}(\langle\pi\rangle A) <: \langle\pi\rangle (a\text{®}A)$, *if* $a \neq \pi, \bar{\pi}$;
7. $\langle\pi\rangle A <: \pi.A$, *if* $\forall P\colon (P \models_\Gamma \langle\pi\rangle A \Longrightarrow \exists P'\colon (\forall P'', \pi'\colon P \xrightarrow{\pi'} P'' \Longrightarrow \pi' = \pi \wedge P'' \approx P'))$;
8. $\pi.A <: [\pi] A$;
9. $(\textbf{\textit{rec}}Z.A) <: A\{(\textbf{\textit{rec}}Z.A)/Z\}$;
10. *If* $\forall P\colon (P \models_\Gamma B \Longrightarrow P \xrightarrow{\pi}\!\!\!\!\!/\ )$, *then* $B <: [\pi] A$;
11. $A_j <: \bigvee_{i\in I} A_i$, *if* $j \in I$;
12. $a\text{®}(!a \mid (\textbf{\textit{rec}}Z.\bar{a}.(a \mid a \mid Z))) <: [\pi] (a\text{®}(!a \mid (\textbf{\textit{rec}}Z.\bar{a}.(a \mid a \mid Z))))$, *for every* π.
13. *If* $A <: [\pi] A$, *for every* π, *then* $A <: \Box A$.

Proof.

1. Let $P \models_\Gamma A$ and $Q = R \mid P$ where $R \models_\Gamma !A$. Then, $Q \models_\Gamma !A \mid A$. Consider the binary relation $\mathcal{S} = \{(P, R \mid P) : P, R \in \mathcal{P}\}$. It is clear that $(P, Q) \in \mathcal{S}$ and if $P \stackrel{e}{\Rightarrow} P'$ then $R \mid P \stackrel{e}{\Rightarrow} R \mid P'$, that is, there exists $Q' \in \mathcal{P}$ such that $Q \stackrel{e}{\Rightarrow} Q'$ and $P' \mathcal{S} Q'$. So Q weakly simulates P, therefore, $A <: !A \mid A$.

2. First, let $P \models_\Gamma A \mid !A$. Then, there exists P_1 and P_2 such that $P \equiv P_1 \mid P_2$, $P_1 \models_\Gamma A$ and $P_2 \models_\Gamma !A$. If $P_2 \models_\Gamma !A$, then, by definition, there exists X and P_{21} such that $P_2 \equiv (\textbf{rec}X.P_{21})$ and $P_{21} \models_{\Gamma,\{X:Z\}} A \mid Z$. Subsequently, there exists P_{22} and P_{23} such that $P_{21} \equiv P_{22} \mid P_{23}$, $P_{22} \models_{\Gamma,\{X:Z\}} A$ and $P_{23} \models_{\Gamma,\{X:Z\}} Z$. Then, by definition 3.2.13, $P_{23} = X$. Using repeatedly proposition 3.2.12, the fact that simulation is transitivity and preserved by all process operators to the structural relations above yields $P <: (\textbf{rec}X.(P_{22} \mid X)) \mid P_2$. Note that either $P_{22} <: P_1$ or $P_1 <: P_{22}$. Suppose without loss of generality that $P_1 <: P_{22}$ and let $Q = (\textbf{rec}X.(P_{22} \mid X))$. Then, $P <: (\textbf{rec}X.(P_{22} \mid X)) \mid P_{22}$ and it is easy to find a weak simulation $\mathcal{S}$ such that $(\textbf{rec}X.(P_{22} \mid X)) \mid P_{22} <: (\textbf{rec}X.(P_{22} \mid X))$. Hence $P <: (\textbf{rec}X.(P_{22} \mid X))$ and, thus, $A \mid !A <: !A$.

Secondly, let $P \models_\Gamma !A$. Then, there exists X and P' such that $P \equiv (\textbf{rec}X.P')$ and $P' \models_\Gamma A \mid X$. Subsequently, there exists P'_1 and P'_2 such that $P' \equiv P'_1 \mid P'_2$, $P'_1 \models_{\Gamma,\{X:Z\}} A$ and $P'_2 \models_{\Gamma,\{X:Z\}} X$. Let $Q = P'_1 \mid P$. Then, $Q \models_\Gamma A \mid !A$. Let $\mathcal{S} = \{(P, R \mid P) : P, R \in \mathcal{P}\}$. Then, $(P, Q) \in \mathcal{S}$ and if

$P \stackrel{\epsilon}{\Rightarrow} P'$ then $P_1' \mid P \stackrel{\epsilon}{\Rightarrow} P_1' \mid P'$, that is, there exists $Q' \in \mathcal{P}$ such that $Q \stackrel{\epsilon}{\Rightarrow} Q'$ and $P' \mathcal{S} Q'$. So Q weakly simulates P, i.e., $!A <: A \mid !A$.

Therefore $!A \mid A <:> !A$.

3. Let $P \models_\Gamma \pi.A$. Then, in particular, there exists P' such that $P \stackrel{\pi}{\rightarrow} P'$ and $P' \models_\Gamma A$. Hence, $P \models_\Gamma \langle \pi \rangle A$ and as, by proposition 3.3.3, $P <: P$ we have that $\pi.A <: \langle \pi \rangle A$.

4. Let $P \models_\Gamma A \mid C$. Then, there exists P_1 and P_2 such that $P \equiv P_1 \mid P_2$, $P_1 \models_\Gamma A$ and $P_2 \models_\Gamma C$. By hypothesis, there exists a process Q_1 such that $P_1 <: Q_1$ and $Q_1 \models_\Gamma \langle \pi \rangle B$. Then, by proposition 3.3.8, $P_1 \mid P_2 <: Q_1 \mid P_2$. By proposition 3.2.12, we have that $P \equiv P_1 \mid P_2$ implies that $P <: P_1 \mid P_2$ and, by proposition 3.3.3 (transitivity of $<:$), we have that $P <: Q_1 \mid P_2$. It remains to be shown that $Q_1 \mid P_2 \models_\Gamma \langle \pi \rangle (B \mid C)$. Thus, as $Q_1 \models_\Gamma \langle \pi \rangle B$, there exists Q_1' such that $Q_1 \stackrel{\pi}{\rightarrow} Q_1'$ and $Q_1' \models_\Gamma B$. Then, by definition 2.2.9, $Q_1 \mid P_2 \stackrel{\pi}{\rightarrow} Q_1' \mid P_2$ and by definition 3.2.13, $Q_1' \mid P_2 \models_\Gamma B \mid C$. Hence, $Q_1 \mid P_2 \models_\Gamma \langle \pi \rangle (B \mid C)$ and we conclude that $A \mid C <: \langle \pi \rangle (B \mid C)$.

5. Let $P \models_\Gamma A_1 \mid A_2$. Then, there exists P_1 and P_2 such that $P \equiv P_1 \mid P_2$, $P_1 \models_\Gamma A_1$ and $P_2 \models_\Gamma A_2$. By hypothesis, there exists processes Q_1 and Q_2 such that $P_1 <: Q_1$, $P_2 <: Q_2$, and $Q_1 \models_\Gamma \langle \alpha \rangle B_1$ and $Q_2 \models_\Gamma \langle \bar{\alpha} \rangle B_2$. At this point it is quite easy to conclude that $P_1 \mid P_2 <: Q_1 \mid Q_2$ so it is omitted. By proposition 3.2.12, we have that $P \equiv P_1 \mid P_2$ implies that $P <: P_1 \mid P_2$ and, by proposition 3.3.3 (transitivity of $<:$), we have that $P <: Q_1 \mid Q_2$. It remains to be shown that $Q_1 \mid Q_2 \models_\Gamma \langle \tau \rangle (B_1 \mid B_2)$. Thus, as $Q_1 \models_\Gamma \langle \alpha \rangle B_1$ and $Q_2 \models_\Gamma \langle \bar{\alpha} \rangle B_2$, there exists Q_1' and Q_2' such that $Q_1 \stackrel{\alpha}{\rightarrow} Q_1'$ and $Q_1' \models_\Gamma B_1$, and, $Q_2 \stackrel{\bar{\alpha}}{\rightarrow} Q_2'$ and $Q_2' \models_\Gamma B_2$, respectively. Then, by definition 2.2.9, $Q_1 \mid Q_2 \stackrel{\tau}{\rightarrow} Q_1' \mid Q_2'$ and by definition 3.2.13, $Q_1' \mid Q_2' \models_\Gamma B_1 \mid B_2$. Hence, $Q_1 \mid Q_2 \models_\Gamma \langle \tau \rangle (B_1 \mid B_2)$ and we conclude that $A_1 \mid A_2 <: \langle \tau \rangle (B_1 \mid B_2)$.

6. Let $P \models_\Gamma a®(\langle \pi \rangle A)$. Then, there exists P_1 such that $P \equiv (\nu a) P_1$ and $P_1 \models_\Gamma \langle \pi \rangle A$. Subsequently, there exists P_2 such that $P_1 \stackrel{\pi}{\rightarrow} P_2$ and $P_2 \models_\Gamma A$. Then, by definition 2.2.9 and as $a \neq \pi, \bar{\pi}$, $(\nu a) P_1 \stackrel{\pi}{\rightarrow} (\nu a) P_2$ and by definition 3.2.13, $(\nu a) P_2 \models_\Gamma a®A$. Hence, $(\nu a) P_1 \models_\Gamma \langle \pi \rangle (a®A)$. By proposition 3.2.12, $P \equiv (\nu a) P_1$ implies that $P <: (\nu a) P_1$ so we conclude that $a®(\langle \pi \rangle A) <: \langle \pi \rangle (a®A)$.

7. Let $P \models_\Gamma \langle \pi \rangle A$. Then, by the side-condition, there exists P' such that for every P'' and π' such that $P \stackrel{\pi'}{\rightarrow} P''$, then $\pi' = \pi$ and $P'' \approx P'$. Hence, by definition 3.2.13, $P \models_\Gamma \pi.A$.

8. Immediate by definition 3.2.13.

9. Let $P \models_\Gamma (\mathbf{rec} Z.A)$. Then there exists a process variable X and a process P such that $P \equiv (\mathbf{rec} X.P')$ and $P' \models_\Gamma A$. Then $P'\{(\mathbf{rec} X.P')/X\} \models_\Gamma A\{(\mathbf{rec} Z.A)/Z\}$ and, by proposition 3.2.12, $P <: (\mathbf{rec} X.P')$. Now, let $\mathcal{S} = \{((\mathbf{rec} X.Q), Q\{(\mathbf{rec} X.Q)/X\}) : Q \in \mathcal{P}\}$. Of course, $((\mathbf{rec} X.P'), P'\{(\mathbf{rec} X.P')/X\}) \in \mathcal{S}$, and it is almost direct to show that $\mathcal{S}$ is a simulation. For suppose that $(\mathbf{rec} X.P') \stackrel{\pi}{\rightarrow} P''$, then by definition 2.2.9, also $P'\{(\mathbf{rec} X.P')/X\} \stackrel{\pi}{\rightarrow} P''$; hence $(\mathbf{rec} X.P') <: P'\{(\mathbf{rec} X.P')/X\}$. Subsequently, $(\mathbf{rec} Z.A) <: A\{(\mathbf{rec} Z.A)/Z\}$.

10. Let $P \models_\Gamma B$. Then, by the side-condition, $P \stackrel{\tau}{\not\rightarrow}$. Hence, by definition 3.2.13, $P \models_\Gamma [\pi] A$.

11. Immediate by definition 3.2.13.

12. Let π be any action and $P \models_\Gamma a®(!a \mid (\mathbf{rec} Z.\bar{a}.(a \mid a \mid Z)))$. The first observation is that the only action that P can perform is τ. Therefore it is easy to prove that $a®(!a \mid (\mathbf{rec} Z.\bar{a}.(a \mid a \mid Z))) <: [\pi]\,(a®(!a \mid (\mathbf{rec} Z.\bar{a}.(a \mid a \mid Z))))$, for every π different from τ, using the definition 3.2.13 almost directly.

So, let π be τ and assume that $Q = \tau.P$. Then, $Q \models_\Gamma \tau.(a®(!a \mid (\mathbf{rec} Z.\bar{a}.(a \mid a \mid Z))))$ and, by definition 3.2.13, $Q \models_\Gamma [\tau]\,(a®(!a \mid (\mathbf{rec} Z.\bar{a}.(a \mid a \mid Z))))$. It remains to be shown that $P <: Q$ which is actually very easy; just notice that whatever action, π, that P performs is matched by an experiment $e = \tau\pi$ of Q.

13. Let $P \models_\Gamma A$ and let $s = \pi_1 \ldots \pi_n$ be any experiment. At this point, lemma 3.3.10 can be used to build a process Q such that $P <: Q$ and $Q \models_\Gamma [s]\, A$ in the following way.

First, assume that the first assertion of lemma 3.3.10 is true. Then there exists a sequence of processes $P_1, \ldots, P_n, P'_0, \ldots, P'_n$ such that:

1. $P'_0 = P$;
2. $P'_k <: P_k$, $0 \leq k \leq n$;
3. $P_k \xrightarrow{\pi_k} P'_k$, $1 \leq k \leq n-1$;
4. $P_k \models_\Gamma \pi_k.A$, $1 \leq k \leq n-1$;
5. $P_n \models_\Gamma [\pi_n]\, A$.

From *(3)* and *(4)* one can easily conclude from definition 3.2.13 that:

6. $P_k <: \pi_k.P'_k$, $1 \leq k \leq n-1$,

and, subsequently, by proposition 3.3.8, that:

7. $\pi_1 \ldots \pi_{k-1}.P_k <: \pi_1 \ldots \pi_{k-1}.\pi_k.P'_k$, $1 \leq k \leq n-1$.

On the other hand, from 2., again using repeatedly proposition 3.3.8, one can also conclude that:

8. $\pi_1 \ldots \pi_k.P'_k <: \pi_1 \ldots \pi_k.P_{k+1}$, $0 \leq k \leq n-1$.

Using transitivity of simulation several times over 7. and 6. we have that $P <: \pi_1 \ldots \pi_{n-1}.P_n$. So, let $Q = \pi_1 \ldots \pi_{n-1}.P_n$. Then $P <: Q$. Moreover, from 5., one can conclude that, for each $1 \leq i \leq n-1$:

9. $\pi_{n-i} \ldots \pi_{n-1}.P_n \models_\Gamma \pi_{n-i}.\,[\pi_{n-i+1}] \ldots [\pi_n]\, A$ and, consequently, by definition 3.2.13
10. $\pi_{n-i} \ldots \pi_{n-1}.P_n \models_\Gamma [\pi_{n-i}]\,.\,[\pi_{n-i+1}] \ldots [\pi_n]\, A$.

So, $\pi_1 \ldots \pi_{n-1}.P_n \models_\Gamma [\pi_1] \ldots [\pi_n]\, A$; that is, $Q \models_\Gamma [\pi_1] \ldots [\pi_n]\, A$.

Secondly, assume that the second assertion of lemma 3.3.10 is true. Then there exists a sequence of processes $P_1, \ldots, P_j, P'_0, \ldots, P'_j$ such that:

11. $P'_0 = P$;
12. $P'_k <: P_k, 0 \leq k \leq j-1$;
13. $P_k \xrightarrow{\pi_k} P'_k, 1 \leq k \leq n-1$;
14. $P_k \models_\Gamma \pi_k.A, 1 \leq k \leq n-1$;
15. $P_j \not\xrightarrow{\pi_j}$.

