CARL VON OSSIETZKY universität OLDENBURG
FAKULTÄT II
DEPARTMENT FÜR INFORMATIK

Model checking π-Calculus against temporal connectedness properties

Sven Linker

December 6, 2008

Prof. Dr. E.-R. Olderog
Dipl.-Inform. R. Meyer

Contents

1 Introduction

In the last decades, computers have changed from big, single systems computing one job at a time to complex networks of small, independent components, each designed only to solve specific problems. Such a network accomplishes its tasks by decomposing a given problem into smaller ones, which are then resolved by its components. Consider for example a modern operating system like Windows Vista, Linux or a variant of BSD. Instead of one monolithic program dealing with every possible task the user wants to perform, these systems start a variety of so-called processes (or daemons) to fulfill these tasks. One process answers all requests to modify the file system, e.g. create or delete files, while another process sends data to the graphics adapter to draw the graphical user interface.

Probably the most popular example for such a network is the Internet. A huge amount of servers waits for requests by clients or other servers, e.g. to deliver HTML-pages or to give access to data stored in a database. In between, routers have to forward the messages sent for example from a client to a server, since the number of connected systems is far too big to allow fully physical linking.

This network-based view can also be applied in a more abstract way to other fields. Mobile phones establish connections with radio stations, which transmit the spoken words and other messages from one participant to the other. Therefore the phones and radio stations can be seen as processes communicating with each other by radio signals.

But this increase of communication and concurrent operations arises the question how the functionality of these networks can be assured. A traditional and well-understood approach to mathematically describe such concurrent systems are *Petri nets* [Pet62], which explicitly define every possible communication between processes. Even though Petri nets are a very intuitive formalism to model concurrent systems, their size grows dramatically with the number of considered processes. Hence the manual construction of Petri nets is very error-prone. But then a lot of research concerning the mathematical analysis of Petri nets has been done and effective algorithms to check properties of Petri nets have been developed.

Other ways to specify concurrent systems are *process algebras* like CSP (Communicating Sequential Processes) [Hoa78] and CCS (Calculus of Communicating Systems) [Mil80]. These formalisms describe processes as algebraic structures.

They normally contain operators resembling the parallel execution of programs and possible choice of alternative control flows. A system modeled by a process algebra is in general more readable than the same system specified as a Petri net, since a great part of the complexity is hidden by the operators of the algebras. A major drawback of process algebras is their general Turing-completeness [Vaa93] in contrast to standard Petri nets.

Another problem with the process algebras mentioned above is that possible pairs of communicating components have to be explicitly defined and can not change during the computation. This problem is taken account of by the π-Calculus [MPW92, Mil99, SW01]. It is designed to model *mobile* communicating and concurrent systems. Mobility in the sense of the π-Calculus has to be understood as mobility in the *linking structure* of processes. Consider the example of mobile phones given above. If a person carrying a mobile phone leaves the cell of the nearest radio station S_1, it establishes a new connection to another station S_2. The link to S_1 is not usable anymore, in a way it is “forgotten”, but the link to S_2 has been created, hence the linking structure of the whole network has changed. In the π-Calculus, this is achieved by the passing of *names*, which also represent the possible links. Hence processes can exchange links between each other. Since the π-Calculus is an extension to CCS, it is Turing-complete.

The identification of subsets of the π-Calculus, for which properties as safeness, liveness or the non-existence of deadlocks can be algorithmically decided, is a field of active research. Various decidable subsets have been found (e.g. the class of Finite Control Processes [Dam96]). A common approach to decide properties of processes is the technique of *model checking* [EC80]. To model check a process against a property, both have to be translated into an automata-theoretic equivalent. The concrete algorithm to check if the process satisfies the property depends heavily on the automata-theory. Since decidability of Petri nets is well-researched, many translations of process algebras to Petri nets have been developed to make use of the effective algorithms on Petri nets.

For specifying properties of concurrent systems, *temporal logics* have a long tradition in the model-checking community. These logics are special modal logics [CZ97], where the modalities are interpreted with respect to a notion of time. In computer science, normally only temporal logics dealing with the future evolution of systems are considered, even though temporal logics would allow also reasoning about the past. Prominent examples for temporal logics used for model checking are Linear Temporal Logic (LTL) [Pnu77] and the Temporal Logic of Actions (TLA) [Lam94].

Recently, Meyer has developed a translation of a large subset of π-Calculus-processes to standard Petri nets, called the *structural semantics* of processes [Mey07,

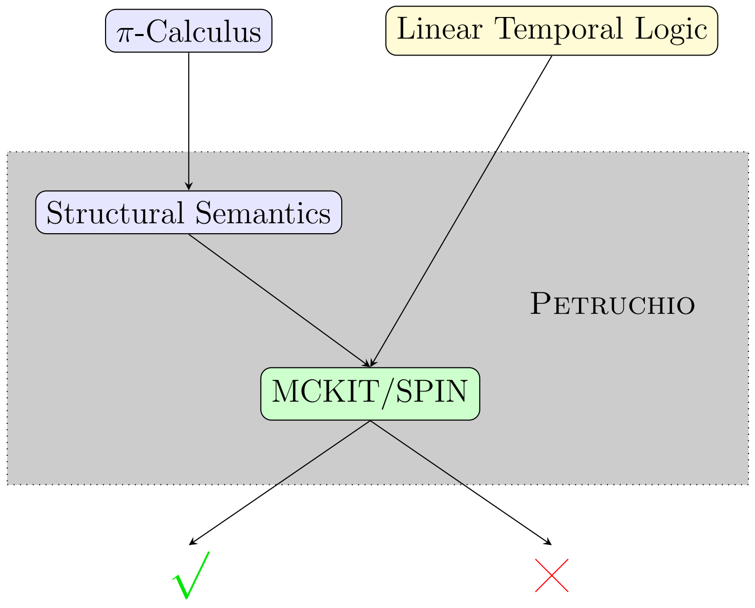

Figure 1.1: Model checking π-Calculus-processes with Petruchio.

Mey08]. This approach has been implemented in the tool Petruchio[1] by Strazny [Str07]. The tool uses the Model Checking Kit (MCKIT) [SSE03] and the model checker SPIN [Hol97] as a back end. Properties the process shall be checked against can be defined in LTL. Figure 1.1 shows the current control flow of the verification. The user specifies a process and a temporal formula on Petri nets. This process is then translated into its structural semantics and checked against the formula. A major drawback is that the user has to know the structure of the Petri net to describe the property the process shall be checked against. Hence the process has to be translated and the user can not specify the formula until then. That is, the work flow is split into the specification of the process, the inspection of the resulting Petri net and the formulation of the property.

The contribution of this thesis is the definition of a logic on π-Calculus-processes, which can be translated into LTL on Petri nets. With the help of this logic, the work flow of model checking with Petruchio can be improved as shown in Figure 1.2.

Even though many logics for describing properties of π-Calculus-processes already exist, most of them are defined with respect to the observational behaviour of the processes [HM85, Dam89, MPW91] and do not take the structural config-

[1]Available at `http://petruchio.informatik.uni-oldenburg.de`

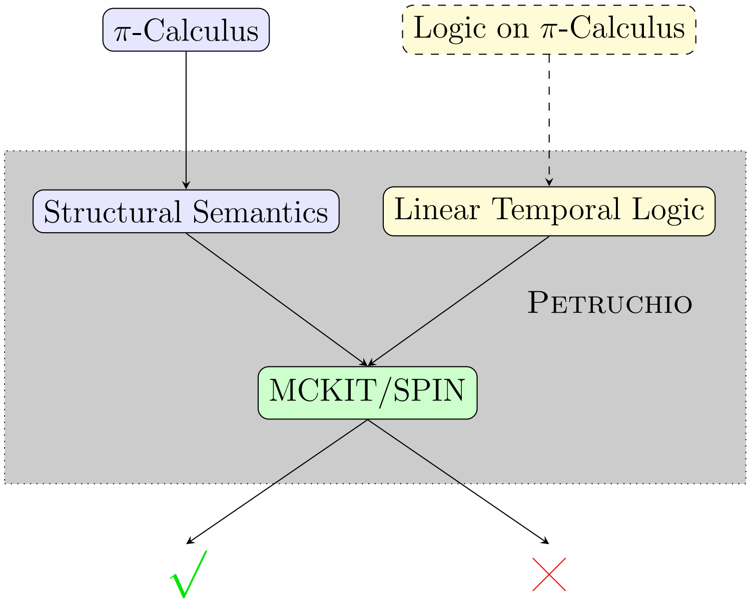

Figure 1.2: Model checking π-Calculus-processes with Petruchio with a logic on π-Calculus-processes.

uration of processes into account. Logics respecting the structure of processes, often called *spatial logics*, have been recently developed for the π-Calculus [CC01, CC02, MP08] and the Ambient Calculus [CG00, CG01, CG06], an extension of the π-Calculus. However, these logics are way to expressive for the desired translation into LTL. For example, one problem is that these logics allow the mixed use of spatial and temporal modalities in the formula.