Thus, in a similar way as above one can conclude that $P <: \pi_1 \ldots \pi_{j-1}.P_j$. So, let $Q = \pi_1 \ldots \pi_{j-1}.P_j$. Then $P <: Q$. Moreover, from 15., one can conclude that, $P_j \models_\Gamma [\pi_j] B$ for every type B. In particular, $P_j \models_\Gamma [\pi_j].([\pi_{j+1}] \ldots [\pi_n] A)$. Using the same reasoning as above, from 14., we have that $\pi_1 \ldots \pi_{j-1}.P_j \models_\Gamma [\pi_1] \ldots [\pi_{j-1}] . [\pi_j] .([\pi_{j+1}] \ldots [\pi_n] A)$; that is, $Q \models_\Gamma [\pi_1] \ldots [\pi_n] A$

In any case $P <: Q$ and $Q \models_\Gamma [s] A$ and, as s is an arbitrary action sequence, we have that $Q \models_\Gamma [s] A$ for every action sequence s. Hence, $Q \models_\Gamma \bigwedge_{s \in Act^*} ([s] A)$; that is $Q \models_\Gamma \Box A$.

□

Finally, we present the last three subtyping rules considered in this thesis. These rules express properties of the active type $\bar{\mathbf{0}}$.

Proposition 3.3.12. ($\bar{\mathbf{0}}$ rules)

1. $A <: \bar{\mathbf{0}}$;
2. $a®(\bar{\mathbf{0}} \mid \bar{\mathbf{0}}) <: \bar{\mathbf{0}} \mid \bar{\mathbf{0}}$;
3. *If* $A <: \bar{\mathbf{0}}$ *and* $B <: \bar{\mathbf{0}}$, *then* $A \mid B <: \bar{\mathbf{0}} \mid \bar{\mathbf{0}}$;

Proof.

1. Let A be any type. Suppose that $P \models_\Gamma A$. There are two cases to consider: whether $P \not\xrightarrow{\pi}$ for every π or there is an action π and a process P' such that $P \xrightarrow{\pi} P'$. In the first case it is very easy to find a simulation $\mathcal{S}$ and a process Q such that $P \mathcal{S} Q$ and $Q \models_\Gamma \bar{\mathbf{0}}$. For instance, take $\mathcal{S} = \{(P, a.P) : P \not\xrightarrow{\pi} \text{ for every } \pi\}$. It vacuously satisfies the definition of a simulation and $a.P \models_\Gamma \bar{\mathbf{0}}$. In the second case note that if $P \xrightarrow{\pi} P'$ for some action π and process P', then $P \models_\Gamma \bar{\mathbf{0}}$ and $P <: P$. In any case $A <: \bar{\mathbf{0}}$.

2. Suppose $P \models_\Gamma a®(\bar{\mathbf{0}} \mid \bar{\mathbf{0}})$. Then, there exists P' such that $P \equiv (\nu a)P'$ and $P' \models_\Gamma \bar{\mathbf{0}} \mid \bar{\mathbf{0}}$. Subsequently, there exists P_1 and P_2 such that $P' \equiv P_1 \mid P_2$, $P_1 \models_\Gamma \bar{\mathbf{0}}$ and $P_2 \models_\Gamma \bar{\mathbf{0}}$. Let $Q = P_1 \mid P_2$. Hence $Q \models_\Gamma \bar{\mathbf{0}} \mid \bar{\mathbf{0}}$. It remains to be shown that $P <: Q$. Thus, as $P \equiv (\nu a)P'$ and $P' \equiv P_1 \mid P_2$, we have that $P \equiv (\nu a)(P_1 \mid P_2)$. Therefore $P <: (\nu a)(P_1 \mid P_2)$, by corollary 3.2.12. Consider $\mathcal{S}$ to be $\{((\nu a)P, P) : P \in \mathcal{P}\}$. It is easy to see that $\mathcal{S}$ is a simulation because if $(\nu a)P \xrightarrow{\pi} Q'$ then $Q' = (\nu a)Q''$ and also $P \xrightarrow{\pi} Q''$ and $(\nu a)Q'' \mathcal{S} Q''$; hence, $(\nu a)(P_1 \mid P_2) <: P_1 \mid P_2$. Lastly, by transitivity $P \equiv (P_1 \mid P_2)$.

3. Suppose $P \models_\Gamma A \mid B$. Then, there exists P_1 and P_2 such that $P \equiv P_1 \mid P_2$, $P_1 \models_\Gamma A$ and $P_2 \models_\Gamma B$. By *Part (1.)* there exists Q_1 and Q_2 such that $P_1 <: Q_1$ and $P_2 <: Q_2$, respectively. Let $Q = Q_1 \mid Q_2$. Then, $Q \models_\Gamma \bar{\mathbf{0}} \mid \bar{\mathbf{0}}$. Also, by corollary 3.2.12, $P <: P_1 \mid P_2$.

In addition, consider $\mathcal{S}$ to be $\{(P_1 \mid P_2, Q_1 \mid Q_2) : P_1 <: Q_1 \text{ and } P_2 <: Q_2\}$ and let $P_1 \mid P_2 \xrightarrow{\pi} P'$. Accordingly to the definition 2.2.9 (the LTS), whether P' is $P_1' \mid P_2$ or $P_1 \mid P_2'$ or $P_1' \mid P_2'$. The first two cases are identical, so let P' be $P_1' \mid P_2$. Then, $P_1 \xrightarrow{\pi} P_1'$ and, as $P_1 <: Q_1$, there exists Q_1' such that $Q_1 \xrightarrow{\pi} Q_1'$ and $P_1' <: Q_1'$. Therefore, $Q_1 \mid Q_2 \xrightarrow{\pi} Q_1' \mid Q_2$ and $(P_1' \mid P_2, Q_1' \mid Q_2) \in \mathcal{S}$. For the third case, let P' be $P_1' \mid P_2'$. Then, $\pi = \tau$, $P_1 \xrightarrow{\alpha} P_1'$ and $P_2 \xrightarrow{\bar{\alpha}} P_2'$. Thus, as $P_1 <: Q_1$ and $P_2 <: Q_2$, there exists Q_1' and Q_2' such that $Q_1 \xrightarrow{\alpha} Q_1'$ and $P_1' <: Q_1'$, and, $Q_2 \xrightarrow{\bar{\alpha}} Q_2'$ and $P_2' <: Q_2'$, respectively. Consequently, $Q_1 \mid Q_2 \xrightarrow{\tau} Q_1' \mid Q_2'$ and $(P_1' \mid P_2', Q_1' \mid Q_2') \in \mathcal{S}$. In conclusion, $P_1 \mid P_2 <: Q_1 \mid Q_2$.

Finally, by transitivity, $P <: Q$ and $Q \models_\Gamma \bar{\mathbf{0}} \mid \bar{\mathbf{0}}$.

□

We end this subsection presenting a few formalities.

Definition 3.3.13. (Subtyping derivation) *The* Subtyping System *consists of all the subtyping rules, that is, all rules given by propositions 3.3.3 - 3.3.12. Therefore,* valid subtyping judgements *are those which can be inferred from these rules and a* subtyping derivation *is a proof of the validity of a certain subtyping judgement.*

The inference rules and lemmas composing the subtyping system are listed in Table 3.2.

3.3.3 Typing rules

The purpose of this last section is to formalize the type system, namely to summarize the rules introduced in the previous sections.

Definition 3.3.16. (Conjunction rule)

$$\frac{\Gamma \vdash P : A_i \text{ for all } i \in I}{\Gamma \vdash P : \bigwedge_{i \in I} A_i} \textit{T-Con}$$

Contrary to disjunction, conjunction can not be inferred through subtyping. Therefore it must be introduced in the system by a separate rule. In addition, as we have not considered conjunction to be a structural connective, for coherence reasons, we present it separated from the structural rules.

This rule will be discussed further ahead. For the moment, it suffices to observe that no condition is imposed over the indexing set I.

Definition 3.3.17. (Typing derivation) *The* Type System *consists of all the structural rules, the subsumption rule and the conjunction rule. Therefore,* valid type judgements *are those which can be inferred from these rules and a* typing derivation *is a proof of the validity of a certain type judgement.*

The inference rules composing the type system are the structural rules listed on table 3.1 and the subsumption and the conjunction rules, that is, definitions 3.3.6 and 3.3.16 respectively.

$$A <: A$$

$$\frac{A <: B \quad B <: C}{A <: C}$$

$$\frac{A <: B}{\pi.A <: \pi.B}$$

$$\frac{A <: B}{A \mid C <: B \mid C}$$

$$\frac{A <: B}{a\circledR A <: a\circledR B}$$

$$\frac{A <: B}{(\mathbf{rec}\mathcal{Z}.A) <: (\mathbf{rec}\mathcal{Z}.B)}$$

$$\frac{A <: B}{[\pi]\, A <: [\pi]\, B}$$

$$\frac{A <: B}{\langle\pi\rangle\, A <: \langle\pi\rangle\, B}$$

$$\frac{A_j <: B_j}{\bigvee_{i\in I} A_i <: \bigvee_{i\in I} B_i}, \text{ if } j \in I$$

$$a\circledR A <:> b\circledR(A\{b/a\}) \ , b \text{ fresh in } A$$

$$(\mathbf{rec}\mathcal{Z}.A) <:> (\mathbf{rec}\mathcal{W}.A\{\mathcal{W}/\mathcal{Z}\}) \ , \mathcal{W} \text{ fresh in } A$$

$$A \mid \mathbf{0} <:> A$$

$$A \mid B <:> B \mid A$$

$$(A \mid B) \mid C <:> A \mid (B \mid C)$$

$$a\circledR\mathbf{0} <:> \mathbf{0}$$

$$b\circledR(a\circledR A) <:> a\circledR(b\circledR A)$$

$$a\circledR(A \mid B) <:> (a\circledR A \mid B) \ , a \notin \text{fn}(B)$$

$$A <: !A \mid A$$

$$A \mid !A <:> !A$$

$$\pi.A <: \langle\pi\rangle\, A$$

$$\pi.A <: [\pi]\, A$$

$$a\circledR(\langle\pi\rangle\, A) <: \langle\pi\rangle\, (a\circledR A) \ , \text{if } a \neq \pi, \bar{\pi}$$

$$(\mathbf{rec}\mathcal{Z}.A) <: A\{(\mathbf{rec}\mathcal{Z}.A)/\mathcal{Z}\}$$

$$A_j <: \bigvee_{i\in I} A_i \ , \text{if } j \in I$$

$$\frac{A <: [\pi]\, A}{A <: \Box A}, \text{ for every } \pi \in Act$$

$$\frac{A <: \langle\pi\rangle\, B}{A \mid C <: \langle\pi\rangle\, (B \mid C)}$$

$$\frac{A_1 <: \langle\alpha\rangle\, B_1 \quad A_2 <: \langle\bar{\alpha}\rangle\, B_2}{A_1 \mid A_2 <: \langle\tau\rangle\, (B_1 \mid B_2)}$$

$$a\circledR(!a \mid (\mathbf{rec}\mathcal{Z}.\bar{a}.(a \mid a \mid \mathcal{Z}))) <: [\pi]\, (a\circledR(!a \mid (\mathbf{rec}\mathcal{Z}.\bar{a}.(a \mid a \mid \mathcal{Z}))))$$
, for every $\pi \in Act$

$$A <: \bar{\mathbf{0}}$$

$$a\circledR(\bar{\mathbf{0}} \mid \bar{\mathbf{0}}) <: \bar{\mathbf{0}} \mid \bar{\mathbf{0}}$$

$$\frac{A <: \bar{\mathbf{0}} \quad B <: \bar{\mathbf{0}}}{A \mid B <: \bar{\mathbf{0}} \mid \bar{\mathbf{0}}}$$

Lemma 3.3.14. *For every process P such that $P \models_\Gamma B$ implies $P \stackrel{\pi}{\not\longrightarrow}$, then $B <: [\pi]\, A$.*

Lemma 3.3.15. *For every process P such that $P \models_\Gamma \langle\pi\rangle\, A$ implies that, there exists a process P' such that if $P \stackrel{\pi'}{\longrightarrow} P''$, then π' is π and $P'' \approx P'$, for every process P'' and action π', then $\langle\pi\rangle\, A <: \pi.A$.*

Table 3.2: The Subtyping System.

Chapter 4

Properties of the Type System

This chapter presents some properties of the type system. Section 4.1 presents an important type existence result and a series of results which are used in the proofs of the main results. In section 4.2, the main section, we prove a weak consistency result and a completeness result. Section 4.3 exhibits a proof of proposition 3.3.9 using completeness.

4.1 Preliminaries

First, we prove that, for every typing derivation, there exists another derived using only structural rules.

Lemma 4.1.1. *If $\Gamma \vdash P : A$, then there is B such that $B <: A$ and $\Gamma \vdash P : B$ is derived using a structural type rule or the conjunction rule in the final step of the derivation.*

Proof. First, note that for any derivation of $\Gamma \vdash P : A$ in the system, the last rule used must be one of the following: a Structural rule, the Conjunction rule or the Subsumption rule.

In the first two cases we can take B to be A, as $A <: A$.

In the last case, assume that $\Gamma \vdash P : A$ is derived using the subsumption rule in the last step of the derivation. As a typing derivation is finite assume that the last n consecutive steps of the derivation were inferred by the subsumption rule and the $n+1$ step was inferred by a structural type rule or the conjunction rule. Then, it must exist a sequence of types $B_1, \ldots, B_{n+1}$, such that $B_{n+1} = A$, $B_1 = B$ and, for each $1 \leq i \leq n$, $B_i <: B_{i+1}$, $\Gamma \vdash P : B_{i+1}$ is obtained from $\Gamma \vdash P : B_i$ by the subsumption rule and $\Gamma \vdash P : B_1$ is derived using a structural type rule or the conjunction rule in the final step of its derivation. By transitivity of the subtyping relation we conclude that $B_1 <: B_{n+1}$; that is $B <: A$. □

Lemma 4.1.2. (Weakening) *If X is fresh in P, Z is fresh in A, $X : Z \notin \Gamma$ and $\Gamma \vdash P : A$, then $\Gamma, X : Z \vdash P : A$.*

Proof. If $\Gamma \vdash P : A$, then, by Lemma 4.1.1, there exists a type B such that $B <: A$ and $\Gamma \vdash P : B$ is derived using a structural rule or the conjunction rule in the last step of the derivation.

First, we prove that $\Gamma, X : \mathcal{Z} \vdash P : B$, by induction on a derivation of $\Gamma \vdash P : B$ in the Type System. For a given derivation, we proceed by a case analysis on the final structural type rule or the conjunction rule used.

***Case* T-Voi:** $P = \mathbf{0}$
$B = \mathbf{0}$

Immediate from rule T-Voi and as $X : \mathcal{Z} \notin \Gamma$.