The document is structured as follows. In Chapter 2 we will give the definitions of the Petri nets used in this thesis, the π-Calculus, the structural semantics and linear temporal logic. We proceed in Chapter 3 by the presentation of our logic together with some nice and useful properties. Chapter 4 introduces new atoms for simplifying the formulas defined in the previous chapter. Subsequently, we present our translation in Chapter 5. Finally, we summarise our results together with a discussion of the related work in Chapter 6.

2 Preliminaries

First we define some notations and recall the basics of the π-Calculus and Petri nets. We then briefly present the translation of π-Calculus processes to Petri nets developed by Meyer [Mey08].

2.1 Notations

Definition 2.1.1 (Equivalence Relation)**.** Let A be an arbitrary set and $\sim \subseteq A\times A$ be a *relation* on A. We will write that $\sim$ relates two elements a and b, i.e. $(a, b) \in\sim$ as $a \sim b$. This relation is an *equivalence relation*, if and only if it satisfies reflexivity, symmetry and transitivity. Formally

$$\begin{array}{llccc} \text{Reflexivity:} & \forall a \in A\colon & a \sim a & & \\ \text{Symmetry:} & \forall a, b \in A\colon & a \sim b & \text{implies} & b \sim a \\ \text{Transitivity:} & \forall a, b, c \in A\colon & a \sim b \text{ and } b \sim c & \text{implies} & a \sim c \end{array}$$

We use the standard notation for functions and lift the notation to work on sets.

Definition 2.1.2. Let A, B be two sets and $f \subseteq A \times B$. We say that f is a *(total) function from A to B*, denoted by $f\colon A \to B$, if and only if f is left-total and functional, i.e.

$$\begin{array}{llccc} \text{Left-Totality:} & \forall a \in A\colon \exists b \in B\colon & (a, b) \in f & & \\ \text{Functionality:} & \forall a \in A\colon \forall b, c \in B\colon & (a, b) \in f \text{ and } (a, c) \in f & \text{implies} & b = c \end{array}$$

If f is not left-total, we call f a *partial function*. We use the notation $f(a) = b$ for $(a, b) \in f$ as usual. Furthermore, if C is a subset of A, we will write $f(C)$ for the set $\{b \mid \exists c \in C\colon f(c) = b\}$. We call the set

$$dom(f) = \{a \mid a \in A \wedge \exists b \in B\colon f(a) = b\}$$

the *domain* of f. If f is a total function, $dom(f) = A$. Likewise, the set $im(f) = \{b \mid \exists a \in A\colon f(a) = b\}$ is the *image* of f.

Furthermore, we will need the following standard definitions for properties of functions. In particular, bijectivity and structure-preservence are of importance.

Definition 2.1.3 (Properties of Functions)**.** Let A and B two sets and f be a total function from a A to B, i.e.,$f\colon A \to B$. This function can have certain properties.

$$\begin{array}{llll} \text{Injectivity:} & \forall a, c \in A\colon \forall b \in B\colon & f(a) = b \text{ and } f(c) = b & \text{implies} \quad a = c \\ \text{Surjectivity:} & \forall b \in B\colon \exists a \in A\colon & f(a) = b & \end{array}$$

We call f *bijective*, if f satisfies both injectivity and surjectivity. For a bijective function, the *inverse function* of f, denoted by f^{-1} is a function from B to A. It is defined by $f^{-1}(y) = x$, if $f(x) = y$.

Furthermore, let A and B be sets, and $\sim, \simeq$ be relations on A, resp. B. We say that f is a *homomorphism*, if for any two elements $a, a' \in A$, $a \sim a'$ implies $f(a) \simeq f(a')$. Finally, if f is bijective and a homomorphism, and the inverse function f^{-1} is a homomorphism, f is an *isomorphism* between A and B.

2.2 Petri nets

Petri nets are an automata-theoretic approach to model concurrent behaviour of possibly distributed systems developed by Petri [Pet62]. A Petri net describes the possible communications of components modeled by tokens on places explicitly by transitions that consume and produce these tokens. We present the standard notation of place/transition Petri nets with weighted arcs [Rei85].

Definition 2.2.1 (Place/Transition Petri Net)**.** An *unmarked place/transition Petri net* is a triple (S, T, λ) where S is a set of places, T a set of transitions, such that S and T are disjoint, and $\lambda\colon S \times T \cup T \times S \to \mathbb{N}$ is a weight function, associating to every pair (s, t) and (t, s) with $s \in S$, $t \in T$ a number of arcs. For each transition $t \in T$, we refer to the set of all places with $\lambda(s, t) > 0$ by ${}^\bullet t$, the *preset* of t. Similarly, we define the *postset* of t by $t^\bullet = \{s \in S \mid \lambda(t, s) > 0\}$.

The sets S and T can both be infinite. The graphical representation of a Petri net is a directed graph $G = (S \cup T, E)$ with $E = \{(s, t) \mid \lambda(s, t) > 0\} \cup \{(t, s) \mid \lambda(t, s) > 0\}$. A node v of G is represented by a circle if $v \in S$ resp. a square if $v \in T$. The edges are defined by the weight function, i.e., there are only edges from places to transitions and vice versa, but never from a place to another place and never from a transition to another transition.

An unmarked Petri net has no defined behaviour. Therefore the places get marked with *tokens*, which are denoted by small black circles.

Definition 2.2.2 (Marking and Support)**.** The state of a Petri net is defined by a *marking* $M\colon S \to \mathbb{N}$. If $M(s) = k$ for $s \in S$, we say that S *is marked by k tokens* (or S carries k tokens). The *support* of a marking M is the set of places which carry at least one token, i.e., $supp(M) := \{s \in S \mid M(s) > 0\}$.

An unmarked place/transition Petri net together with an *initial marking* M_0 is called a *marked place/transition Petri net* or simply a Petri net. The set of all Petri nets is $\mathcal{PN}$.

Example 2.2.1. Figure 2.1 shows the marked Petri net with

$$\begin{aligned} S &= \{s_1, s_2, s_3, s_4, s_5\}, \\ T &= \{t_1, t_2, t_3\}, \\ \lambda &= \{(s_1, t_1, 1), (s_2, t_2, 1), (t_1, s_3, 1), (t_2, s_4, 1), (s_3, t_3, 1), (s_4, t_3, 1), \\ &\quad (t_3, s_1, 1), (t_3, s_2, 1), (t_3, s_5, 1)\} \\ M_0 &= (1, 1, 0, 0, 0) \end{aligned}$$

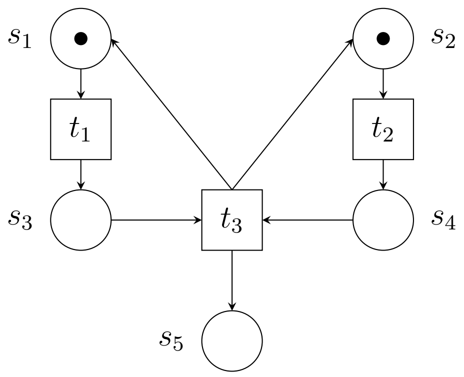

Figure 2.1: Example of a Petri Net. The transitions t_1 and t_2 can fire.

With these markings it is possible to define the behaviour of the Petri net via a transition relation. We therefore need the definition of an enabled transition.

Definition 2.2.3 (Enabled Transition)**.** A transition is *enabled* under the marking M, if $M(s) > \lambda(s, t)$ for all $s \in {}^\bullet t$. We also say that the transition *can fire.*

Now we can define the transition relation and the transition system of a Petri net. Informally, an enabled transition t fires by removing a number of tokens from the places in its preset according to the weight function. The number of tokens a firing transition puts on the places in its postset is again defined by the weight function λ.

Definition 2.2.4 (Transition Relation and Transition System)**.** The behaviour of a Petri net is defined by the *transition relation* $[\rangle : T \times \mathbb{N}^S \to \mathbb{N}^S$, with $M\,[t\rangle\, M'$, if and only if t is enabled under M and $M'(s) = M(s) - \lambda(s,t) + \lambda(t,s)$ for all $s \in S$. In this thesis, we omit the firing transition t when referring to the transition relation. The *transition system* $\mathit{Reach}(\mathcal{N})$ of a marked Petri net $\mathcal{N} = (S, T, \lambda, M_0)$ consists of all markings reachable by the transition relation, i.e., $M_i \in \mathit{Reach}(\mathcal{N})$ if and only if there is a sequence $M_0 M_1 \dots M_i$ such that $M_0\,[\rangle\, M_1\,[\rangle\, \dots\, [\rangle\, M_i$ $(i \in \mathbb{N})$.

Example 2.2.2. Consider the Petri net $\mathcal{N}$ of Example 2.2.1 (Figure 2.1). In the depicted state M_0, the transitions t_1 and t_2 are enabled. If t_1 fires, we get the state $(0,1,1,0,0)$. For t_2 we get $(1,0,0,1,0)$. The state after both transitions have fired is shown in Figure 2.2 (a). In this state, the transition t_3 is enabled and the state after the firing of t_3 is depicted in Figure 2.2 (b). Observe that we have reached the initial state with the exception that s_3 now carries one token. Hence, the transition system of $\mathcal{N}$ is $\mathit{Reach}(\mathcal{N}) = \{(1,1,0,0,c), (0,1,1,0,c), (1,0,0,1,c), (0,0,1,1,c) \mid c \in \mathbb{N}\}$.

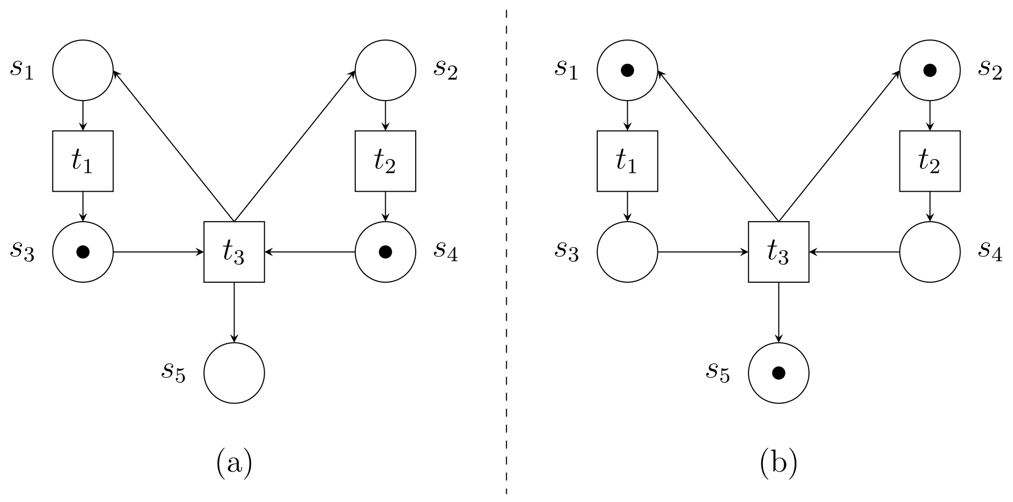

Figure 2.2: Example of a Petri Net. (a) The transitions t_1 and t_2 both have fired. (b) The transition t_3 has fired once.

2.3 The π-Calculus

The π-Calculus [MPW92, Mil99, SW01] is a process algebra designed to model processes communicating on *channels*. Messages sent on a channel consist of *names*, which represent channels. With this mechanism, processes can exchange channels with other processes, thereby creating new possibilities to communicate. In the notation of the π-Calculus, this exchange is called *mobility*, because a process can alter its "location", i.e., its links to other processes. In this thesis, we use a monadic π-Calculus with *recursion* [SW01].

The possibilities of a process to communicate are formalized via so-called *prefixes* π. Given a countably infinite set of names $\mathcal{A}$, the prefixes are defined by

$$\pi ::= \overline{x}\langle y\rangle \mid x(y) \mid \tau,$$

where $x, y \in \mathcal{A}$. The output action prefix $\overline{x}\langle y\rangle$ sends the name y along the channel x, while the input action prefix $x(y)$ receives a name on x that replaces y. An internal, unobservable communication is represented by τ.

We will abbreviate a sequence of names $y_1, \ldots, y_n$ with $\tilde{y}$. The definition of parametrised recursion is based on *process identifiers* denoted by upper-case letters like K or L. Every process identifier has a *defining equation* $K(\tilde{x}) := P_K$ identifying the process P_K with K. A process *call*, denoted by $K\lfloor\tilde{a}\rfloor$, results in a replacement of this term with P_K, where the names $\tilde{x}$ are substituted by $\tilde{a}$. The remaining operators are denoted and defined as usual. The *parallel composition* of two processes, meaning that the processes can communicate via an input and an associated output action, is written $P_1 \mid P_2$. Channels can be hidden by the *restriction* (or *hidden name*) quantifier $\nu a.P$, i.e. a is different from any other name outside the scope of νa. For expressing that a process can react on more than one channel, the *choice* operator $\pi.P + M$ is used. Formally, the syntax of processes is inductively defined.

$$\begin{array}{rcl} M & ::= & \mathbf{0} \mid \pi.P \mid M_1 + M_2 \\ P & ::= & K\lfloor\tilde{a}\rfloor \mid M \mid P_1 \mid P_2 \mid \nu a.P \end{array}$$

We denote the set of all processes according to the definition above by $\mathcal{P}$.

Example 2.3.1 (Client/Server)**.** We model a simple client and server. The client can send its unique IP (νip) to a certain URL, specified by the channel *url* and then waits for a new session object (s) on this IP. Afterwards it receives data linked to this session. The server on the other hand waits for an incoming message on its URL ($url(y)$), which it will interpret as an IP of a client. It creates a new session object and sends it to received the IP ($\nu ses.\overline{y}\langle ses\rangle$). Finally it uses the established session to send data to the client. The data sent in this example is just the session itself.

$$\begin{array}{rcl} C(url) & := & \nu ip.\overline{url}\langle ip\rangle.ip(s).s(x).C\lfloor url\rfloor \\ S(url) & := & url(y).\nu ses.\overline{y}\langle ses\rangle.\overline{ses}\langle ses\rangle.S\lfloor url\rfloor \end{array}$$

Convention 2.3.1. We adopt the following conventions for syntactic abbreviations and operator priorities.

(i) We call a choice composition of the form $M = \mathbf{0} + \cdots + \mathbf{0}$ *empty*. If M contains at least one prefixed process $\pi.P$, we call M *non-empty*.

(ii) We omit the process $\mathbf{0}$ at the end of processes. That is, we write π for $\pi.\mathbf{0}$.

(iii) The natural numbers contain 0, i.e., $\mathbb{N} := \{0, 1, 2, \dots\}$.

(iv) To abbreviate the parallel composition of multiple processes, we write

$$\begin{array}{rcl} \Pi^k P & := & \underbrace{P \mid \dots \mid P}_{k \text{ times}} \\ \Pi_{i=m}^n P_i & := & P_m \mid \dots \mid P_n \\ \Pi_{i\in I} P_i & := & P_{i_0} \mid \dots \mid P_{i_l}, \end{array}$$

where $k, l, m, n \in \mathbb{N}$, $n \geq m$ and $I = \{i_0, \dots, i_l\} \subset \mathbb{N}$ is finite with $i_0 < \dots < i_l$.For $k = 0$ and for $I = \emptyset$, we define $\Pi^0 P := \mathbf{0}$ and $\Pi_{i\in\emptyset} P_i := \mathbf{0}$.

To avoid parentheses in processes, we define the following operator priorities. Prefixing a process with π binds stronger than choice composition $+$. Choice and restriction νa bind stronger than parallel composition $\mid$.

Processes of the form M and $K\lfloor\tilde{a}\rfloor$, where M is a non-empty summation, are called *sequential processes*. The set of all sequential processes is denoted by $\mathcal{P}^{\text{seq}}$.

We further need the notation of *free* and *bound* names. The restriction $\nu y.P$ and the input action $x(y)$ both bind the name y. The intuition of bound and free names is formalised in the following definition.