***Case* T-Var:** $P = X_1$
$B = \mathcal{Z}_1$

As X is fresh in X_1 and $\mathcal{Z}$ is fresh in $\mathcal{Z}_1$, we must have that $X_1 \neq X$ and $\mathcal{Z}_1 \neq \mathcal{Z}$. We also have that $X : \mathcal{Z} \notin \Gamma$, so, from rule T-Var, $\Gamma, X_1 : \mathcal{Z}_1, X : \mathcal{Z} \vdash X_1 : \mathcal{Z}_1$.

***Case* T-Pref:** $P = \pi.P'$
$B = \pi.A'$

The rule T-Pref has the following assumption,

1. $\Gamma \vdash P' : A'$.

By definition, $\text{fv}(\pi.P') = \text{fv}(P')$, $\text{bv}(\pi.P') = \text{bv}(P')$, $\text{fv}(\pi.A') = \text{fv}(A')$ and $\text{bv}(\pi.A') = \text{bv}(A')$; hence, from the hypothesis, X and $\mathcal{Z}$ are fresh in P' and A', respectively, and, subsequently, by induction hypothesis, $\Gamma, X : \mathcal{Z} \vdash P' : A'$. Then, $\Gamma, X : \mathcal{Z} \vdash \pi.P' : \pi.A'$, by T-Pref.

***Case* T-Par:** $P = P_1 \mid P_2$
$B = A_1 \mid A_2$

The rule T-Par has the following two assumptions,

1. $\Gamma \vdash P_1 : A_1$,
2. $\Gamma \vdash P_2 : A_2$.

By definition, $\text{fv}(P_1 \mid P_2) = \text{fv}(P_1) \cup \text{fv}(P_2)$, $\text{bv}(P_1 \mid P_2) = \text{bv}(P_1) \cup \text{bv}(P_2)$, $\text{fv}(A_1 \mid A_2) = \text{fv}(A_1) \cup \text{fv}(A_2)$ and $\text{bv}(A_1 \mid A_2) = \text{bv}(A_1) \cup \text{bv}(A_2)$; hence, from the hypothesis, X and $\mathcal{Z}$ are fresh in P_1, A_1, and in P_2 and A_2, respectively. So, by induction hypothesis,

3. $\Gamma, X : \mathcal{Z} \vdash P_1 : A_1$,
4. $\Gamma, X : \mathcal{Z} \vdash P_2 : A_2$,

respectively. Finally, by T-Par, $\Gamma, X : \mathcal{Z} \vdash P_1 \mid P_2 : A_1 \mid A_2$.

***Case* T-New:** $P = (\nu a)P'$
$B = a\circledR A'$

The rule T-New has the following assumption,

1. $\Gamma \vdash P' : A'$.

By definition, $\text{fv}((\nu a)P') = \text{fv}(P')$, $\text{bv}((\nu a)P') = \text{bv}(P')$, $\text{fv}(a\circledR A') = \text{fv}(A')$ and $\text{bv}(a\circledR A') = \text{bv}(A')$; hence, from the hypothesis, X and $\mathcal{Z}$ are fresh in P' and A', respectively. So, by induction

hypothesis, $\Gamma, X : \mathcal{Z} \vdash P' : A'$ and, by T-New, $\Gamma, X : \mathcal{Z} \vdash (\nu a)P' : a \circledR A'$.

***Case* T-Rec:** $P = (\mathbf{rec} X_1.P_1)$
$B = (\mathbf{rec} \mathcal{Z}_1.A_1)$

The rule T-Rec has the following assumption,

1. $\Gamma, X_1 : \mathcal{Z}_1 \vdash P' : A'$.

By hypothesis, X and $\mathcal{Z}$ are fresh in $(\mathbf{rec} X_1.P_1)$ and $(\mathbf{rec} \mathcal{Z}_1.A_1)$, respectively. So, X and $\mathcal{Z}$ must be different from X_1 and $\mathcal{Z}_1$, respectively; hence, X and $\mathcal{Z}$ are also fresh in P_1 and A_1, respectively. Then, by induction hypothesis, $\Gamma, X_1 : \mathcal{Z}_1, X : \mathcal{Z} \vdash P' : A'$ and, by T-Rec, $\Gamma, X : \mathcal{Z} \vdash (\mathbf{rec} X_1.P_1) : (\mathbf{rec} \mathcal{Z}_1.A_1)$.

***Case* T-Con:** $P = P$
$B = \bigwedge_{i \in I} B_i$

The rule T-Con has the following assumption,

1. $\Gamma \vdash P : B_i$ for all $i \in I$.

By hypothesis, $\mathcal{Z}$ is fresh in $\bigwedge_{i \in I} B_i$. So, $\mathcal{Z}$ is fresh in B_i for every $i \in I$, by definition. Then, by induction hypothesis, $\Gamma, X : \mathcal{Z} \vdash P : B_i$ for every $i \in I$. By T-Con, $\Gamma, X : \mathcal{Z} \vdash P : \bigwedge_{i \in I} B_i$.

So we conclude that $\Gamma, X : \mathcal{Z} \vdash P : B$. Then, by the subsumption rule and since $B <: A$, $\Gamma, X : \mathcal{Z} \vdash P : A$. □

Lemma 4.1.3. (Strengthening) *If $X \notin fv(P)$, $\mathcal{Z} \notin fv(A)$ and $\Gamma, X : \mathcal{Z} \vdash P : A$, then $\Gamma \vdash P : A$.*

Proof. If $\Gamma \vdash P : A$, then, by Lemma 4.1.1, there exists a type B such that $B <: A$ and $\Gamma \vdash P : B$ is derived using a structural rule or the conjunction rule in the last step of the derivation.

First, we prove that $\Gamma \vdash P : B$, by induction on a derivation of $\Gamma, X : \mathcal{Z} \vdash P : B$ in the Type System. For a given derivation, we proceed by a case analysis on the final structural type rule or the conjunction rule used.

***Case* T-Voi:** $P = \mathbf{0}$
$B = \mathbf{0}$

Immediate from rule T-Voi.

***Case* T-Var:** $P = X_1$
$B = \mathcal{Z}_1$

As $X \notin \text{fv}(X_1) = \{X_1\}$ and $\mathcal{Z} \notin \text{fv}(\mathcal{Z}_1) = \{\mathcal{Z}_1\}$, we must have that $X_1 \neq X$ and $\mathcal{Z}_1 \neq \mathcal{Z}$. Hence, exists $X_1 : \mathcal{Z}_1$ in Γ and, consequently, it is possible to derive $\Gamma \vdash X_1 : \mathcal{Z}_1$ in the system.

***Case* T-Pref:** $P = \pi.P'$
$B = \pi.A'$

The rule T-Pref has the following assumption,

1. $\Gamma, X : \mathcal{Z} \vdash P' : A'$.

By definition, $\text{fv}(\pi.P') = \text{fv}(P')$ and $\text{fv}(\pi.A') = \text{fv}(A')$; hence, from the hypothesis, X and $\mathcal{Z}$ are not free in P' and A', respectively. Subsequently, by induction hypothesis, $\Gamma \vdash P' : A'$. Then, $\Gamma \vdash \pi.P' : \pi.A'$, by T-Pref.

***Case* T-Par:** $P = P_1 \mid P_2$
$B = A_1 \mid A_2$

The rule T-Par has the following two assumptions,

1. $\Gamma, X : \mathcal{Z} \vdash P_1 : A_1$,
2. $\Gamma, X : \mathcal{Z} \vdash P_2 : A_2$.

By definition, $\text{fv}(P_1 \mid P_2) = \text{fv}(P_1) \cup \text{fv}(P_2)$ and $\text{fv}(A_1 \mid A_2) = \text{fv}(A_1) \cup \text{fv}(A_2)$; hence, from the hypothesis, X and $\mathcal{Z}$ are not free in P_1, P_2, and in A_1 and A_2, respectively. So, by induction hypothesis,

3. $\Gamma \vdash P_1 : A_1$,
4. $\Gamma \vdash P_2 : A_2$,

respectively. Finally, by T-Par, $\Gamma \vdash P_1 \mid P_2 : A_1 \mid A_2$.

***Case* T-New:** $P = (\nu a)P'$
$B = a\circledR A'$

The rule T-New has the following assumption,

1. $\Gamma, X : \mathcal{Z} \vdash P' : A'$.

By definition, $\text{fv}((\nu a)P') = \text{fv}(P')$ and $\text{fv}(a\circledR A') = \text{fv}(A')$; hence, from the hypothesis, X and $\mathcal{Z}$ are fresh in P' and A', respectively. So, by induction hypothesis, $\Gamma \vdash P' : A'$ and, by T-New, $\Gamma \vdash (\nu a)P' : a\circledR A'$.

***Case* T-Rec:** $P = (\mathbf{rec} X_1.P_1)$
$B = (\mathbf{rec} \mathcal{Z}_1.A_1)$

The rule T-Rec has the following assumption,

1. $\Gamma, X : \mathcal{Z}, X_1 : \mathcal{Z}_1 \vdash P' : A'$.

There are four cases to consider resulting from whether X is X_1 or not and $\mathcal{Z}$ is $\mathcal{Z}_1$ or not.

First, note that the cases where X is X_1 or $\mathcal{Z}$ is $\mathcal{Z}_1$ can not happen. This results from the fact that if it could happen, then, by the assumption of T-Rec and by hypothesis, the system could derive, for instance, $\Gamma, X : \mathcal{Z}, X : \mathcal{Z} \vdash P_1 : A_1$ which violates the definition of the environment Γ.

So, X and $\mathcal{Z}$ are different from X_1 and $\mathcal{Z}_1$, respectively. From the hypothesis and because $\text{fv}((\mathbf{rec} X_1.P_1)) = \text{fv}(P_1) \setminus \{X_1\}$ and $\text{fv}((\mathbf{rec} \mathcal{Z}_1.A_1)) = \text{fv}(A_1) \setminus \{\mathcal{Z}_1\}$, we have that X and $\mathcal{Z}$ are free in P_1 and A_1, respectively. By induction hypothesis, $\Gamma, X_1 : \mathcal{Z}_1 \vdash P' : A'$ and, by T-Rec, $\Gamma \vdash (\mathbf{rec} X_1.P_1) : (\mathbf{rec} \mathcal{Z}_1.A_1)$.

***Case* T-Con:** $P = P$
$B = \bigwedge_{i \in I} B_i$

The rule T-Con has the following assumption,

1. $\Gamma, X : \mathcal{Z} \vdash P : B_i$ for all $i \in I$.

By hypothesis, $\mathcal{Z}$ is free in $\bigwedge_{i\in I} B_i$. So, $\mathcal{Z}$ is free in B_i for every $i \in I$, by definition. Then, by induction hypothesis, $\Gamma \vdash P : B_i$ for every $i \in I$. By T-Con, $\Gamma \vdash P : \bigwedge_{i\in I} B_i$.

So we conclude that $\Gamma \vdash P : B$. Then, by the subsumption rule and since $B <: A$, we have that $\Gamma \vdash P : A$. □

Lemma 4.1.4. *If X is free in P and $\Gamma \vdash P : A$, then there are $\mathcal{Z}$ and Γ' such that $\mathcal{Z}$ is free in A and Γ is equal to $\Gamma', X : \mathcal{Z}$.*

Proof. Immediate, by construction. Just notice that the only rule that introduces free variables is the rule T-Var and its environment contains at least these variables. All other rules preserve the environment. □

Theorem 4.1.5. (Typability) *If P is a process, then there is a type A and an environment Γ such that $\Gamma \vdash P : A$ is derived structurally.*

Proof. By induction on the structure of P.

***Case* Inactive:** $P = \mathbf{0}$

Immediate from rule T-Voi, $\Gamma \vdash \mathbf{0} : \mathbf{0}$.

***Case* Parallel:** $P = P_1 \mid P_2$

By induction hypothesis there are A_1, A_2, Γ_1 and Γ_2 such that $\Gamma_1 \vdash P_1 : A_1$ and $\Gamma_2 \vdash P_2 : A_2$. At this point, we can assume that Γ_1 and Γ_2 are the same, because we want each process variable common to P_1 and P_2 to be mapped to the same type variable, and of course we can use weakening to produce the same Γ. So $\Gamma_1 \vdash P_1 : A_1$ and $\Gamma_2 \vdash P_2 : A_2$. Then, using the T-Par rule we obtain $\Gamma_1 \vdash P_1 \mid P_2 : A_1 \mid A_2$.

***Case* Restriction:** $P = (\nu a)P_1$

By induction hypothesis there are A_1 and Γ_1 such that $\Gamma_1 \vdash P_1 : A_1$ is derived structurally. By rule T-New, $\Gamma_1 \vdash (\nu a)P_1 : a®A_1$.

***Case* Prefix:** $P = \pi.P_1$

By induction hypothesis there are A_1 and Γ_1 such that $\Gamma_1 \vdash P_1 : A_1$ is derived structurally. By rule T-Pre, $\Gamma_1 \vdash \pi.P_1 : \pi.A_1$.

***Case* Recursion:** $P = (\mathbf{rec}X.P_1)$

By induction hypothesis there are A_1 and Γ_1 such that $\Gamma_1 \vdash P_1 : A_1$ is derived structurally. Notice that X is free in P_1. Then, by lemma 4.1.4, there are $\mathcal{Z}$ and Γ' such that $\mathcal{Z}$ is free in A_1 and $\Gamma', X : \mathcal{Z} \vdash P_1 : A_1$. By rule T-Rec, $\Gamma' \vdash (\mathbf{rec}X.P_1) : (\mathbf{rec}\mathcal{Z}.A_1)$.

***Case* Process Variable:** $P = X$

Let $\mathcal{Z}$ be any type variable, and let Γ be $\Gamma', X : \mathcal{Z}$, where Γ' is chosen so that Γ is a well defined environment. Then, by T-Var, $\Gamma', X : \mathcal{Z} \vdash X : \mathcal{Z}$. □

Lemma 4.1.6. (Substitution of variables) *If* $\Gamma, X : \mathcal{Z} \vdash P : A$ *and* $\Gamma \vdash Q : B$*, then* $\Gamma \vdash P\{Q/X\} : A\{B/\mathcal{Z}\}$.

Proof. If $\Gamma, X : \mathcal{Z} \vdash P : A$, then, by Lemma 4.1.1, there exists a type C such that $C <: A$ and $\Gamma, X : \mathcal{Z} \vdash P : C$ is derived using a structural rule or the conjunction rule in the last step of the derivation.

First, we prove that $\Gamma \vdash P\{Q/X\} : C\{B/\mathcal{Z}\}$, by induction on a derivation of $\Gamma, X : \mathcal{Z} \vdash P : C$ in the Type System. For a given derivation, we proceed by a case analysis on the final structural type rule or the conjunction rule used.

***Case* T-Voi:** $P = \mathbf{0}$
$C = \mathbf{0}$

Immediate from the hypothesis, noting that, by definition of substitution, $\mathbf{0}\{Q/X\} = \mathbf{0}$ and $\mathbf{0}\{B/\mathcal{Z}\} = \mathbf{0}$.

***Case* T-Var:** $P = X_1$
$C = \mathcal{Z}_1$

We have four subcases to consider resulting from whether X_1 is X or not and $\mathcal{Z}_1$ is $\mathcal{Z}$ or not. First, note that the cases where X_1 is X but $\mathcal{Z}_1$ is not $\mathcal{Z}$, and vice-versa, can not happen, by definition of Γ.