Definition 2.3.1 (Free and Bound Names)**.** The inductive definitions of the *free names* of a process $P \in \mathcal{P}$, denoted by $fn(P)$ is given in Figure 2.3. The set of *bound names* of a process is defined likewise.

$$
\begin{array}{rclrcl}
\mathit{fn}(\mathbf{0}) & = & \emptyset & \mathit{bn}(\mathbf{0}) & = & \emptyset \\
\mathit{fn}(\tau.P) & = & \mathit{fn}(P) & \mathit{bn}(\tau.P) & = & \mathit{bn}(P) \\
\mathit{fn}(\overline{x}\langle y\rangle.P) & = & \mathit{fn}(P) \cup \{x, y\} & \mathit{bn}(\overline{x}\langle y\rangle.P) & = & \mathit{bn}(P) \\
\mathit{fn}(x(y).P) & = & (\mathit{fn}(P) \setminus \{y\}) \cup \{x\} & \mathit{bn}(x(y).P) & = & \mathit{bn}(P) \cup \{y\} \\
\mathit{fn}(M_1 + M_2) & = & \mathit{fn}(M_1) \cup \mathit{fn}(M_2) & \mathit{bn}(M_1 + M_2) & = & \mathit{bn}(M_1) \cup \mathit{bn}(M_2) \\
\mathit{fn}(K\lfloor\tilde{a}\rfloor) & = & \{\tilde{a}\} & \mathit{bn}(K\lfloor\tilde{a}\rfloor) & = & \emptyset \\
\mathit{fn}(P_1 \mid P_2) & = & \mathit{fn}(P_1) \cup \mathit{fn}(P_2) & \mathit{bn}(P_1 \mid P_2) & = & \mathit{bn}(P_1) \cup \mathit{bn}(P_2) \\
\mathit{fn}(\nu a.P) & = & \mathit{fn}(P) \setminus \{a\} & \mathit{bn}(\nu a.P) & = & \mathit{bn}(P) \cup \{a\}
\end{array}
$$

Figure 2.3: Definitions of free and bound names of π-Calculus-processes

Example 2.3.2. Let $P \equiv \nu ip.\overline{url}\langle ip\rangle.ip(s).s(x).C\lfloor url\rfloor$. Then $\mathit{fn}(P) = \{url\}$ and $\mathit{bn}(P) = \{ip, s, x\}$.

Convention 2.3.2. Let $P \in \mathcal{P}$ with the defining equations $K_i(\tilde{x_i}) := P_i$ for $1 \leq i \leq n$.

(i) The free names of a process are included in the parameter list of its defining equation, $\mathit{fn}(P_i) \subseteq \tilde{x_i}$ for all $1 \leq i \leq n$.

(ii) The bound and free names of two processes are disjoint, $\mathit{fn}(Q) \cap \mathit{bn}(R) = \emptyset$ for $Q, R \in \{P, P_1, \ldots, P_n\}$.

(iii) For all processes in $Q, R \in \{P, P_1, \ldots, P_n\}$, the bound names of Q differ from the bound names of R, i.e. $\mathit{bn}(Q) \cap \mathit{bn}(R) = \emptyset$.

(iv) No name is bound twice in a process. That is, for $\nu a.P$ and $b(a).P$, we get $a \notin \mathit{bn}(P)$. Furthermore, $\mathit{bn}(P) \cap \mathit{bn}(Q) = \emptyset$ for $P \mid Q$.

Substitutions are functions renaming free names of a process. Since substitutions are crucial for the definition of the operational semantics of process equations, we recall their definition. A substitution maps names onto names, i.e., $\sigma\colon \mathcal{A} \to \mathcal{A}$. The image of a name x under the substitution σ is denoted by $\sigma(x)$. If we restrict the domain and image of σ to two sets of names $A, B \subseteq \mathcal{A}$, then $\sigma(x) \in B$ if $x \in A$ and $\sigma(x) = x$ otherwise. As an abbreviation, we use $\{a_1, \ldots, a_n/x_1, \ldots, x_n\}$ for the substitution $\sigma(x_i) = a_i$, where $1 \leq i \leq n$ and $\sigma(x) = x$ otherwise. To achieve a well-formed application of a substitution σ to a process P, we demand that $\mathit{bn}(P) \cap (\mathit{im}(\sigma) \cup \mathit{dom}(\sigma)) = \emptyset$.

Definition 2.3.2 (Application of Substitutions)**.** Let $\sigma\colon \mathcal{A} \to \mathcal{A}$ be a substitution and P be a process. The application of σ to P, denoted by $P\sigma$ is defined inductively by:

$$\begin{array}{rcl}
(\tau.P)\sigma & = & \tau.(P\sigma) \\
(x(y).P)\sigma & = & \sigma(x)(y).(P\sigma) \\
(\overline{x}\langle y\rangle.P)\sigma & = & \overline{\sigma(x)}\langle\sigma(x)\rangle.(P\sigma) \\
(M+N)\sigma & = & M\sigma + N\sigma \\
(P \mid Q)\sigma & = & P\sigma \mid Q\sigma \\
(\nu a.P)\sigma & = & \nu a.(P\sigma) \\
K\lfloor\tilde{a}\rfloor\sigma & = & K\lfloor\sigma(\tilde{a})\rfloor \\
\mathbf{0}\sigma & = & \mathbf{0}
\end{array}$$

A relation compatible with the operators of the π-Calculus is called a congruence relation. That is, the relation is preserved under all possible compositions.

Definition 2.3.3 (Congruence Relation)**.** Let $\sim$ be an equivalence relation on $\mathcal{P}$, i.e., $\sim$ is reflexive, symmetric and transitive. If $\sim$ satisfies the following properties, we call $\sim$ a *congruence relation.*

$$\begin{array}{rlll}
\forall P,Q,M \in \mathcal{P}\colon \forall\pi\colon & P \sim Q & \text{implies} & \pi.P + M \sim \pi.Q + M \\
\forall P,Q,R \in \mathcal{P}\colon & P \sim Q & \text{implies} & P \mid R \sim Q \mid R \\
\forall P,Q \in \mathcal{P}\colon \forall a \in \mathcal{A} & P \sim Q & \text{implies} & \nu a.P \sim \nu a.Q
\end{array}$$

We use the structural congruence to identify processes with similar structural properties. This congruence will play an important role for defining the behaviour of processes. Furthermore, it is crucial for the structural semantics of processes and for the semantics of the logic we define in Chapter 3.

Definition 2.3.4 (Structural Congruence)**.** The *structural congruence* $\equiv$ is the smallest congruence relation on processes allowing the alpha-conversion of bound names, where $+$ and $\mid$ are commutative and associative operators with $\mathbf{0}$ as the neutral element. The quantifier ν is commutative and is absorbed by $\mathbf{0}$. The scope of νn can be extended and shrunk over all processes which do not contain n free. These properties are formalised as follows, where M, N, O are summations and P, Q, R processes.

$$\begin{array}{llll}
M & \equiv & M + \mathbf{0} & & \text{(Neutr-Sum)}\\
M + N & \equiv & N + M & & \text{(Com-Sum)}\\
M + (N + O) & \equiv & (M + N) + O & & \text{(Ass-Sum)}\\
P & \equiv & P \,|\, \mathbf{0} & & \text{(Neutr-Par)}\\
P \,|\, Q & \equiv & Q \,|\, P & & \text{(Com-Par)}\\
(P \,|\, Q) \,|\, R & \equiv & P \,|\, (Q \,|\, R) & & \text{(Ass-Par)}\\
\nu a.\mathbf{0} & \equiv & \mathbf{0} & & \text{(Neutr-Res)}\\
\nu a.\nu b.P & \equiv & \nu b.\nu a.P & & \text{(Com-Res)}\\
\nu a.(P|Q) & \equiv & P|\nu a.Q & a \notin \mathit{fn}(P) & \text{(Scope-Extr)}
\end{array}$$

We will denote the equivalence class of a process P with respect to structural congruence by $[P]$.

A well-known representation of a process P is the standard form of P [Mil99, SW01]. Informally, the standard form of P can be obtained by increasing the scope of every restricted name not inside a summation as much as possible and by the application of alpha-conversion where necessary.

Definition 2.3.5 (Standard Form)**.** A process P^{sf} is in *standard form* if and only if it can be constructed with the following syntax.

$$\begin{array}{rcl}
P^{\neq\nu} & ::= & M \;\mid\; K\lfloor\tilde{a}\rfloor \;\mid\; P_1^{\neq\nu} \,|\, P_2^{\neq\nu}\\
P^{sf} & ::= & \mathbf{0} \;\mid\; P^{\neq\nu} \;\mid\; \nu a.P^{sf}
\end{array}$$

where $a \in \mathit{fn}(P^{sf})$ and M a non-empty summation. We will denote the set of all processes in standard form by $\mathcal{P}^{sf}$.

The well-known fact that every process is structurally congruent with a process in standard form is often used in proofs.

Lemma 2.3.1. *Every process $P \in \mathcal{P}$ is congruent to a process in standard form $P^{sf} \in \mathcal{P}^{sf}$.*

While the structural congruence will play an important role for the definition of our logic (see forth Chapter 3) we want to reason about how the structure of processes changes over time. Therefore, we need the reaction relation as defined by Milner [Mil99, SW01]. This relation models the internal communication behaviour of a process P, i.e., how the components of P interact with each other.

Definition 2.3.6 (Reaction Relation)**.** The *reaction relation* $\rightarrow \subseteq \mathcal{P} \times \mathcal{P}$ is defined by the following rules.

$$\begin{array}{rc}
\text{(Tau)} & \tau.P + M \rightarrow P \\
\text{(React)} & (x(y).P + M) \,|\, (\overline{x}\langle z\rangle.Q + N) \rightarrow P\{z/y\} \,|\, Q \\
\text{(Const)} & K\lfloor\tilde{a}\rfloor \rightarrow P\{\tilde{a}/\tilde{x}\}, \text{ if } K(\tilde{x}) := P
\end{array}$$

$$\text{(Par)}\ \frac{P \rightarrow P'}{P \,|\, Q \rightarrow P' \,|\, Q} \qquad \text{(Res)}\ \frac{P \rightarrow P'}{\nu a.P \rightarrow \nu a.P'}$$

$$\text{(Struct)}\ \frac{Q \equiv P \qquad P \rightarrow P' \qquad P' \equiv Q'}{Q \rightarrow Q'}$$

For every $P \in \mathcal{P}$ there are only finitely many structural incongruent processes Q with $P \rightarrow Q$, i.e the reaction relation is image-finite up to structural congruence. The set of all processes reachable from P is denoted by $\mathit{Reach}(P) = \{P' \mid P \rightarrow^* P'\}$.

Example 2.3.3. Consider the client and server processes from Example 2.3.1 and the process $P \equiv C\lfloor url\rfloor \,|\, S\lfloor url\rfloor$. Then P can react by (Const) and (Par) to

$$\begin{array}{rl}
& \nu ip.\overline{url}\langle ip\rangle.ip(s).s(x).C\lfloor url\rfloor \,|\, S\lfloor url\rfloor \\
\rightarrow & \nu ip.\overline{url}\langle ip\rangle.ip(s).s(x).C\lfloor url\rfloor \,|\, url(y).\nu ses.\overline{y}\langle ses\rangle.\overline{ses}\langle ses\rangle.S\lfloor url\rfloor \\
\rightarrow & \nu ip.(ip(s).s(x).C\lfloor url\rfloor \,|\, \nu ses.\overline{ip}\langle ses\rangle.\overline{ses}\langle ses\rangle.S\lfloor url\rfloor) \\
\rightarrow & \nu ses.(ses(x).C\lfloor url\rfloor \,|\, \overline{ses}\langle ses\rangle.S\lfloor url\rfloor) \\
\rightarrow & C\lfloor url\rfloor \,|\, S\lfloor url\rfloor
\end{array}$$

The first reaction above is again an application of (Const) and (Par), while the last three reactions can be derived by the application of (React), (Struct), and (Res). Observe that first two reactions could also be swapped. Since the last process shown above is structurally congruent to P, we conclude that the transition system $\mathit{Reach}(P)$ is finite up to structural congruence.

We proceed with the definition of bisimilarity.

Definition 2.3.7 (Bisimulation)**.** Let P and Q be processes. We call the relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ a *simulation* if and only if whenever $(P, Q) \in \mathcal{R}$ and $P \rightarrow P'$, then there exists a Q' such that $Q \to Q'$ and $(P', Q') \in \mathcal{R}$. If additionally for $(P, Q) \in \mathcal{R}$ and for all Q' with $Q \to Q'$ there exists a P' such that $P \to P'$ and $(P', Q') \in \mathcal{R}$, we call $\mathcal{R}$ a *bisimulation.* If $\mathcal{R}$ is a bisimulation and $(P, Q) \in \mathcal{R}$, we call P and Q *bisimilar.*

2.4 Structural Semantics

Since the development of process algebras, there have been many approaches to model the operational semantics by automata-theoretic models. These models are normally intended to represent the concurrent behaviour of the processes. In this section, we present the structural semantics by Meyer [Mey08] which is a translation of π-Calculus processes into Petri nets, but with a different intention. Instead of modelling the smallest components of processes, i.e. the sequential processes, as places of the Petri nets, the structural semantics identify substructures of the processes connected by restricted names with the places. Therefore the restricted form, a normal form like the standard form of Definition 2.3.5 is employed. It is a parallel composition of *fragments* which are the mentioned substructures connected by restricted names.

Definition 2.4.1 (Fragments and the Restricted Form)**.** A process is a *fragment* if it can be constructed by the following definition

$$F \ ::= \ M \ \mid \ K\lfloor \tilde{a} \rfloor \ \mid \ \nu a.(F_1 \mid \ldots \mid F_n),$$

where M is a non-empty choice and $a \in \mathit{fn}(F_i)$ for all $1 \leq i \leq n$. We denote the set of all fragments by $\mathcal{P}_{\mathcal{F}}$. A process is in *restricted form* if it is a parallel composition of fragments, i.e., $P_{\mathit{rf}} = \Pi_{i \in I} F_i$. The set of all processes in restricted form is $\mathcal{P}_{\nu}$.

As for the standard form, every process is structurally congruent to a process in restricted form. Intuitively, the scope of restricted names has to be shrunk as much as possible to get the restricted form of a process. The formal definition of this intuition is given by Definition 2.4.2 while Lemma 2.4.1 is the equivalent to Lemma 2.3.1.

Lemma 2.4.1. *Every process P is structurally congruent with a process $\mathit{rf}(P)$ in restricted form. For a process in restricted form P_{rf} we have $\mathit{rf}(P_{\mathit{rf}}) = P_{\mathit{rf}}$.*

This lemma has been proved by Meyer [Mey08] by explicitly defining the function $\mathit{rf} : \mathcal{P} \to \mathcal{P}_{\nu}$. Since this function is essentially for the translation we will define, we briefly recall its definition.

Definition 2.4.2 ($\mathit{rf}\colon \mathcal{P} \to \mathcal{P}_\nu$)**.**

$$\begin{aligned}
\mathit{rf}(M+N) &= M+N\\
\mathit{rf}(K\lfloor\tilde{a}\rfloor) &= K\lfloor\tilde{a}\rfloor\\
\mathit{rf}(P_1 \mid P_2) &= \mathit{rf}(P_1) \mid \mathit{rf}(P_2)\\
\mathit{rf}(\nu a.P) &= \nu a.(\Pi_{i\in I} F_i) | \Pi_{i\in I\setminus I_a} F_i,
\end{aligned}$$

where $\mathit{rf}(P) = \Pi_{i\in I} F_i$ with $a \in \mathit{fn}(F_i)$ if and only if $i \in I_a \subseteq I$.

Furthermore, the restricted form is invariant up to the equivalence relation $\equiv_{\mathit{rf}}$. This relation is the smallest equivalence relation satisfying commutativity and associativity of parallel composition ((Com-Par) and (Ass-Par)) and the replacement of fragments by structural congruent fragments, i.e., if F and G are fragments and $F \equiv G$, then $F \mid P \equiv_{\mathit{rf}} G \mid P$.

Lemma 2.4.2. *For two processes $P, Q \in \mathcal{P}$, $P \equiv Q$ if and only if $\mathit{rf}(P) \equiv_{\mathit{rf}} \mathit{rf}(Q)$.*

Another crucial element for the structural semantics is the decomposition function *dec* which yields for a process in restricted form P_{rf} and a fragment F of P_{rf} the number of occurrences of F inside of P_{rf}. This function will also be important for the translation of our logic to LTL in Chapter 5.

Definition 2.4.3 (Decomposition Function $\mathit{dec}\colon \mathcal{P}_\nu \to \mathbb{N}^{\mathcal{P}_{\mathcal{F}/\equiv}}$)**.** For every process in restricted form $P_{\mathit{rf}} = \Pi_{i\in I} F_i$, we define the function $\mathit{dec}(P_{\mathit{rf}})\colon \mathcal{P}_{\mathcal{F}/\equiv} \to \mathbb{N}$ by $\mathit{dec}(P_{\mathit{rf}})([F]) := |I_F|$, where $I_F \subseteq I$ such that $F \equiv F_i$ if and only if $i \in I_F$.

The decomposition function is invariant under $\equiv_{\mathit{rf}}$ and is compatible with parallel composition.

Lemma 2.4.3. *Let $P_{\mathit{rf}}, Q_{\mathit{rf}} \in \mathcal{P}_\nu$. Then $P_{\mathit{rf}} \equiv_{\mathit{rf}} Q_{\mathit{rf}}$ if and only if $\mathit{dec}(P_{\mathit{rf}}) = \mathit{dec}(Q_{\mathit{rf}})$. Furthermore $\mathit{dec}(P_{\mathit{rf}} \mid Q_{\mathit{rf}}) = \mathit{dec}(P_{\mathit{rf}}) + \mathit{dec}(Q_{\mathit{rf}})$.*

Lemma 2.4.2 and Lemma 2.4.3 again have been proved by Meyer [Mey08].