So, let $X_1 = X$ and, $\mathcal{Z}_1 = \mathcal{Z}$. By definition of substitution, we have that $X\{Q/X\} = Q$ and $\mathcal{Z}\{B/\mathcal{Z}\} = B$, and the result follows directly from the hypothesis.

Now let $X_1 \neq X$ and $\mathcal{Z}_1 \neq \mathcal{Z}$. By definition of substitution, we have that $X_1\{Q/X\} = X_1$ and $\mathcal{Z}_1\{B/\mathcal{Z}\} = \mathcal{Z}_1$, and the result follows directly from the hypothesis.

***Case* T-Pre:** $P = \pi.P'$
$C = \pi.A'$

The rule T-Pre has the following assumption,

1. $\Gamma, X : \mathcal{Z} \vdash P' : A'$.

By induction hypothesis, $\Gamma \vdash P'\{Q/X\} : A'\{B/\mathcal{Z}\}$. Subsequently, applying T-Pre, we get $\Gamma \vdash \pi.(P'\{Q/X\}) : \pi.(A'\{B/\mathcal{Z}\})$. But, by definition of substitution, $\pi.(P'\{Q/X\}) = (\pi.P')\{Q/X\}$ and $\pi.(A'\{B/\mathcal{Z}\}) = (\pi.A')\{B/\mathcal{Z}\}$.

Hence, $\Gamma \vdash (\pi.P')\{Q/X\} : (\pi.A')\{B/\mathcal{Z}\}$.

***Case* T-Par:** $P = P_1 \mid P_2$
$C = A_1 \mid A_2$

The rule T-Par has the following two assumptions,

1. $\Gamma, X : \mathcal{Z} \vdash P_1 : A_1$,
2. $\Gamma, X : \mathcal{Z} \vdash P_2 : A_2$.

By induction hypothesis, from 1. and 2., we get

3. $\Gamma \vdash P_1\{Q/X\} : A_1\{B/\mathcal{Z}\}$,

4. $\Gamma \vdash P_2\{Q/X\} : A_2\{B/\mathcal{Z}\}$,

respectively. By rule T-Par, from 3. and 4., $\Gamma \vdash P_1\{Q/X\} \mid P_1\{Q/X\} : A_1\{B/\mathcal{Z}\} \mid A_2\{B/\mathcal{Z}\}$. But, by definition of substitution, we have that $P_1\{Q/X\} \mid P_1\{Q/X\} = (P_1 \mid P_2)\{Q/X\}$ and $A_1\{B/\mathcal{Z}\} \mid A_2\{B/\mathcal{Z}\} = (A_1 \mid A_2)\{B/\mathcal{Z}\}$. So $\Gamma \vdash (P_1 \mid P_2)\{Q/X\} : (A_1 \mid A_2)\{B/\mathcal{Z}\}$, as desired.

***Case* T-New:** $P = (\nu a)P'$
$C = a\circledR A'$

The rule T-New has the following assumption,

1. $\Gamma, X : \mathcal{Z} \vdash P' : A'$.

By induction hypothesis, $\Gamma \vdash P'\{Q/X\} : A'\{B/\mathcal{Z}\}$. Subsequently, applying T-New, we get $\Gamma \vdash (\nu a)(P'\{Q/X\}) : a\circledR(A'\{B/\mathcal{Z}\})$. But, by definition of substitution $(\nu a)(P'\{Q/X\}) = ((\nu a)P')\{Q/X\}$ and $a\circledR(A'\{B/\mathcal{Z}\}) = (a\circledR A')\{B/\mathcal{Z}\}$, so we obtain, $\Gamma \vdash ((\nu a)P')\{Q/X\} : (a\circledR A')\{B/\mathcal{Z}\}$, as desired.

***Case* T-Rec:** $P = (\mathbf{rec}X_1.P_1)$
$C = (\mathbf{rec}\mathcal{Z}_1.A_1)$

We have four subcases to consider resulting from whether X_1 is X or not and $\mathcal{Z}_1$ is $\mathcal{Z}$ or not.

First, note that the cases where X is X_1 or $\mathcal{Z}$ is $\mathcal{Z}_1$ can not happen. This results from the fact that if it could happen, then, by the assumption of T-Rec and by hypothesis, the system could derive, for instance, $\Gamma, X : \mathcal{Z}, X : \mathcal{Z} \vdash P_1 : A_1$ which violates the definition of the environment Γ.

So, X and $\mathcal{Z}$ are different from X_1 and $\mathcal{Z}_1$, respectively. Then, from T-Rec we know that $\Gamma, X : \mathcal{Z}, X_1 : \mathcal{Z}_1 \vdash P_1 : A_1$. Then, $\Gamma, X_1 : \mathcal{Z}_1 \vdash P_1\{Q/X\} : A_1\{B/\mathcal{Z}\}$, by induction hypothesis. By T-Rec, $\Gamma \vdash (\mathbf{rec}X_1.(P_1\{Q/X\})) : (\mathbf{rec}\mathcal{Z}_1.(A_1\{B/\mathcal{Z}\}))$. As, by definition, $(\mathbf{rec}X_1.(P_1\{Q/X\})) = (\mathbf{rec}X_1.P_1)\{Q/X\}$ and $(\mathbf{rec}\mathcal{Z}_1.(A_1\{B/\mathcal{Z}\})) = (\mathbf{rec}\mathcal{Z}_1.A_1)\{B/\mathcal{Z}\}$, we get $\Gamma \vdash (\mathbf{rec}X_1.P_1)\{Q/X\} : (\mathbf{rec}\mathcal{Z}_1.A_1)\{B/\mathcal{Z}\}$, as desired.

***Case* T-Con:** $P = P$
$C = \bigwedge_{i\in I} C_i$

The rule T-Con has the following assumption,

1. $\Gamma \vdash P : C_i$ for every $i \in I$.

By induction hypothesis, we have that

2. $\Gamma \vdash P : C_i\{B/\mathcal{Z}\}$ for every $i \in I$,

respectively. Then, by rule T-Con, $\Gamma \vdash P : \bigwedge_{i\in I}(C_i\{B/\mathcal{Z}\})$. As, by definition of substitution of variables in tupes, $\bigwedge_{i\in I}(C_i\{B/\mathcal{Z}\}) = (\bigwedge_{i\in I} C_i)\{B/\mathcal{Z}\}$, we get $\Gamma \vdash P : (\bigwedge_{i\in I} C_i)\{B/\mathcal{Z}\}$.

So we conclude that $\Gamma \vdash P\{Q/X\} : C\{B/\mathcal{Z}\}$. Since $C <: A$ we have that $C\{B/\mathcal{Z}\} <: A\{B/\mathcal{Z}\}$. Then, by the subsumption rule, $\Gamma \vdash P\{Q/X\} : A\{B/\mathcal{Z}\}$. □

Lemma 4.1.7. *If $\Gamma \vdash P : A$ and b is fresh in P, then:*

1. *b is fresh in A', for some A' such that $A' \equiv^t_\alpha A$;*
2. $\Gamma \vdash P\{b/a\} : A\{b/a\}$.

Proof. If $\Gamma \vdash P : A$, then, by Lemma 4.1.1, there exists a type B such that $B <: A$ and $\Gamma \vdash P : B$ is derived using a structural rule or the conjunction rule in the last step of the derivation.

First, we prove that b is fresh in B and $\Gamma \vdash P\{b/a\} : B\{b/a\}$, by induction on a derivation of $\Gamma \vdash P : B$ in the Type System. For a given derivation, we proceed by a case analysis on the final structural type rule or the conjunction rule used.

***Case* T-Voi:** $P = \mathbf{0}$
$B = \mathbf{0}$

Part (1.): Immediate, noting that by definition of free and bound names in types, $\text{fn}(\mathbf{0}) = \emptyset$ and $\text{bn}(\mathbf{0}) = \emptyset$, respectively.

Part (2.): Immediate, noting that by definition of substitution in both processes and types, $\mathbf{0}\{b/a\} = \mathbf{0}$.

***Case* T-Var:** $P = X$
$B = \mathcal{Z}$

Part (1.): Immediate, noting that by definition of free and bound names in types, $\text{fn}(X) = \emptyset$ and $\text{bn}(X) = \emptyset$, respectively.

Part (2.): Immediate, noting that by definition of substitution in processes and types, $X\{b/a\} = X$ and $\mathcal{Z}\{b/a\} = \mathcal{Z}$, respectively.

***Case* T-Pre:** $P = \pi.P'$
$B = \pi.A'$

The rule T-Pre has the following assumption,

1. $\Gamma \vdash P' : A'$.

Part (1.): We have two consider two cases: whether π is τ or it is an α.

First, consider that π is τ. By hypothesis, $b \notin \text{fn}(\tau.P') = \text{fn}(P')$ and $b \notin \text{bn}(\tau.P') = \text{bn}(P')$. Then, b is fresh in P'. By induction hypothesis, w infer that b is free in A'. So $b \notin \text{fn}(A') = \text{fn}(\tau.A')$ and $b \notin \text{bn}(A') = \text{bn}(\tau.A')$, that is b is fresh in $\tau.A'$.

Secondly, consider that π is an α. By hypothesis, $b \notin \text{fn}(\alpha.P') = \{\alpha\} \cup \text{fn}(P')$ and $b \notin \text{bn}(\alpha.P') = \text{bn}(P')$. Then $b \neq \alpha$ and b is fresh in P'. So, by induction hypothesis we infer that b is fresh in A', that is $b \notin \text{fn}(A')$ and $b \notin \text{bn}(A')$. As $b \neq \alpha$, it is also true that $b \notin \{\alpha\} \cup \text{fn}(A') = \text{fn}(\alpha.A')$. So b is fresh in $\alpha.A'$.

Part (2.): We have two major cases: whether π is τ or it is an α.

First, consider that π is τ. Then, as b is fresh in P, by definition, $b \notin \text{fn}(\tau.P') = \text{fn}(P')$ and $b \notin \text{bn}(\tau.P') = \text{bn}(P')$. So, from this fact and 1., applying the induction hypothesis we obtain $\Gamma \vdash P'\{b/a\} :$

$A'\{b/a\}$. Then, applying the rule T-Pre, $\Gamma \vdash \tau.(P'\{b/a\}) : \tau.(A'\{b/a\})$. As $\tau.(P'\{b/a\}) = (\tau.P')\{b/a\}$ and $\tau.(A'\{b/a\}) = (\tau.A')\{b/a\}$, we conclude as desired that $\Gamma \vdash (\tau.P')\{b/a\} : (\tau.A')\{b/a\}$.

Secondly, consider that π is an α. As b is a fresh name in P, by definition, we have that: $b \notin \text{fn}(\alpha.P') = \{\alpha\} \cup \text{fn}(P')$ and $b \notin \text{bn}(\alpha.P') = \text{bn}(P')$. So, $b \notin \text{fn}(P')$ and $b \notin \text{bn}(P')$, in other words, b is fresh in P'. So, from this fact and 1., and applying the induction hypothesis, $\Gamma \vdash P'\{b/a\} : A'\{b/a\}$. At this point we have to consider three subcases: whether α is a or $\bar{a}$ or not.
(*subcase $\alpha = a$*): By T-Pre, $\Gamma \vdash b.(P'\{b/a\}) : b.(A'\{b/a\})$. As $b.(P'\{b/a\}) = (a.P')\{b/a\}$ and $b.(A'\{b/a\}) = (a.A')\{b/a\}$, we have that $\Gamma \vdash (a.P')\{b/a\} : (a.A')\{b/a\}$.
(*subcase $\alpha = \bar{a}$*): By T-Pre, $\Gamma \vdash \bar{b}.(P'\{b/a\}) : \bar{b}.(A'\{b/a\})$. As $\bar{b}.(P'\{b/a\}) = (\bar{a}.P')\{b/a\}$ and $\bar{b}.(A'\{b/a\}) = (\bar{a}.A')\{b/a\}$, we have that $\Gamma \vdash (\bar{a}.P')\{b/a\} : (\bar{a}.A')\{b/a\}$.
(*subcase $\alpha \neq a$ and $\alpha \neq \bar{a}$*): By T-Pre, $\Gamma \vdash \alpha.(P'\{b/a\}) : \alpha.(A'\{b/a\})$. As, by definition, $\alpha.(P'\{b/a\}) = (\alpha.P')\{b/a\}$ and $\alpha.(A'\{b/a\}) = (\alpha.A')\{b/a\}$, we obtain the expected result, that $\Gamma \vdash (\alpha.P')\{b/a\} : (\alpha.A')\{b/a\}$.

***Case* T-Par:** $P = P_1 \mid P_2$
$B = A_1 \mid A_2$

The rule T-Par has the following two assumptions,

1. $\Gamma \vdash P_1 : A_1$,
2. $\Gamma \vdash P_2 : A_2$.

Note that $b \notin \text{fn}(P_1 \mid P_2) = \text{fn}(P_1) \cup \text{fn}(P_2)$ implies that $b \notin \text{fn}(P_1)$ and $b \notin \text{fn}(P_2)$ and similarly with bounded names.

Part (1.): By induction hypothesis, from 1. and 2., we get

3. b is fresh in A_1,
4. b is fresh in A_2,

respectively. So, from 3. and 4., we have that $b \notin \text{fn}(A_1) \cup \text{fn}(A_2) = \text{fn}(A_1 \mid A_2)$ and similarly with bounded names, so b is fresh in $A_1 \mid A_2$, as intended.

Part (2.): Note that $b \notin \text{fn}(P_1 \mid P_2) = \text{fn}(P_1) \cup \text{fn}(P_2)$ and $b \notin \text{bn}(P_1 \mid P_2) = \text{bn}(P_1) \cup \text{bn}(P_2)$ implies that $b \notin \text{fn}(P_1)$, $b \notin \text{fn}(P_2)$, $b \notin \text{bn}(P_1)$ and $b \notin \text{bn}(P_2)$. So, b is fresh in P_1 and in P_2.

By induction hypothesis, from 1. and 2., we get

5. $\Gamma \vdash P_1\{b/a\} : A_1\{b/a\}$,
6. $\Gamma \vdash P_2\{b/a\} : A_2\{b/a\}$,

respectively. So, from 5. and 6., by T-Par, $\Gamma \vdash P_1\{b/a\} \mid P_2\{b/a\} : A_1\{b/a\} \mid A_2\{b/a\}$. As, $P_1\{b/a\} \mid P_2\{b/a\} = (P_1 \mid P_2)\{b/a\}$ and $A_1\{b/a\} \mid A_2\{b/a\} = (A_1 \mid A_2)\{b/a\}$, we have that $\Gamma \vdash (P_1 \mid P_2)\{b/a\} : (A_1 \mid A_2)\{b/a\}$.

***Case* T-New:** $P = (\nu c)P'$
$B = c\circledR A'$

The rule T-New has the following assumption,

1. $\Gamma \vdash P' : A'$.