We are now ready to define the structural semantics $\mathcal{N}[\![P]\!]$ of a π-Calculus process P. The places of $\mathcal{N}[\![P]\!]$ are all reachable fragments of P, the transitions represent the possible reactions of P. The weight of every arc from (resp. to) a transition represents the number of fragments needed (resp. created) by this transition. The initial marking of $\mathcal{N}[\![P]\!]$ is simply the decomposition of P into fragments.

Definition 2.4.4 (Structural Semantics $\mathcal{N}[\![\cdot]\!] \colon \mathcal{P} \to \mathcal{PN}$)**.** The *structural semantics* is a mapping of a π-Calculus process $P \in \mathcal{P}$ to a Petri net $\mathcal{N}[\![P]\!] := (S, T, \lambda, M_0)$ defined as follows:

$$\begin{aligned} S &:= fg(rf(Reach(P)))_{/\equiv} \\ T &:= \{([F],[Q]) \in S \times \mathcal{P}_{/\equiv} \mid F \to Q\} \\ &\cup \{([F_1 \,|\, F_2],[Q]) \in \mathcal{P}_{/\equiv} \times \mathcal{P}_{/\equiv} \mid [F_1],[F_2] \in S \text{ and } F_1 \,|\, F_2 \to Q\} \\ M_0 &:= dec(rf(P)) \end{aligned}$$

For the definition of the weight function λ, consider the place $[F] \in S$ and two transitions $([F'],[Q]), ([F_1 \,|\, F_2],[Q]) \in T$:

$$\begin{aligned} \lambda([F],([F'],[Q])) &:= dec(F')([F]) \\ \lambda([F],([F_1 \,|\, F_2],[Q])) &:= dec(F_1 \,|\, F_2)([F]) \\ \lambda(([F'],[Q]),[F]) &:= \lambda(([F_1 \,|\, F_2],[Q]),[F]) := dec(rf(Q))([F]) \end{aligned}$$

Example 2.4.1. Consider the client/server definitions of Example 2.3.1 and the process

$$P \equiv C\lfloor url \rfloor \,|\, C\lfloor url \rfloor \,|\, S\lfloor url \rfloor$$

The set of reachable fragments factorized by $\equiv$ is P are

$$\begin{aligned} fg(rf(Reach(P)))_{/\equiv} = \; & \{[C\lfloor url \rfloor], [S\lfloor url \rfloor], \\ & [\nu ip.\overline{url}\langle ip \rangle.ip(s).s(x).C\lfloor url \rfloor], \\ & [url(y).\nu ses.\overline{y}\langle ses \rangle.\overline{ses}\langle ses \rangle.S\lfloor url \rfloor], \\ & [\nu ip.(ip(s).s(x).C\lfloor url \rfloor \,|\, \nu ses.\overline{ip}\langle ses \rangle.\overline{ses}\langle ses \rangle.S\lfloor url \rfloor)], \\ & [\nu ses.(ses(x).C\lfloor url \rfloor \,|\, \overline{ses}\langle ses \rangle.S\lfloor url \rfloor)]\} \end{aligned}$$

For the sake of clarity, we will abbreviate the fragments by

$$\begin{aligned} F_1 &\equiv C\lfloor url \rfloor \\ F_2 &\equiv S\lfloor url \rfloor \\ F_3 &\equiv \nu ip.\overline{url}\langle ip \rangle.ip(s).s(x).C\lfloor url \rfloor \\ F_4 &\equiv url(y).\nu ses.\overline{y}\langle ses \rangle.\overline{ses}\langle ses \rangle.S\lfloor url \rfloor \\ F_5 &\equiv \nu ip.(ip(s).s(x).C\lfloor url \rfloor \,|\, \nu ses.\overline{ip}\langle ses \rangle.\overline{ses}\langle ses \rangle.S\lfloor url \rfloor) \\ F_6 &\equiv \nu ses.(ses(x).C\lfloor url \rfloor \,|\, \overline{ses}\langle ses \rangle.S\lfloor url \rfloor) \end{aligned}$$

The initial marking of $\mathcal{N}[\![P]\!]$ is $M_0 = (2, 1, 0, 0, 0, 0)$ since P consists of two occurrences of F_1 and one occurrence of F_2. The structural semantics of P, i.e., $\mathcal{N}[\![P]\!]$ is shown in Figure 2.4.

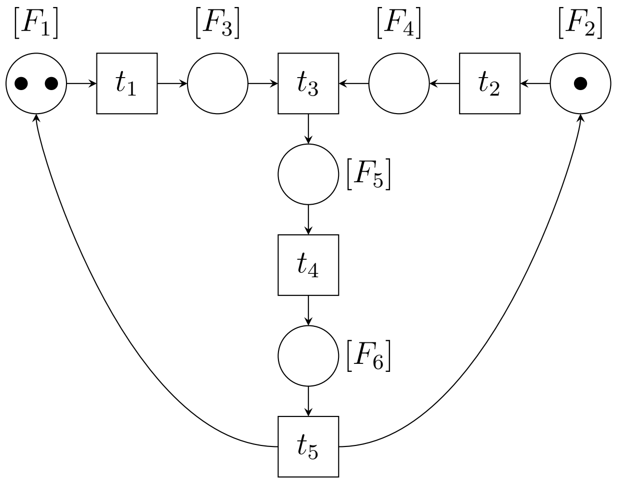

Figure 2.4: The structural semantics of $P \equiv C\lfloor url \rfloor \,|\, C\lfloor url \rfloor \,|\, S\lfloor url \rfloor$.

A important result found by Meyer is that the transition systems of a process and its structural semantics are isomorphic. The *dec* function yields a Petri net marking for a process $P \in \mathcal{P}$ and the inverse of *dec* is given by the following definition.

Definition 2.4.5 (Retrieve Function $\mathit{retrieve}\colon \mathbb{N}^S \to \mathcal{P}_{/\equiv}$)**.** Let $\mathcal{N}[\![P]\!] = (S, T, \lambda, M_0)$ be the structural semantics of $P \in \mathcal{P}$. For every marking $M \in \mathit{Reach}(\mathcal{N}[\![P]\!])$ the function $\mathit{retrieve}\colon \mathbb{N}^S \to \mathcal{P}_{/\equiv}$ yields the process class $[Q] \in \mathcal{P}_{/\equiv}$ defined by

$$\mathit{retrieve}(M) := \left[\Pi_{[F]\in \mathit{supp}(M)} \Pi^{M([F])} F\right]$$

With this function, we can define an isomorphism relating $\mathit{Reach}(P)_{/\equiv}$ with $\mathit{Reach}(\mathcal{N}[\![P]\!])$.

Proposition 2.4.1. *Given a process $P \in \mathcal{P}$ and its structural semantics $\mathcal{N}[\![P]\!]$, there is an isomorphism $f\colon \mathit{Reach}(P)_{/\equiv}$ mapping $[Q]$ to $\mathit{dec}(\mathit{rf}(Q))$. The process associated to a marking can be reconstructed by $\mathit{retrieve}(f([Q])) = [Q]$.*

Furthermore, Meyer has identified a very large decidable subset of the π-Calculus, the *structurally stationary* processes. This class of processes includes all other currently known decidable sets of π-Calculus processes [Mey08]. A process is structurally stationary, if the set of reachable fragments up to structural congruence is finite. Meyer used an equivalent definition which we recall in Definition 2.4.6.

Definition 2.4.6 (Structural Stationarity)**.** We call a process $P \in \mathcal{P}$ *structurally stationary*, if there is a finite set S of fragments such that every fragment of a reachable process of P is structurally congruent to a fragment in S.

$$\exists \{F_1, \dots, F_n\} \colon \forall Q \in \mathit{Reach}(P) \colon \forall F \in \mathit{fg}(\mathit{rf}(Q)) \colon \exists i \colon F \equiv F_i.$$

For every process $P \in \mathcal{P}$ and a reachable process P', i.e., $P \rightarrow^* P'$, we have that the structural semantics of P' is either equal or smaller than the structural semantics of P, i.e., the places of $\mathcal{N}[\![P']\!]$ are a subset of the places of $\mathcal{N}[\![P]\!]$ and the same relation holds for the transitions and the weight function. This is reasonable, since all processes P' can react to are also reachable from P. Even though Lemma 2.4.4 is almost immediate by Definition 2.4.4, we prove it explicitly.