Part (1.): There are two subcases to consider: whether b is c, or not. In the first subcase, it is immediate that b is not fresh in P. In the second subcase, if b is different from c, then, $b \notin \text{fn}(P') \setminus \{c\}$ implies that $b \notin \text{fn}(P')$ and $b \notin \text{bn}((\nu c)P') = \{c\} \cup \text{bn}(P')$ implies that $b \notin \text{bn}(P')$. So, By induction hypothesis, b is fresh in A' which implies that $b \notin \text{fn}(A') \setminus \{c\} = \text{fn}(c\text{®}A')$. Moreover, as b is different from c, $b \notin \text{bn}(A') \cup \{c\} = \text{bn}(c\text{®}A')$ and we conclude that b is fresh in $c\text{®}A'$.

Part (2.): First note that, as b is fresh in P, $b \notin \text{fn}((\nu c)P') = \text{fn}(P') \setminus \{c\}$ and $b \notin \text{bn}((\nu c)P') = \text{bn}(P') \cup \{c\}$. In particular, b must be different from c. So b is fresh in P'.

By induction hypothesis, $\Gamma \vdash P'\{b/a\} : A'\{b/a\}$. Then, by T-New, $\Gamma \vdash (\nu c)(P'\{b/a\}) : c\text{®}(A'\{b/a\})$. Finally, as $(\nu c)(P'\{b/a\}) = ((\nu c)P')\{b/a\}$ and $c\text{®}(A'\{b/a\}) = (c\text{®}A')\{b/a\}$, $\Gamma \vdash ((\nu c)P')\{b/a\} : (c\text{®}A')\{b/a\}$.

***Case* T-Rec:** $P = (\textbf{rec}X.P')$
$B = (\textbf{rec}Z.A')$

The rule T-Rec has the following assumption,

1. $\Gamma, X : Z \vdash P' : A'$.

Note that, by definition, $\text{fn}((\textbf{rec}X.P')) = \text{fn}(P')$ and $\text{bn}((\textbf{rec}X.P')) = \text{bn}(P')$, so $b \notin \text{fn}(P')$ and $b \notin \text{bn}(P')$.

Part (1.): By induction hypothesis, b is fresh in A'. But, by definition, $\text{fn}(A') = \text{fn}((\textbf{rec}Z.A'))$ and $\text{bn}(A') = \text{bn}((\textbf{rec}Z.A'))$, so, equally, $b \notin \text{fn}((\textbf{rec}Z.A'))$ and $b \notin \text{bn}((\textbf{rec}Z.A'))$, that is it is fresh in $(\textbf{rec}Z.A')$.

Part (2.): By induction hypothesis, $\Gamma, X : Z \vdash P'\{b/a\} : A'\{b/a\}$. Then, by T-Rec, $\Gamma \vdash (\textbf{rec}X.(P'\{b/a\})) : (\textbf{rec}Z.(A'\{b/a\}))$. As, $(\textbf{rec}X.(P'\{b/a\})) = ((\textbf{rec}X.P')\{b/a\}$ and $(\textbf{rec}Z.(A'\{b/a\})) = (\textbf{rec}Z.A')\{b/a\}$, we have $\Gamma \vdash (\textbf{rec}X.P')\{b/a\} : (\textbf{rec}Z.A')\{b/a\}$.

***Case* T-Con:** $P = P$
$B = \bigwedge_{i \in I} B_i$

The rule T-Con has the following assumption,

1. $\Gamma \vdash P : B_i$ for every $i \in I$.

Note that, by definition, $\text{fn}(\bigwedge_{i \in I} B_i) = \bigcup_{i \in I} \text{fn}(B_i)$ and $\text{bn}(\bigwedge_{i \in I} B_i) = \bigcup_{i \in I} \text{bn}(B_i)$, so $b \notin \text{fn}(B_i), \text{bn}(B_i)$ for every $i \in I$.

Part (1.): By induction hypothesis, b is fresh in B_i for every $i \in I$, respectively. But, by definition, $\bigcup_{i \in I} \text{fn}(B_i) = \text{fn}(\bigwedge_{i \in I} B_i)$ and $\bigcup_{i \in I} \text{bn}(B_i) = \text{bn}(\bigwedge_{i \in I} B_i)$, so, equally, $b \notin \text{fn}(\bigwedge_{i \in I} B_i)$ and $b \notin \text{bn}(\bigwedge_{i \in I} B_i)$, that is it is fresh in $\bigwedge_{i \in I} B_i$.

Part (2.): By induction hypothesis, $\Gamma \vdash P\{b/a\} : B_i\{b/a\}$ for every $i \in I$. Then, by T-Con, $\Gamma \vdash P\{b/a\} : \bigwedge_{i \in I} (B_i\{b/a\})$. As, $\bigwedge_{i \in I} (B_i\{b/a\}) = (\bigwedge_{i \in I} B_i)\{b/a\}$, we have $\Gamma \vdash P\{b/a\} : (\bigwedge_{i \in I} B_i)\{b/a\}$.

So we conclude that b is fresh in B and $\Gamma \vdash P\{b/a\} : B\{b/a\}$. For the first part of the Lemma, the fact that b is fresh in B and that $B <: A$ do not imply that b is fresh in A. Nevertheless, we can always use

a suitable renaming of names of A. Therefore, there exists A' such that $A' \equiv^t_\alpha A$ and b is fresh in A'. For the second part of the Lemma, since $B <: A$ we have that $B\{b/a\} <: A\{b/a\}$. Then, by the subsumption rule, $\Gamma \vdash P\{b/a\} : A\{b/a\}$. □

Proposition 4.1.8. *If $\Gamma \vdash P : A$ and $P \equiv Q$, then $\Gamma \vdash Q : B$, for some B such that $B \equiv^t A$.*

Proof. By induction on a derivation of $P \equiv Q$ in the Structural Congruence Relation. For a given derivation, we proceed by a case analysis on the final derivation rule used.

***Case* 1:** $P \equiv_\alpha Q$

We have to consider two subcases: *Alpha Res* and *Alpha Rec* (in both directions).

***Subcase* 1.1.a):** $(\nu a)P' \equiv_\alpha (\nu b)(P'\{b/a\})$, where b is fresh in P'

By T-New, $A = a\circledR A'$ and $\Gamma \vdash P' : A'$. Then, by lemma 4.1.7.(2), $\Gamma \vdash P'\{b/a\} : A'\{b/a\}$, as b is fresh in P'. By T-New, $\Gamma \vdash (\nu b)(P'\{b/a\}) : b\circledR(A'\{b/a\})$.

To see that $b\circledR(A'\{b/a\}) \equiv^t a\circledR A'$, notice that, by lemma 4.1.7.(1), b is fresh in A' which implies that $b\circledR(A'\{b/a\}) \equiv^t_\alpha a\circledR A'$.

***Subcase* 1.1.b):** $(\nu b)(P'\{b/a\}) \equiv_\alpha (\nu a)P'$, where b is fresh in P'

By T-New, $A = a\circledR A'$ and $\Gamma \vdash P'\{b/a\} : A'$. It is not hard to notice that a is fresh in $P'\{b/a\}$ and $P'\{b/a\}\{a/b\} = P'$. Hence, by lemma 4.1.7, we have that $\Gamma \vdash P' : A'\{a/b\}$ and a is fresh in A'. Then, by T-New, $\Gamma \vdash (\nu a)P' : a\circledR(A'\{a/b\})$. Moreover, as a is fresh in A', by definition of *Alpha t-Res*, $a\circledR(A'\{a/b\}) \equiv^t_\alpha b\circledR A'$, and consequently, by Structural congruence, $a\circledR(A'\{a/b\}) \equiv^t b\circledR A'$

***Subcase* 1.2.a):** $(\mathbf{rec}X.P') \equiv_\alpha (\mathbf{rec}Y.P'\{Y/X\})$, where Y is fresh in P'

By T-Rec, $A = (\mathbf{rec}\mathcal{Z}.A')$ and $\Gamma, X : \mathcal{Z} \vdash P' : A'$. Let $\mathcal{W}$ be fresh in A'. Then, by T-Var, we know that $\Gamma, Y : \mathcal{W} \vdash Y : \mathcal{W}$. From lemma 4.1.2, as Y and $\mathcal{W}$ are fresh in P' and A', respectively, $\Gamma, X : \mathcal{Z}, Y : \mathcal{W} \vdash P' : A'$. So, by lemma 4.1.6, $\Gamma, Y : \mathcal{W} \vdash P'\{Y/X\} : A'\{\mathcal{W}/\mathcal{Z}\}$. Then, by T-Rec, $\Gamma \vdash (\mathbf{rec}Y.P'\{Y/X\}) : (\mathbf{rec}\mathcal{W}.A'\{\mathcal{W}/\mathcal{Z}\})$. Moreover, as $\mathcal{W}$ is fresh in A', $(\mathbf{rec}\mathcal{W}.A'\{\mathcal{W}/\mathcal{Z}\}) \equiv^t_\alpha (\mathbf{rec}\mathcal{Z}.A')$ and, consequently, $(\mathbf{rec}\mathcal{W}.A'\{\mathcal{W}/\mathcal{Z}\}) \equiv^t (\mathbf{rec}\mathcal{Z}.A')$, as intended.

***Subcase* 1.2.b):** $(\mathbf{rec}Y.P'\{Y/X\}) \equiv_\alpha (\mathbf{rec}X.P')$, where Y is fresh in P'

By T-Rec, $A = (\mathbf{rec}\mathcal{W}.A')$ and $\Gamma, Y : \mathcal{W} \vdash P'\{Y/X\} : A'$. Let $\mathcal{Z}$ be fresh in A'. Then, by T-Var, we know that $\Gamma, X : \mathcal{Z} \vdash X : \mathcal{Z}$. Clearly X is fresh in $P'\{Y/X\}$. By lemma 4.1.2, $\Gamma, Y : \mathcal{W}, X : \mathcal{Z} \vdash P'\{Y/X\} : A'$. By definition of substitution $P'\{Y/X\}\{X/Y\} = P'$. Hence, by lemma 4.1.6, $\Gamma, X : \mathcal{Z} \vdash P' : A'\{\mathcal{Z}/\mathcal{W}\}$ and, by T-Rec, $\Gamma \vdash (\mathbf{rec}X.P) : (\mathbf{rec}\mathcal{Z}.A'\{\mathcal{Z}/\mathcal{W}\})$. Moreover, as $\mathcal{Z}$ is fresh in A', $(\mathbf{rec}\mathcal{Z}.A'\{\mathcal{Z}/\mathcal{W}\}) \equiv^t_\alpha (\mathbf{rec}\mathcal{W}.A')$ and, consequently, we conclude that $(\mathbf{rec}\mathcal{Z}.A'\{\mathcal{Z}/\mathcal{W}\}) \equiv^t (\mathbf{rec}\mathcal{W}.A')$.

***Case* 2.a):** $P' \mid \mathbf{0} \equiv P'$

By T-Par, we must have $A = A_1 \mid A_2$, for some A_1 and A_2, and the following assumptions

$\Gamma \vdash P' : A_1$, for some A_1

$\Gamma \vdash \mathbf{0} : A_2$, for some A_2

By T-Voi we conclude that $A_2 = \mathbf{0}$ and as $A_1 \equiv^t A_1 \mid 0$ we get the result.

***Case* 2.b):** $P' \equiv P' \mid \mathbf{0}$

By T-Par and T-Voi, we have that

$\Gamma \vdash P \mid \mathbf{0} : A \mid \mathbf{0}$.

As $A \mid 0 \equiv^t A$ we get the required.

***Case* 3.a):** $P_1 \mid P_2 \equiv P_2 \mid P_1$

By T-Par, we must have that $A = A_1 \mid A_2$, for some A_1 and A_2, and the following

$\Gamma \vdash P_1 : A_1$,

$\Gamma \vdash P_2 : A_2$.

Again by T-Par, we have that $\Gamma \vdash P_2 \mid P_1 : A_2 \mid A_1$. So, as $A_2 \mid A_1 \equiv^t A_1 \mid A_2$ we get the required.

***Case* 3.b):** $P_2 \mid P_1 \equiv P_1 \mid P_2$

Similar to 3.a).

***Case* 4.a):** $P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$

By T-Par, we must have that $A = A_1 \mid A_2$, for some A_1 and A_2, and the following

1. $\Gamma \vdash P_1 : A_1$,
2. $\Gamma \vdash P_2 \mid P_3 : A_2$.

Again by T-Par, this time applied to 2., we must have that $A_2 = A_{21} \mid A_{22}$, for some A_{21} and A_{22}, and the following

3. $\Gamma \vdash P_2 : A_{21}$,
4. $\Gamma \vdash P_3 : A_{22}$.

Once more by applying T-Par to 1. and 3. we get

5. $\Gamma \vdash P_1 \mid P_2 : A_1 \mid A_{21}$.

Yet another appliance of T-Par, this time to 5. and 4., yields

6. $\Gamma \vdash (P_1 \mid P_2) \mid P_3 : (A_1 \mid A_{21}) \mid A_{22}$.

As $(A_1 \mid A_{21}) \mid A_{22} \equiv^t A_1 \mid (A_{21} \mid A_{22})$, we get the required.

***Case* 4.b):** $(P_1 \mid P_2) \mid P_3 \equiv P_1 \mid (P_2 \mid P_3)$

Similar to 4.a).

***Case* 5.a):** $(\nu a)\mathbf{0} \equiv \mathbf{0}$

By T-New, we must have that $A = a\circledR A_1$, for some A_1 and a, and the following

$$\Gamma \vdash (\nu a)\mathbf{0} : A_1$$

By T-Voi $A_1 = \mathbf{0}$, and, as $\mathbf{0} \equiv^t a\circledR A_1$, we get the required.

***Case* 5.b):** $\mathbf{0} \equiv (\nu a)\mathbf{0}$

By T-Voi $A = \mathbf{0}$ and $\Gamma \vdash \mathbf{0} : \mathbf{0}$. Then, by T-New, we have that $\Gamma \vdash (\nu a)\mathbf{0} : a\circledR\mathbf{0}$. As $a\circledR\mathbf{0} \equiv^t \mathbf{0}$, we get the required.

***Case* 6.a):** $(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$

By T-New, $A = a\circledR A'$ and $\Gamma \vdash (\nu b)P : A'$. Again by T-New, $A' = a\circledR A''$ and $\Gamma \vdash P : A''$. Now applying the rule T-New two more times we obtain $\Gamma \vdash (\nu a)P : a\circledR A''$ and $\Gamma \vdash (\nu b)(\nu a)P : b\circledR(a\circledR A'')$. Now, as $b\circledR(a\circledR A'') \equiv^t a\circledR(b\circledR A'')$, we get the required.

***Case* 6.b):** $(\nu b)(\nu a)P \equiv (\nu a)(\nu b)P$

Similar to 6.a).

***Case* 7.a):** $(\nu a)(P_1 \mid P_2) \equiv ((\nu a)P_1) \mid P_2$, where $a \notin \text{fn}(P_2)$

First, notice that $(\nu a)(P_1 \mid P_2) \equiv ((\nu a)P_1) \mid P_2$ has a side-condition: $a \notin \text{fn}(P_2)$. If a appears bounded in P_2 we can use renaming; therefore assume that a is not bounded in P_2. Moreover, a is fresh in P_2. By T-New, from the hypothesis, we must have that $A = a\circledR A'$ and $\Gamma \vdash P_1 \mid P_2 : A'$. Then, by T-Par, we must have that $A' = A'_1 \mid A'_2$ and the following

1. $\Gamma \vdash P_1 : A'_1$,
2. $\Gamma \vdash P_2 : A'_2$.

From 1., by T-New, $\Gamma \vdash (\nu a)P_1 : a\circledR A'_1$. Then, $\Gamma \vdash ((\nu a)P_1) \mid P_2 : (a\circledR A'_1) \mid A'_2$, using 2. and T-Par. By lemma 4.1.7.(1), as a is fresh in P_2, we have that a is fresh in A'_2. In particular, a is free in A'_2 and, consequently, $(a\circledR A'_1) \mid A'_2 \equiv^t a\circledR(A'_1 \mid A'_2)$, as desired.

***Case* 7.b):** $((\nu a)P_1) \mid P_2 \equiv (\nu a)(P_1 \mid P_2)$, where $a \notin \text{fn}(P_2)$

First, notice that $((\nu a)P_1) \mid P_2 \equiv (\nu a)(P_1 \mid P_2)$ has a side-condition: $a \notin \text{fn}(P_2)$. Again, if a appears bounded in P_2 we can use renaming; therefore assume that a is not bounded in P_2. Moreover, a is fresh in P_2. By T-Par, we must have that $A = A_1 \mid A_2$ and

1. $\Gamma \vdash (\nu a)P_1 : A_1$,
2. $\Gamma \vdash P_2 : A_2$.

By T-New, from 1., we must have that $A_1 = a\circledR A'_1$ and $\Gamma \vdash P_1 : A'_1$. Then, $\Gamma \vdash P_1 \mid P_2 : A'_1 \mid A_2$, using 2. and T-Par. Finally, by T-New, $\Gamma \vdash (\nu a)(P_1 \mid P_2) : a\circledR(A'_1 \mid A_2)$. By lemma 4.1.7.(1), as a is fresh in P_2, we have that a is fresh in A_2. In particular, a is free in A_2 and, consequently, $a\circledR(A'_1 \mid A'_2) \equiv^t a\circledR A'_1 \mid A'_2$, as desired. □

4.2 Main results

This section presents two main properties of the system.

4.2.1 Consistency

Theorem 4.2.1. (Consistency of the Type System) *If $\Gamma \vdash P : A$, then there exists a process Q such that $P <: Q$ and $Q \models_\Gamma A$, where Γ is a well-defined environment.*

Proof. If $\Gamma \vdash P : A$, then, by Lemma 4.1.1, there exists a type B such that $B <: A$ and $\Gamma \vdash P : B$ is derived using a structural rule or the conjunction rule in the last step of the derivation.

First, we prove that $P \models_\Gamma B$, by induction on a derivation of $\Gamma \vdash P : B$ in the Type System. For a given derivation, we proceed by a case analysis on the final structural type rule or the conjunction rule used.

***Case* T-Voi:** $P = \mathbf{0}$
$B = \mathbf{0}$

Immediate, noting that by definition of satisfaction $\mathbf{0} \models_\Gamma \mathbf{0}$.

***Case* T-Var:** $P = X$
$B = \mathcal{Z}$

By hypothesis, $\Gamma, X : \mathcal{Z} \vdash X : \mathcal{Z}$. Let $\Gamma' = \Gamma, X : \mathcal{Z}$. Then, by definition of judgement, $\Gamma'(X) = \mathcal{Z}$ and, consequently, $X \models_{\Gamma'} \mathcal{Z}$, *i.e.* $X \models_{\Gamma,\{X:\mathcal{Z}\}} \mathcal{Z}$.

***Case* T-Pre:** $P = \pi.P'$
$B = \pi.A'$

The rule T-Pre has the following assumption,

1. $\Gamma \vdash P' : A'$.

By induction hypothesis, $P' \models_\Gamma A'$.

First, note that, by rule Act of the LTS, $\pi.P' \xrightarrow{\pi} P'$.

Secondly, let π' be any action and Q be any process such that $\pi.P' \xrightarrow{\pi'} Q$. By the LTS it is true that the only rules which have $\pi.P' \xrightarrow{\pi'} Q$ in the conclusion are rules Act and SC. If rule Act was the one used, then $\pi' = \pi$ and $Q = P'$ which implies $Q \approx P'$. If rule SC was the one used, then there are P'' and Q' such that $\pi.P' \equiv P''$, $Q \equiv Q'$ and $P'' \xrightarrow{\pi'} Q'$. Note that the only such P'' is $\pi.P' \mid \mathbf{0}$. Then, using rule SC again and rule Act yields the same result.

Then, by definition of satisfaction, we have that $\pi.P' \models_\Gamma \pi.A'$.

***Case* T-Par:** $P = P_1 \mid P_2$
$B = A_1 \mid A_2$

The rule T-Par has the following two assumptions,

1. $\Gamma \vdash P_1 : A_1$,
2. $\Gamma \vdash P_2 : A_2$.

By induction hypothesis, from 1. and 2., we get

3. $P_1 \models_\Gamma A_1$,

4. $P_2 \models_\Gamma A_2$,

respectively. As $P = P_1 \mid P_2$, we have that $P \equiv P_1 \mid P_2$. Then, by definition of satisfaction, from 3. and 4., $P_1 \mid P_2 \models_\Gamma A_1 \mid A_2$.

***Case* T-New:** $P = (\nu a)P'$
$B = a\circledR A'$

The rule T-New has the following assumption,

1. $\Gamma \vdash P' : A'$.

By induction hypothesis, $P' \models_\Gamma A'$. Then, by definition of satisfaction, we have that $(\nu a)P' \models_\Gamma a\circledR A'$.

***Case* T-Rec:** $P = (\mathbf{rec}X.P')$
$B = (\mathbf{rec}Z.A')$

The rule T-Rec has the following assumption,

1. $\Gamma, X : Z \vdash P' : A'$.

By induction hypothesis, $P' \models_{\Gamma,\{X:Z\}} A'$. Then, by definition of satisfaction, we have that $(\mathbf{rec}X.P') \models_\Gamma (\mathbf{rec}Z.A')$. It is easy to see that Γ must be an well-defined environment, by (1.).

***Case* T-Con:** $P = P$
$B = \bigwedge_{i\in I} B_i$

If $\Gamma \vdash P : \bigwedge_{i\in I} B_i$, then, by the conjunction rule, $\Gamma \vdash P : B_i$ for every $i \in I$. Then, by induction hypothesis, $P \models_\Gamma B_i$ for every $i \in I$, respectively. Finally, by definition of interpretation, $P \models_\Gamma \bigwedge_{i\in I} B_i$.

So we conclude that $P \models_\Gamma B$. Then, since $B <: A$ we have that, by definition of subtyping, there exists Q such that $P <: Q$ and $Q \models_\Gamma A$ as desired. □

Note that consistency is actually weak consistency. This is due to the subsumption rule which ties subtyping to the system. Hence, the system must comply with the definition of subtyping, which leads us to the previous result.

Corollary 4.2.2. (Consistency) *If $\vdash P : A$, then there exists a process Q such that $P <: Q$ and $Q \models A$.*

Proof. Direct from 4.2.1, with $\Gamma = \emptyset$. □

4.2.2 Completeness

Lemma 4.2.3. *If $P \xrightarrow{\pi} P'$ and $\Gamma \vdash P' : A'$, then $\Gamma \vdash P : A$ for some A such that $A <: \langle \pi \rangle A'$.*

Proof. By induction on a derivation of $P \xrightarrow{\pi} Q$ in the Labelled Transition System. For a given derivation, we proceed by a case analysis on the final transition rule used in the proof.

***Case* Pref:** $P = \pi.P'$

By hypothesis, $\Gamma \vdash P' : A'$. Then, by T-Pref, $\Gamma \vdash \pi.P' : \pi.A'$ and $\pi.A' <: \langle \pi \rangle A'$, by Proposition 3.3.11.

***Case* Par1:** $P = P_1 \mid Q$
$P' = P_2 \mid Q$

By hypothesis, $P_1 \xrightarrow{\pi} P_2$. As $\Gamma \vdash P_2 \mid Q : A'$, we have that $A' = A_1 \mid A_2$, $\Gamma \vdash P_2 : A_1$ and $\Gamma \vdash Q : A_2$. By induction hypothesis, $\Gamma \vdash P_1 : B$ and $B <: \langle \pi \rangle A_1$. Then, by T-Par, $\Gamma \vdash P_1 \mid Q : B \mid A_2$; that is $\Gamma \vdash P : A$, where $A = B \mid A_2$. Moreover, by Proposition 3.3.11, as $B <: \langle \pi \rangle A_1$, we have that $B \mid A_2 <: \langle \pi \rangle (A_1 \mid A_2)$; that is, $A <: \langle \pi \rangle A'$.

***Case* Par2:** $P = P_1 \mid P_2$
$P' = P_1' \mid P_2'$

By hypothesis, $P_1 \xrightarrow{\alpha} P_1'$ and $P_2 \xrightarrow{\bar{\alpha}} P_2'$, for some α. As $\Gamma \vdash P_1' \mid P_2' : A'$, we have that $A' = A_1' \mid A_2'$, $\Gamma \vdash P_1' : A_1'$ and $\Gamma \vdash P_2' : A_2'$. By induction hypothesis, $\Gamma \vdash P_1 : A_1$, $\Gamma \vdash P_2 : A_2$, $A_1 <: \langle \alpha \rangle A_1'$ and $A_2 <: \langle \bar{\alpha} \rangle A_2'$. Then, by T-Par, $\Gamma \vdash P_1 \mid P_2 : A_1 \mid A_2$; that is $\Gamma \vdash P : A$, where $A = A_1 \mid A_2$. Moreover, by Proposition 3.3.11, as $A_1 <: \langle \alpha \rangle A_1'$ and $A_2 <: \langle \bar{\alpha} \rangle A_2'$, we have that $A_1 \mid A_2 <: \langle \tau \rangle (A_1' \mid A_2')$; that is, $A <: \langle \pi \rangle A'$.

***Case* Res:** $P = (\nu a)Q$
$P' = (\nu a)Q'$

By hypothesis, $Q \xrightarrow{\pi} Q'$, where a is different from π and $\bar{\pi}$. As $\Gamma \vdash (\nu a)Q' : A'$, we have that $A' = a\circledR A''$ and $\Gamma \vdash Q' : A''$. By induction hypothesis, $\Gamma \vdash Q : B$ and $B <: \langle \pi \rangle A''$. Then, by T-New, $\Gamma \vdash (\nu a)Q : a\circledR B$; that is $\Gamma \vdash P : A$, where $A = a\circledR B$. Moreover, by Proposition 3.3.8, as $B <: \langle \pi \rangle A''$, we have that $a\circledR B <: a\circledR(\langle \pi \rangle A'')$. Subsequently, as a is different from π and $\bar{\pi}$, we have that $a\circledR(\langle \pi \rangle A'') <: \langle \pi \rangle (a\circledR A'')$, by Proposition 3.3.11. Finally, by transitivity, $A <: \langle \pi \rangle A'$.

***Case* Rec:** $P = (\mathbf{rec} X.Q)$
$P' = Q'$

By hypothesis, $(\mathbf{rec} X.Q) \xrightarrow{\pi} Q'$ and $\Gamma \vdash Q' : A'$. Then $Q\{(\mathbf{rec} X.Q)/X\} \xrightarrow{\pi} Q'$, by rule T-Rec. By induction hypothesis we have that $\Gamma \vdash Q\{(\mathbf{rec} X.Q)/X\} : B$ and $B <: \langle \pi \rangle A'$, for some type B.

On the other hand, by Theorem 4.1.5, there is a type A such that $\Gamma \vdash (\mathbf{rec} X.Q) : A$. Then, by rule T-Rec, $\Gamma, X : Z \vdash Q : A_1$ and $A = (\mathbf{rec} Z.A_1)$. Therefore, by Proposition 4.1.6, $\Gamma \vdash Q\{(\mathbf{rec} X.Q)/X\} : A_1\{(\mathbf{rec} Z.A_1)/Z\}$. It is simple to observe that $A_1\{(\mathbf{rec} Z.A_1)/Z\} <: B$. By Proposition 3.3.11, $(\mathbf{rec} Z.A_1) <: A_1\{(\mathbf{rec} Z.A_1)/Z\}$, so, by transitivity of the subtyping relation, $(\mathbf{rec} Z.A_1) <: B$ and, again by transitivity, $(\mathbf{rec} Z.A_1) <: \langle \pi \rangle A'$; that is, $A <: \langle \pi \rangle A'$.

***Case* SC:** $P = P_1$
$P' = Q_1$

By hypothesis, $P_1 \equiv P_2$, $Q_1 \equiv Q_2$ and $P_2 \xrightarrow{\pi} Q_2$. As $\Gamma \vdash Q_1 : A'$ and $Q_1 \equiv Q_2$, by lemma 4.1.8, there exists a type B_1 such that $\Gamma \vdash Q_2 : B_1$ and $A' \equiv^t B_1$. Then, By induction hypothesis, $\Gamma \vdash P_2 : B_2$ and $B_2 <: \langle\pi\rangle B_1$. Again, as $P_2 \equiv P_1$, we have, by lemma 4.1.8, that there exists a type A such that $\Gamma \vdash P_1 : A$ and $B_2 \equiv^t A$.

One can conclude that $A <: \langle\pi\rangle A'$ by repeated use of the following three results: Propositions 3.3.8 and Proposition 3.3.9, and transitivity of $<:$. First, by Proposition 3.3.9, as $B_2 \equiv^t A$, we have that $A <: B_2$. Then using transitivity and the fact that $B_2 <: \langle\pi\rangle B_1$, we conclude $A <: \langle\pi\rangle B_1$. On the other hand, by Proposition 3.3.9, from $A' \equiv^t B_1$, we have that $B_1 <: A'$. Subsequently, by Proposition 3.3.8, $\langle\pi\rangle B_1 <: \langle\pi\rangle A'$.

Finally, as $A <: \langle\pi\rangle B_1$ and $\langle\pi\rangle B_1 <: \langle\pi\rangle A'$, by transitivity, we have that $A <: \langle\pi\rangle A'$. □

Theorem 4.2.4. (Completeness of the Type System) *If $P \models_\Gamma A$, then $\Gamma \vdash P : A$, where Γ is a well-defined environment.*

Proof. By structural induction on the definition of A.

***Case* Void:** $A = \mathbf{0}$

If $P \models_\Gamma \mathbf{0}$, then $P \equiv \mathbf{0}$. On the other hand, $\Gamma \vdash \mathbf{0} : \mathbf{0}$, by T-Inact. Hence, by Proposition 4.1.8, $\Gamma \vdash P : B$ for some B such that $B \equiv^t \mathbf{0}$. Again by Proposition 4.1.8, as $P \equiv P$ and $\mathbf{0} \equiv^t B$ we have that $\Gamma \vdash P : \mathbf{0}$.

***Case* NonVoid:** $A = \bar{\mathbf{0}}$

If $P \models_\Gamma \bar{\mathbf{0}}$, then $P \not\equiv \mathbf{0}$. By theorem 4.1.5, there is a type A and an environment Γ' such that $\Gamma' \vdash P : A$. By Proposition 3.3.12, $A <: \bar{\mathbf{0}}$; hence, by subsumption, $\Gamma' \vdash P : \bar{\mathbf{0}}$.