Lemma 2.4.4. *For two processes $P, P' \in \mathcal{P}$ and their Petri net semantics $\mathcal{N}[\![P]\!] = (S, T, \lambda, M_0)$ and $\mathcal{N}[\![P']\!] = (S', T', \lambda', M_0')$, $P \rightarrow P'$ implies that $S' \subseteq S$, $T' \subseteq T$ and $\lambda' \subseteq \lambda$.*

Proof *of Lemma 2.4.4*
Places: Let $P \rightarrow P'$. Then by definition $P' \in \mathit{Reach}(P)$. Furthermore $\mathit{Reach}(P') \subseteq \mathit{Reach}(P)$ and hence $\mathit{fg}(\mathit{Reach}(P'))_{/\equiv} \subseteq \mathit{fg}(\mathit{Reach}(P))_{/\equiv}$. Since $S' = \mathit{fg}(\mathit{Reach}(P'))_{/\equiv}$ and $S = \mathit{fg}(\mathit{Reach}(P))_{/\equiv}$ we conclude $S' \subseteq S$.
Transitions: Let t be a transition of $\mathcal{N}[\![P']\!]$, i.e., $t \in T'$. Then there are two possibilities for the structure of t. If $t = ([F], [Q])$ with $[F] \in S'$ and $[Q] \in \mathcal{P}_{/\equiv}$, then $[F] \in S$ since $S' \subseteq S$ and hence $t \in T$. If $t = ([F_1 | F_2], [Q])$ with $[F_1] \in S'$, $[F_2] \in S'$ and $[Q] \in \mathcal{P}_{/\equiv}$, then similarly $[F_1], [F_2] \in S$ and again $t \in T$.
Weight function: The definition of the weight function λ only depends on the *dec* function, i.e., λ' is the restriction of λ to $S' \times T' \cup T' \times S'$ and hence $\lambda' \subseteq \lambda$. $\square$

2.5 Linear Temporal Logic

Temporal logics are a subset of modal logics. The interpretation of time differentiates the possible temporal logics. First, time could be *dense* (*continuous*) or *discrete*. The former is often used in philosophical temporal logics but also in the analysis of real-time systems. Since neither standard Petri nets nor the π-Calculus are designed to deal with this type of systems, we concentrate our view on discrete time, i.e., the time has "gaps" between single moments x_1 and x_2.

There are essentially two different types of discrete-time logics, defined by the different views on the flow of time. On the one hand, *linear temporal logics* assume that every point in time x has a unique successor. On the other hand, the time in *branching time logics* can split such that every x may have more than one successor. Both approaches of temporal logics can be used to express properties of computational systems like safeness or liveness. In this thesis, we will restrict our view on *propositional linear temporal logic* (LTL or PTL) [Pnu77] on Petri nets. For the definition of the syntax of LTL, we need a fixed set of atomic propositions *Prop*. In the case of LTL on Petri nets, the atomic propositions have the form $p \geq c$, where p is a place of the Petri net and $c \in \mathbb{N}$.

Definition 2.5.1 (Syntax)**.** The syntax of LTL formulas is given by the following BNF.

$$\varphi ::= p \geq c \mid (\neg\varphi) \mid (\varphi \wedge \psi) \mid (\bigcirc\varphi) \mid (\Diamond\varphi)$$

We denote the set of all LTL formulas by $\mathcal{LTL}$.

The remaining Boolean operators are defined as abbreviations the usual way. The atomic formulas will be satisfied by the state of a Petri net, if and only if the state p carries at least c tokens. The Boolean operators are interpreted as usual. The formula $\bigcirc\varphi$ (read as "next φ") represents that in every state reachable by exactly one firing transition, φ holds. Similarly, $\Diamond\varphi$ (read as "finally φ" or "eventually φ") is true, if every computation of the Petri net reaches a state, where φ holds.

As convention, we use that $\neg$, $\bigcirc$ and $\Diamond$ bind stronger than $\wedge$. Then we can eliminate unnecessary parentheses to increase the readability of formulas.

Example 2.5.1. The formula

$$(\neg(\bigcirc p_1 \geq c)) \wedge (\Diamond p_2 \geq c)$$

can be written as

$$\neg\bigcirc p_1 \geq c \wedge \Diamond p_2 \geq c.$$

However, we will sometimes use parentheses which are unnecessary due to this convention, to highlight the structure of formulas.

For defining the semantics, we need the notation of sequences of states.

Definition 2.5.2 (Occurence Sequence)**.** An *occurrence sequence* is an infinite sequence $\sigma = M_0 M_1 M_2 \ldots$ of markings of a Petri net, where $M_i \,[\rangle\, M_{i+1}$. If there is a k such that M_k has no successors, we define $M_n = M_k$ for all $n \geq k$. We will denote the set of all occurrence sequences by $\mathcal{OC}$.

We define the semantics of a formula by the consequence relation $\models_{\text{LTL}}$.

Definition 2.5.3 (Semantics)**.** The truth of a formula φ is defined inductively with respect to an occurrence sequence and a point in time x, according to the structure of φ.

$$\begin{array}{rcl}
\sigma, x \models_{\text{LTL}} & p \geq c & \text{iff } M_x(p) \geq c \\
\sigma, x \models_{\text{LTL}} & \neg\varphi & \text{iff } \sigma, x \not\models_{\text{LTL}} \varphi \\
\sigma, x \models_{\text{LTL}} & \varphi \wedge \psi & \text{iff } \sigma, x \models_{\text{LTL}} \varphi \text{ and } \sigma, x \models_{\text{LTL}} \psi \\
\sigma, x \models_{\text{LTL}} & \bigcirc\varphi & \text{iff } \sigma, x+1 \models_{\text{LTL}} \varphi \\
\sigma, x \models_{\text{LTL}} & \Diamond\varphi & \text{iff } \exists y\colon y \geq x \text{ and } \sigma, y \models_{\text{LTL}} \varphi
\end{array}$$

A formula φ is *satisfied* by an occurrence sequence σ, if and only if $\sigma, 0 \models_{\text{LTL}} \varphi$. A Petri net P satisfies a formula φ, denoted by $P \models_{\text{LTL}} \varphi$, if and only if *all* occurrence sequences of the Petri net satisfy φ.

A typical abbreviation of LTL is the formula $\Box\varphi$ (read as "always" φ) defined by

$$\Box\varphi \overset{\text{def}}{=} \neg\Diamond\neg\varphi$$

Hence an occurrence sequence and a point in time x satisfy $\Box\varphi$ if and only if no $y \geq x$ exists such that $\sigma, y \models_{\text{LTL}} \neg\varphi$, i.e., $\forall y\colon y \geq x$ implies $\sigma, y \models_{\text{LTL}} \varphi$.

Example 2.5.2. Consider the Petri net $\mathcal{N}$ of Example 2.4 and

$$\varphi = [F_1] \geq 1 \wedge [F_2] \geq 1 \Rightarrow \Diamond[F_6] \geq 1$$

Then $\mathcal{N} \models_{\text{LTL}} \varphi$ since for all occurrence sequences σ of $\mathcal{N}$ we get $\sigma(0)([F_1]) = 2 \geq 1$ and $\sigma(0)([F_2]) = 1 \geq 1$ and all computations reach a marking M with $M([F_6]) = 1$ with at most 4 transitions.

Furthermore $\mathcal{N} \models_{\text{LTL}} \Box\varphi$ because after t_5 fires, we reach again a state where φ holds like described above and the markings in between trivially satisfy φ.

However, for

$$\varphi' = [F_1] \geq 1 \wedge [F_2] \geq 1 \Rightarrow \Diamond[F_6] \geq 2,$$

we get that $\mathcal{N} \not\models_{\text{LTL}} \varphi'$, since $M([F_6]) = 1$ or $M([F_6]) = 0$ for every marking $M \in \mathit{Reach}(\mathcal{N})$.

3 A connection based logic for the π-Calculus

This chapter deals with the definition of a structural and temporal logic for the π-Calculus called PSTL (*P*i-Calculus *S*tructural *T*emporal *L*ogic, pronounced "pistol"). We define purely structural formulas describing the configuration of a process at a single moment. These formulas will be used as atoms for the temporal formulas, defined by a syntax similar to LTL. Besides the standard Boolean connectors, we will define an operator resembling the parallel composition of processes and a quantifier on restricted names. Furthermore, the temporal modalities "next" ($\bigcirc$) and "eventually" ($\Diamond$) of LTL are elements of the syntax of PSTL. This logic has a strong connection to standard LTL, because we do not allow for every combination of the structural and temporal operators.

The chapter is organised as follows: In Section 3.1 we will define the structural formulas of PSTL together with some abbreviations. As mentioned above, these formulas serve as atoms of the temporal logic we present in Section 3.2. Finally we present some properties of PSTL, i.e., in particular equivalences of structural formulas in Section 3.3.