***Case* Variable:** $A = \mathcal{Z}$

If $P \models_\Gamma \mathcal{Z}$, then $P = X$ and Γ is $\Gamma', X : \mathcal{Z}$, for some X and Γ'. By T-Var, $\Gamma', X : \mathcal{Z} \vdash X : \mathcal{Z}$; that is $\Gamma \vdash X : \mathcal{Z}$.

***Case* Prefix:** $A = \pi.A'$

If $P \models_\Gamma \pi.A'$, then there is P' such that $P \xrightarrow{\pi} P'$, $P' \models_\Gamma A'$ and $\forall\pi', P'' : (P \xrightarrow{\pi'} P'' \Longrightarrow \pi' = \pi \wedge P'' \approx P')$. In particular, $P \models_\Gamma \langle\pi\rangle A'$. By completeness of the diamond case, we have that $\Gamma \vdash P : \langle\pi\rangle A'$. Finally, by Proposition 3.3.11, $\langle\pi\rangle A' <: \pi.A'$; hence, by the subsumption rule, $\Gamma \vdash P : \pi.A'$.

***Case* Parallel:** $A = A_1 \mid A_2$

If $P \models_\Gamma A_1 \mid A_2$, then exists P_1 and P_2 such that $P \equiv P_1 \mid P_2$, $P_1 \models_\Gamma A_1$ and $P_2 \models_\Gamma A_2$. By induction hypothesis, $\Gamma \vdash P_1 : A_1$ and $\Gamma \vdash P_2 : A_2$. Then, by T-Par, $\Gamma \vdash P_1 \mid P_2 : A_1 \mid A_2$. By Proposition 4.1.8, as $P \equiv P_1 \mid P_2$ there is B such that $\Gamma \vdash P : B$ and $B \equiv^t A_1 \mid A_2$. Again by Proposition 4.1.8, as $P \equiv P$ and $A_1 \mid A_2 \equiv^t B$, $\Gamma \vdash P : A_1 \mid A_2$.

***Case* Restriction:** $A = a\circledR A'$

If $P \models_\Gamma a\circledR A'$, then exists P' such that $P \equiv (\nu a)P'$ and $P' \models_\Gamma A'$. By induction hypothesis, $\Gamma \vdash P' : A'$. Then, by T-New, $\Gamma \vdash (\nu a)P' : a\circledR A'$. By Proposition 4.1.8, as $P \equiv (\nu a)P'$ there is B such that $\Gamma \vdash P : B$ and $B \equiv^t a\circledR A'$. Again by Proposition 4.1.8, as $P \equiv P$ and $a\circledR A' \equiv^t B$, $\Gamma \vdash P : a\circledR A'$.

***Case* Recursion:** $A = (\mathbf{rec}\, Z.A')$

If $P \models_\Gamma (\mathbf{rec}\, Z.A')$, then, by definition, there exist X and P' such that $P \equiv (\mathbf{rec}\, X.P')$ and $P' \models_{\Gamma,\{X:Z\}} A'$. By induction hypothesis, we have that $\Gamma, X : Z \vdash P' : A'$. Therefore, by T-Rec, $\Gamma \vdash (\mathbf{rec}\, X.P') : (\mathbf{rec}\, Z.A')$.

Now, as $P \equiv (\mathbf{rec}\, X.P')$, we have, by Proposition 4.1.8, that $\Gamma \vdash P : B$, for some $B \equiv^t (\mathbf{rec}\, Z.A')$. Again by Proposition 4.1.8, as $P \equiv P$, and $(\mathbf{rec}\, Z.A') \equiv^t B$, we have that $\Gamma \vdash P : (\mathbf{rec}\, Z.A')$.

***Case* Diamond:** $A = \langle\pi\rangle\, A'$

If $P \models_\Gamma \langle\pi\rangle\, A'$, then, by definition, there is a P' such that $P \xrightarrow{\pi} P'$ and $P' \models_\Gamma A'$. By induction hypothesis, we have that $\Gamma \vdash P' : A'$. Therefore, by lemma 4.2.3, $\Gamma \vdash P : B$ and $B <: \langle\pi\rangle\, A'$, for some B; hence, by Subsumption, $\Gamma \vdash P : \langle\pi\rangle\, A'$.

***Case* Box:** $A = [\pi]\, A'$

If $P \models_\Gamma [\pi]\, A'$, then, by definition, either $P \models_\Gamma \pi.A'$ or $P \not\xrightarrow{\pi}$.

First, suppose that $P \models_\Gamma \pi.A'$. Then, by completeness, $\Gamma \vdash P : \pi.A'$. By Proposition 3.3.11, $\pi.A' <: [\pi]\, A'$; hence, by the Subsumption rule, $\Gamma \vdash P : [\pi]\, A'$.

Secondly, suppose that $P \not\xrightarrow{\pi}$. In the first case, let B be the structural type of P, obtained by lemma 4.1.1. By Proposition 3.3.11, $B <: [\pi]\, A$. Then, by the Subsumption rule, $\Gamma \vdash P : [\pi]\, A$.

***Case* Conjunction:** $A = \bigwedge_{i\in I} A_i$

If $P \models_\Gamma \bigwedge_{i\in I} A_i$, then, by definition, $P \models_\Gamma A_i$ for all $i \in I$. Then, by induction hypothesis, $\Gamma \vdash P : A_i$ for all $i \in I$, respectively. Finally, by the Conjunction rule, $\Gamma \vdash P : \bigwedge_{i\in I} A_i$.

***Case* Disjunction:** $A = \bigvee_{i\in I} A_i$

If $P \models_\Gamma \bigvee_{i\in I} A_i$, then, by definition, $P \models_\Gamma A_i$ for some $i \in I$. Thus, there exists $j \in I$ such that $P \models_\Gamma A_j$, then, by induction hypothesis, $\Gamma \vdash P : A_j$. By Proposition 3.3.11, $A_j <: \bigvee_{i\in I} A_i$, as $j \in I$; subsequently, by the subsumption rule, $\Gamma \vdash P : \bigvee_{i\in I} A_i$. □

Corollary 4.2.5. (Completeness) *If $P \models A$, then $\vdash P : A$.*

Proof. Direct from 4.2.4, with $\Gamma = \emptyset$. □

4.3 Further results

In this section we prove proposition 3.3.9 using results of previous sections.

First, the following three auxiliary lemmas are needed.

Lemma 4.3.1. *If $\Gamma \vdash P : A$ and a is fresh in A, then a is fresh in P.*

Proof. Similar to 4.1.7. □

Lemma 4.3.2. *If $P \models_\Gamma A$ and a is fresh in A, then a is fresh in P.*

Proof. By completeness, if $P \models_\Gamma A$, then $\Gamma \vdash P : A$. Then as a is fresh in A, by lemma 4.3.1, a is fresh in P. □

Proof of proposition 3.3.9

***Case* 1:** $A \equiv_\alpha B$

We have to consider two subcases: *Alpha Res* and *Alpha Rec* (in both directions).

***Subcase* 1.1.a):** $a®A' \equiv^t_\alpha b®(A'\{b/a\})$, where b is fresh in A'

Let $P \models_\Gamma a®A'$. Then there exists P' such that $P \equiv (\nu a)P'$ and $P' \models_\Gamma A'$. By lemma 4.3.2, as b is fresh in A, we have that b is fresh in P and so, b is also fresh in P'. Then, there exists Q such that $P'\{b/a\} <: Q$ and $Q \models_\Gamma A'\{b/a\}$. Therefore, $(\nu b)Q \models_\Gamma b®(A'\{b/a\})$.

Moreover, as $P'\{b/a\} <: Q$, we have that $(\nu b)(P'\{b/a\}) <: (\nu b)Q$. On the other hand, as b is fresh in P', we have that $(\nu a)P' \equiv_\alpha (\nu b)(P'\{b/a\})$ and subsequently, $(\nu a)P' \equiv (\nu b)(P'\{b/a\})$. Then, by proposition 3.2.12, $P <: (\nu a)P'$ and $(\nu a)P' <: (\nu b)(P'\{b/a\})$. Hence, by transitivity of $<:$, we finally obtain $P <: (\nu b)Q$. Therefore, $a®A' <: b®(A'\{b/a\})$.

***Subcase* 1.1.b):** $b®(A'\{b/a\}) \equiv_\alpha a®A'$, where b is fresh in A'

Similar to 1.1.a).

***Subcase* 1.2.a):** $(\mathbf{rec}\mathcal{Z}.A') \equiv_\alpha (\mathbf{rec}\mathcal{W}.A'\{\mathcal{W}/X\})$, where $\mathcal{W}$ is fresh in A'

Let $P \models_\Gamma (\mathbf{rec}\mathcal{Z}.A')$. Then there exists a process variable X and a process P' such that $P \equiv (\mathbf{rec}X.P')$ and $P' \models_{\Gamma, X:\mathcal{Z}} A'$. By completeness, $\Gamma, X : \mathcal{Z} \vdash P' : A'$. Let Y be a fresh variable in P'. Using proposition 4.1.2, as $\mathcal{W}$ is fresh in A', we have that $\Gamma, X : \mathcal{Z}, Y : \mathcal{W} \vdash P' : A'$. On the other hand, we have that $\Gamma, Y : \mathcal{W} \vdash Y : \mathcal{W}$, by rule T-Var. Therefore, $\Gamma, Y : \mathcal{W} \vdash P'\{Y/X\} : A'\{\mathcal{W}/\mathcal{Z}\}$ by proposition 4.1.6. Now, using rule T-Rec, we obtain $\Gamma \vdash (\mathbf{rec}Y.P'\{Y/X\}) : (\mathbf{rec}\mathcal{W}.A'\{\mathcal{W}/\mathcal{Z}\})$.

Moreover, $(\mathbf{rec}Y.P'\{Y/X\}) \equiv_\alpha (\mathbf{rec}X.P')$ because Y is chosen fresh in P'. Therefore, we have that $(\mathbf{rec}Y.P'\{Y/X\}) \equiv (\mathbf{rec}X.P')$, and so, by proposition 3.2.12, $(\mathbf{rec}X.P') <: (\mathbf{rec}Y.P'\{Y/X\})$. In the same way, as $P \equiv (\mathbf{rec}X.P')$ we have that $P <: (\mathbf{rec}X.P')$. Finally, by transitivity of $<:$,

$P <: (\mathbf{rec}Y.P'\{Y/X\})$; hence, $(\mathbf{rec}\mathcal{Z}.A') <: (\mathbf{rec}\mathcal{W}.A'\{\mathcal{W}/X\})$.

***Subcase* 1.2.b):** $(\mathbf{rec}\mathcal{W}.A'\{\mathcal{W}/X\}) \equiv_\alpha (\mathbf{rec}\mathcal{Z}.A')$, where $\mathcal{W}$ is fresh in A'

Similar to 1.2.a).

***Cases* 2 - 4:**

Done in lemma 3.3.7.

***Case* 5.a):** $a\circledR\mathbf{0} \equiv^t \mathbf{0}$

Let $P \models_\Gamma a\circledR\mathbf{0}$. Then there exists a process P' such that $P \equiv (\nu a)P'$ and $P' \models_\Gamma \mathbf{0}$. Subsequently, we have that $P' \overset{\pi}{\not\rightarrow}$ for every π. Then also $(\nu a)P' \overset{\pi}{\not\rightarrow}$ for every π. Therefore, $(\nu a)P' \models_\Gamma \mathbf{0}$. On the other hand, by proposition 3.2.12, $P <: (\nu a)P'$. Hence $a\circledR\mathbf{0} \models_\Gamma \mathbf{0}$.

***Case* 5.b):** $\mathbf{0} \equiv^t a\circledR\mathbf{0}$

Let $P \models_\Gamma \mathbf{0}$. Then $P \overset{\pi}{\not\rightarrow}$ for every π. Therefore, it is not hard to notice that $P <: (\nu a)P$. On the other hand $(\nu a)P \models_\Gamma a\circledR\mathbf{0}$. Hence $\mathbf{0} <: a\circledR\mathbf{0}$.

***Case* 6.a):** $a\circledR b\circledR A' \equiv^t b\circledR a\circledR A'$

Let $P \models_\Gamma a\circledR b\circledR A'$. Then there exists a process P' such that $P \equiv (\nu a)P'$ and $P' \models_\Gamma b\circledR A'$. In the same way, there exists a process P'' such that $P' \equiv (\nu b)P''$ and $P'' \models_\Gamma A'$. Then, $(\nu a)P'' \models_\Gamma a\circledR A'$ and, subsequently, $(\nu b)(\nu a)P'' \models_\Gamma b\circledR a\circledR A'$. On the other hand, as $P \equiv (\nu a)P'$ and $P' \equiv (\nu b)P''$, we have that $P \equiv (\nu a)(\nu b)P''$. But by definition 2.2.6, $(\nu a)(\nu b)P'' \equiv (\nu b)(\nu a)P''$ so $P \equiv (\nu b)(\nu a)P''$ and, by proposition 3.2.12, $P <: (\nu b)(\nu a)P''$. Hence $a\circledR b\circledR A' <: b\circledR a\circledR A'$.

***Case* 6.b):** $b\circledR a\circledR A \equiv^t a\circledR b\circledR A$

Similar to 6.a).

***Case* 7.a):** $a\circledR(A_1 \mid A_2) \equiv^t (a\circledR A_1) \mid A_2$, where $a \notin \text{fn}(A_2)$

Let $P \models_\Gamma a\circledR(A_1 \mid A_2)$. Then there exists a process P' such that $P \equiv (\nu a)P'$ and $P' \models_\Gamma A_1 \mid A_2$. Subsequently, there exists processes P_1 and P_2 such that $P' \equiv P_1 \mid P_2$, $P_1 \models_\Gamma A_1$ and $P_2 \models_\Gamma A_2$. Hence, $P \equiv (\nu a)(P_1 \mid P_2)$. On the other hand, $((\nu a)P_1) \mid P_2 \models_\Gamma (a\circledR A_1) \mid A_2$. It remains to be shown that $P <: ((\nu a)P_1) \mid P_2$. For this purpose, we first note that $P <: (\nu a)(P_1 \mid P_2)$ by proposition 3.2.12. So if we show that $(\nu a)(P_1 \mid P_2) <: ((\nu a)P_1) \mid P_2$ we may conclude that $P <: ((\nu a)P_1) \mid P_2$ by transitivity of $<:$. Thus,

***Case* 7.b):** $(a\circledR A_1) \mid A_2 \equiv^t a\circledR(A_1 \mid A_2)$, where $a \notin \text{fn}(A_2)$

Similar to 7.b).

Chapter 5

An Application

This chapter resumes and further discusses the application introduced in chapter 1, consisting of a derivation of a spatial invariant about a process. The process is *Looper*, and the application is described next.

First, consider the *Looper* process

$$Looper \stackrel{\text{def}}{=} (\nu a)(a \mid (\mathbf{rec} X.\bar{a}.(a \mid a \mid X))).$$

The goal is to prove that

$$\Gamma \vdash Looper : \Box(\bar{\mathbf{0}} \mid \bar{\mathbf{0}}).$$

This states that *Looper* will always have two separate threads running in parallel.

Consider the following two abbreviations

$LoopType \stackrel{\text{abv}}{=} (\mathbf{rec} \mathcal{Z}.\bar{a}.(a \mid a \mid \mathcal{Z}))$, and

$MsgsType \stackrel{\text{abv}}{=} \quad !a \mid a.$

The type system first proves

$$\vdash Looper : a®(MsgsType \mid LoopType) \tag{5.1}$$

using structural rules in all steps except one. Table 5.1 gives a derivation of *Looper*. Note that the steps of the derivation are all structural except the one on the left-hand side where subsumption is used along with the subtyping rule $a <: !a \mid a$, proved on lemma 3.3.11. The non-structural step could be "pushed forward" to the subtyping part without any impact; it is left there to simplify.

Tables 5.2 and 5.3 exhibit subtyping derivations for

$$a®(!a \mid a \mid (\mathbf{rec} \mathcal{Z}.\bar{a}.(a \mid a \mid \mathcal{Z}))) <: a®(!a \mid (\mathbf{rec} \mathcal{Z}.\bar{a}.(a \mid a \mid \mathcal{Z}))) \tag{5.2}$$

and

$$[\pi]\,(a®(!a \mid (\mathbf{rec} \mathcal{Z}.\bar{a}.(a \mid a \mid \mathcal{Z}))) <: [\pi]\,(a®(!a \mid a \mid (\mathbf{rec} \mathcal{Z}.\bar{a}.(a \mid a \mid \mathcal{Z})))) \tag{5.3}$$

respectively. The derivation of tables 5.2 and 5.3 both use lemmas 3.3.7, 3.3.11 and 3.3.3 in the first steps of the derivation and uses proposition 3.3.8 in the last steps.

Also, from proposition 3.3.11 we know that

$$a\circledR(!a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z))) <: [\pi]\,(a\circledR(!a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z)))), \text{ for every } \pi. \tag{5.4}$$

So, using transitivity of subtyping (lemma 3.3.3) twice, we can conclude from expressions (5.2), (5.4) and (5.3) the following relation

$$a\circledR(!a \mid a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z))) <: [\pi]\,(a\circledR(!a \mid a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z)))), \text{ for every } \pi. \tag{5.5}$$

That is,

$$a\circledR(MsgsType \mid LoopType) <: [\pi]\,(a\circledR(MsgsType \mid LoopType)), \text{ for every } \pi. \tag{5.6}$$

Then, using the last subtyping rule in proposition 3.3.11, we conclude from expression 5.5 that

$$a\circledR(MsgsType \mid LoopType) <: \Box(a\circledR(MsgsType \mid LoopType)). \tag{5.7}$$

On the other hand, table 5.4 exhibits a derivation of

$$\Box(a\circledR(MsgsType \mid LoopType)) <: \Box(\bar{\mathbf{0}} \mid \bar{\mathbf{0}}). \tag{5.8}$$

It uses all rules in proposition 3.3.12, rules from proposition 3.3.3 and 3.3.8. The last step in the derivation of table 5.4 is not proved. This issue will be addressed in the end of this chapter. For now, assume that it is proved.

Summing up, we have that:

1. $\vdash Looper : a\circledR(MsgsType \mid LoopType)$ from (5.1);
2. $a\circledR(MsgsType \mid LoopType) <: \Box(a\circledR(MsgsType \mid LoopType))$ from (5.7);
3. $\Box(a\circledR(MsgsType \mid LoopType)) <: \Box(\bar{\mathbf{0}} \mid \bar{\mathbf{0}})$ from (5.8).

Therefore, using transitivity of the subtyping relation, we obtain from 2. and 3.

$$a\circledR(MsgsType \mid LoopType) <: \Box(\bar{\mathbf{0}} \mid \bar{\mathbf{0}}) \tag{5.9}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \mathbf{0} : \mathbf{0}}{\vdash a : a}\,\text{T-Pre} \quad a <: !a \mid a}{\vdash a : !a \mid a}\,\text{T-Sub} \quad \dfrac{\dfrac{\dfrac{\dfrac{X : Z \vdash \mathbf{0} : \mathbf{0}}{X : Z \vdash a : a}\,\text{T-Pre} \quad \dfrac{X : Z \vdash \mathbf{0} : \mathbf{0}}{X : Z \vdash a : a}\,\text{T-Pre} \quad X : Z \vdash X : Z}{X : Z \vdash a \mid X : a \mid Z}\,\text{T-Par}}{X : Z \vdash a \mid a \mid X : a \mid a \mid Z}\,\text{T-Par} \quad \dfrac{X : Z \vdash \bar{a}.(a \mid a \mid X) : \bar{a}.(a \mid a \mid Z)}{\vdash (\mathbf{rec}X.\bar{a}.(a \mid a \mid X)) : (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z))}\,\text{T-Rec}}{\vdash a \mid (\mathbf{rec}X.\bar{a}.(a \mid a \mid X)) : MsgsType \mid LoopType}\,\text{T-Par}}{\vdash Looper : a\circledR(MsgsType \mid LoopType)}\,\text{T-New}$$

Table 5.1: Structural derivation.

$$\dfrac{\dfrac{\dfrac{!a \mid a <: a \mid !a \qquad a \mid !a <: !a}{!a \mid a <: !a}}{!a \mid a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z)) <: !a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z))}}{a®(!a \mid a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z))) <: a®(!a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z)))}$$

Table 5.2: Subtyping derivation 1.

$$\dfrac{\dfrac{\dfrac{\dfrac{!a <: a \mid !a \qquad a \mid !a <: !a \mid a}{!a <: !a \mid a}}{!a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z)) <: !a \mid a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z))}}{a®(!a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z))) <: a®(!a \mid a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z)))}}{[\pi]\,(a®(!a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z)))) <: [\pi]\,(a®(!a \mid a \mid (\mathbf{rec}Z.\bar{a}.(a \mid a \mid Z))))}$$

Table 5.3: Subtyping derivation 2.

Finally, using the subsumption rule we conclude from (5.9) that $\vdash \mathit{Looper} : \Box(\bar{\mathbf{0}} \mid \bar{\mathbf{0}})$.

The last step

Table 5.4 exhibits a derivation of

$$\Box(a®(MsgsType \mid LoopType)) <: \Box(\bar{\mathbf{0}} \mid \bar{\mathbf{0}}). \tag{5.10}$$

The last step (the dashed line), uses preservation of subtyping by $\Box$. By definition 3.2.14, $\Box A$ is an abbreviation of $\bigwedge_{t \in Act^*} [t]\, A$. Therefore, it actually uses preservation of subtyping by conjunction, which was not proved thus far.

Hence, the derivation of the example remains an open problem.

We end this chapter with a few remarks concerning this issue. First, consider the following attempt to prove that binary (to simplify) conjunction preserves subtyping. Assume that $A <: B$ and suppose that $P \models_\Gamma A \wedge C$. Then, $P \models_\Gamma A$ and $P \models_\Gamma C$. As $A <: B$, there is a process Q such that $P <: Q$ and $Q \models_\Gamma B$. But, is it true that $Q \models_\Gamma C$? If the answer is yes, then $A \mid C <: B \mid C$ and the derivation is valid. If not, then it remains open.

A careful observation of the last step in table 5.4 suggests that such a strong result is actually not

$$\dfrac{\dfrac{\dfrac{\dfrac{MsgsType <: \bar{\mathbf{0}} \qquad LoopType <: \bar{\mathbf{0}}}{MsgsType \mid LoopType <: \bar{\mathbf{0}} \mid \bar{\mathbf{0}}}}{a®(MsgsType \mid LoopType) <: a®(\bar{\mathbf{0}} \mid \bar{\mathbf{0}})} \qquad a®(\bar{\mathbf{0}} \mid \bar{\mathbf{0}}) <: \bar{\mathbf{0}} \mid \bar{\mathbf{0}}}{a®(MsgsType \mid LoopType) <: \bar{\mathbf{0}} \mid \bar{\mathbf{0}}}}{\Box(a®(MsgsType \mid LoopType)) <: \Box(\bar{\mathbf{0}} \mid \bar{\mathbf{0}})}$$

Table 5.4: Subtyping derivation 3.

needed. Instead, if we prove that

$$\text{If } A <: B, \text{ then } \bigwedge_{t \in Act^*} [t]\, A <: \bigwedge_{t \in Act^*} [t]\, B, \tag{5.11}$$

then the derivation is still valid.

By proposition 3.3.8,

$$\text{If } A <: B, \text{ then } [\pi]\, A <: [\pi]\, B, \text{ for every } \pi. \tag{5.12}$$

Consider an arbitrary sequence of actions $t = \pi_n \dots \pi_1$. Therefore, by a simple induction over n,

$$\text{If } A <: B, \text{ then } [\pi_1] \dots [\pi_n]\, A <: [\pi_1] \dots [\pi_n]\, B. \tag{5.13}$$

That is,

$$\text{If } A <: B, \text{ then } [t]\, A <: [t]\, B, \tag{5.14}$$

and as t is arbitrary,

$$\text{If } A <: B, \text{ then } [t]\, A <: [t]\, B, \text{ for every } t. \tag{5.15}$$

Hence, remains to be shown the following assertion

$$\text{If } [t]\, A <: [t]\, B, \text{ for every } t, \text{ then } \bigwedge_{t \in Act^*} [t]\, A <: \bigwedge_{t \in Act^*} [t]\, B. \tag{5.16}$$

In conclusion, the validity of the derivation of the spatial invariant depends either of subtyping being preserved by conjunction or of the validity of assertion (5.16).

Chapter 6

Conclusions

This last chapter summarizes the work developed and points out future possible developments.

6.1 Achievements

Right from the beginning it was clear that a logic to express spatial properties of processes would be useful. Achieving the right combination of operators and semantics and defining a suitable compatible subtyping relation proved to be an extensive and complex process. Also, being able to derive the motivation example was a constant requirement which increased the amount of obstacles to deal with. Nevertheless, we were able to define a logic based on:

- a syntax capable of expressing both spatial and behavioural properties of processes;
- a semantics defined with respect to two different relations over processes;
- a deductive system based on types, having its power focused on a subtyping relation.

In addition, important properties of the logic were proved. This ensures its usefulness. In the course of this process, other results came to light helping to further characterize the logic. Namely, it was proved:

- that every process has a type;
- weak consistency and completeness results;
- the soundness of all subtyping rules with respect to the definition of subtyping.

6.2 Future work

During the research process it was necessary to make some decisions which impact was not totally visible at the time. Looking back, it seems as though other alternatives were equally possible, however leading to necessarily different results. Some of them are worthwhile further investigating. This work is left to be done in the future.

Some of these issues are addressed in the following.

Treatment of variables and semantics: In sections 3.3.1 and 3.2.2, the satisfaction relation uses environments to map process variables into type variables. In fact, it demands a one-to-one correspondence. This decision was to simplify the treatment of recursion, but it is rather abusive. Instead, it would be more reasonable to consider each type variable to be satisfied by a set of process variables, *i.e.*, defining *valuation*. Moreover, an alternative definition of semantics should be considered. For instance, it would be more natural to define *satisfaction* through definitions of *valuation* and *denotation*.

Interpretation of prefix: In definition 3.2.13, the interpretation of the prefix operator requires that every π-derivative of a process satisfying $\pi.A$, must be bisimilar. This condition is rather demanding. Instead, one could require that any π-derivative of a process satisfying $\pi.A$ must satisfy A.

Recursive types: Recursive types are still an open field of discussion nowadays, due to their complexity. Therefore, this is an aspect that, most certainly, will be worthwhile discussing in a future approach. Nevertheless, in definition 3.2.13, recursive types are interpreted using structural congruence. This choice was made to simplify the work. Instead, type recursion could have been defined as a fixed point of an operator as is done in [CC03].

Subtyping rules: The rules listed throughout section 3.3.2 suffice to ensure completeness and to derive the motivating example. But, most probably, there are rules which can be derived from others and therefore should be dropped. On the other hand, most probably, there are rules yet to be devise which generalize some of the rules considered, for instance the twelfth rule of proposition 3.3.11. In other words, the set of rules presented is not the minimal one.

Conjunction rule: The rule adopted for conjunction, was deliberately considered in the infinitary case to achieve completeness of the modality $\Box A$. Instead, giving up completeness, one could consider two separate rules: a global binary rule for conjunction and a local infinitary rule for the conjunction $[t]\, A$, where t is a sequence of actions.

Bibliography

[Cai03] Luís Caires. Behavioral and Spatial Properties in a Logic for the π-Calculus. In I. Walukiwicz, editor, *Proc. of Foundations of Software Science and Computation Structures'04*. Lecture Notes in Computer Science, Springer Verlag, 2003.

[CC03] L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part I). *Journal of Information and Computation*, 186(2):194–235, 2003.

[CC04] L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part II). *Journal of Theoretical Computer Science (Germany)*, 2004.

[CV06] L. Caires and H. T. Vieira. Extensionality of Spatial Observations in Distributed Systems. 2006.

[GG03] L.Cardelli, P. Gardner and G. Ghelli. Manipulating trees with hidden labels. In A. D. Gordon, editor, *Proc. of Foundations of Software Science and Computation Structures (FoSSaCS'03)*. Lecture Notes in Computer Science, Springer-Verlag, 2003.

[Han94] Chris Hankin. *Lambda calculi: a guide for computer scientists*. Oxford: Clarendon Press, 1994.

[Hoa85] C. A. R. Hoare. *Communication and Sequential Processes*. Prentice Hall, 1985.

[IK04] A. Igarashi and N. Kobayashi. A Generic Type System for the π-Calculus. *Theoretical Computer Science*, 1–3(311):121–163, 2004.

[IO01] S. Ishtiaq and P. O'Hearn. BI as an Assertion Language for Mutable Data Structures. In *28th ACM Symp. on Principles of Programming Languages*, 2001.

[LG00] L.Cardelli and A. D. Gordon. Anytime, Anywhere. Modal Logics for Mobile Ambients. In *27th ACM Symp. on Principles of Programming Languages*, pages 365–377. ACM, 2000.

[Mil89] Robin Milner. *Communication and concurrency*. New York : Prentice Hall, 1989.

[Mil99] Robin Milner. *Communicating and mobile systems : the π-calculus*. Cambridge University Press, c1999.

[OP99] P. O'Hearn and D. Pym. The Logic of Bunched Implications. *The Bulletin of Symbolic Logic*, 5(2):215–243, 1999.

[Pie02] Benjamin Pierce. *Types and Programming languages*. The MIT Press, 2002.

[PW92] R. Milner, J. Parrow and D. Walker. A Calculus of Mobile Processes, parts I and II. *Information and Computation*, 100(1):1–77, 1992.

[RR01] S. K. Rajamani and J. Rehof. A Behavioral Module System for the π-Calculus. In *SAS'01: 8th International Static Analysis Symposium (Paris, France), LNCS*. Springer, 2001.

[RV00] A. Ravara and V. Vasconcelos. Typing Non-Uniform Concurrent Objects. In C. Palamidessi, editor, *CONCURR'00*, volume 1877 of *Lecture Notes in Computer Science*, pages 474–488. Springer-Verlag, 2000.

[SW01] D. Sangiorgi and D. Walker. *The π-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.