3.1 Reasoning about structural properties

Syntax

Our logic PSTL contains a complete set of operators to describe a momentary configuration of a process P.

Definition 3.1.1 (Syntax)**.** The syntax of the structural fragment of PSTL is given by the following BNF.

$$\varphi ::= \top \mid P \mid \mathit{free}(a) \mid (\neg\varphi) \mid (\varphi \wedge \psi) \mid (\text{res } a\ \varphi) \mid (\varphi \between \psi),$$

where $P \in \mathcal{P}$ and $a \in \mathcal{A}$. We will denote the set of all structural formulas by $\mathcal{PSL}$

The parallel composition operator $\varphi \between \psi$ mimics the parallel composition of two processes Q and R such that Q satisfies φ and R satisfies ψ. That a name m in a process P is bound by a restriction, i.e., $P \equiv \nu m.P'$ can be specified by the restriction quantifier res. To guarantee that $a \in fn(P)$, we use $\mathit{free}(a)$. The notation $\mathit{free}(\tilde{a})$ will be used as an abbreviation for $\mathit{free}(a_1) \wedge \mathit{free}(a_2) \wedge \cdots \wedge \mathit{free}(a_n)$, where $\tilde{a} = a_1, a_2, \ldots, a_n$. We chose different symbols than $|$ and ν to clearly distinguish the formulas from processes. We will denote the set of all formulas of the structural fragment by $\mathcal{PSL}$. The remaining standard Boolean operators $\vee, \Rightarrow$ and $\Leftrightarrow$ are defined as usual.

We adopt a priority of the operators to enhance the readability of formulas by omitting parentheses. The negation $\neg$ and the restriction quantifier res a bind stronger than conjunction $\wedge$, which again binds stronger than the parallel composition operator $\between$.

Example 3.1.1. The following formulas φ, ψ and χ are structural formulas of PSTL.

$$\begin{aligned}
\varphi &= \text{res } a \ \text{res } b \ (\overline{a}\langle x\rangle \between \neg\overline{b}\langle x\rangle.c(y)) \\
\psi &= \text{res } b \ \text{res } a \ (\neg\overline{b}\langle x\rangle.c(y) \between \overline{a}\langle x\rangle) \\
\chi &= \mathit{free}(a) \wedge (\text{res } b \ \overline{b}\langle x\rangle \between \text{res } c \ (\overline{c}\langle x\rangle + \overline{d}\langle x\rangle))
\end{aligned}$$

We call the elements a formula φ consists of the subformulas of φ. This notation will be used for a case distinction in the translation of structural formulas in Chapter 5.

Definition 3.1.2 (Subformula)**.** For every structural formula φ, the *set of subformulas* of φ is denoted by $sub(\varphi)$ and defined by

$$\begin{aligned}
sub(\top) &= \{\top\} & sub(P) &= \{P\} \\
sub(\mathit{free}(a)) &= \{\mathit{free}(a)\} & sub(\neg\varphi) &= \{\neg\varphi\} \cup sub(\varphi) \\
sub(\varphi \wedge \psi) &= \{\varphi \wedge \psi\} \cup sub(\varphi) \cup sub(\psi) \\
sub(\text{res } a \ \varphi) &= \{\text{res } a \ \varphi\} \cup sub(\varphi) \\
sub(\varphi \between \psi) &= \{\varphi \between \psi\} \cup sub(\varphi) \cup sub(\psi)
\end{aligned}$$

We call a formula ψ with $\psi \in sub(\varphi)$ a *subformula* of φ.

Example 3.1.2. The subformulas of φ in Example 3.1.1 are

$$\begin{aligned}
sub(\varphi) = \{&\text{res } a \ \text{res } b \ (\overline{a}\langle x\rangle \between \neg\overline{b}\langle x\rangle.c(y)), \text{res } b \ (\overline{a}\langle x\rangle \between \neg\overline{b}\langle x\rangle.c(y)), \\
&\overline{a}\langle x\rangle \between \neg\overline{b}\langle x\rangle.c(y), \overline{a}\langle x\rangle, \neg\overline{b}\langle x\rangle.c(y), \overline{b}\langle x\rangle.c(y)\}
\end{aligned}$$

Since the formulas of PSTL are strongly related to π-Calculus processes, it is possible to define the sets of free and bound names of a formula. The free names of a formula are important for defining the semantics of the restriction operator res. The definition is based on the free names of π-Calculus processes, the atoms of PSTL.

Definition 3.1.3 (Free Names of a Formula)**.** The set of *free names of a formula* φ is $\mathit{fn}(\varphi)$, given by the recursive definition shown in Figure 3.1. Similarly, Figure 3.1 shows the definitions of the *bound names* of a formula φ, denoted by $bn(\varphi)$.

$$\begin{aligned}
\mathit{fn}(P) &= \mathit{fn}(P) & bn(P) &= bn(P)\\
\mathit{fn}(\mathit{free}(a)) &= \{a\} & bn(\mathit{free}(a)) &= \emptyset\\
\mathit{fn}(\neg\varphi) &= \mathit{fn}(\varphi) & bn(\neg\varphi) &= bn(\varphi)\\
\mathit{fn}(\varphi \wedge \psi) &= \mathit{fn}(\varphi) \cup \mathit{fn}(\psi) & bn(\varphi \wedge \psi) &= bn(\varphi) \cup bn(\psi)\\
\mathit{fn}(\text{res } a\ \varphi) &= \mathit{fn}(\varphi) \setminus \{a\} & bn(\text{res } a\ \varphi) &= bn(\varphi) \cup \{a\}\\
\mathit{fn}(\varphi \between \psi) &= \mathit{fn}(\varphi) \cup \mathit{fn}(\psi) & bn(\varphi \between \psi) &= bn(\varphi) \cup bn(\psi)
\end{aligned}$$

Figure 3.1: Definitions of *free* and *bound* names of a formula.

Example 3.1.3. Consider φ in Example 3.1.1. Then $\mathit{fn}(\varphi) = \{c, x\}$ and $bn(\varphi) = \{a, b, y\}$.

To replace a free name in a formula by a new one, we lift the *substitution of names* to formulas. A substitution σ is a function of type $\mathcal{A} \to \mathcal{A}$. For $0 \leq i \leq n$, we denote by $\{a_0, \ldots, a_n \leftarrow m_0, \ldots, m_n\}$ or $\{\tilde{a} \leftarrow \tilde{m}\}$ the substitution with $\sigma(a_i) = m_i$ and $\sigma(x) = x$ for all $x \neq a_i$, where $\tilde{a} = a_0, \ldots, a_n$ and $\tilde{m} = m_0, \ldots, m_n$. For the safe application of substitution, i.e., renaming only free and not bound names, we assume that the bound names of a formula and the image and domain of the substitution are disjoint, i.e., $bn(\varphi) \cap (im(\sigma) \cup dom(\sigma)) = \emptyset$.

Definition 3.1.4 (Application of Substitution)**.**

$$\begin{aligned}
P\{\tilde{a} \leftarrow \tilde{m}\} &= P\{\tilde{m}/\tilde{a}\}\\
\mathit{free}(a)\sigma &= \mathit{free}(\sigma(a))\\
(\neg\varphi)\sigma &= \neg(\varphi\sigma)\\
(\varphi \wedge \psi)\sigma &= \varphi\sigma \wedge \psi\sigma\\
(\text{res } b\ \varphi)\sigma &= \text{res } b\ (\varphi\sigma)\\
(\varphi \between \psi)\sigma &= \varphi\sigma \between \varphi\sigma
\end{aligned}$$

The following Lemma 3.1.1 is heavily used to prove equivalence of formulas. It shows commutativity of substitutions σ and σ', where the domains and images of the substitutions are disjoint.

Lemma 3.1.1 (Commutativity of Substitution)**.** *Let φ be a structural formula and $a, b, m, n \in \mathcal{A}$ different names. Then*

$$\varphi\{a \leftarrow m\}\{b \leftarrow n\} = \varphi\{b \leftarrow n\}\{a \leftarrow m\}$$

Proof *of Lemma 3.1.1*
By induction on the structure of formulas. □

Semantics

We present the semantics based on the structural consequence relation $\models_S$. This relation is defined according to the structure of formulas.

Definition 3.1.5. Let $P \in \mathcal{P}$ and a be a name, i.e., $a \in \mathcal{A}$. Furthermore let $\varphi, \psi \in \mathcal{PSL}$ be structural formulas according to Definition 3.1.1. Then the consequence relation $\models_S$ is defined by

$$\begin{array}{ll}
P \models_S \top & \text{for all } P \in \mathcal{P} \\
P \models_S P' & \text{iff } P \equiv P' \\
P \models_S \mathit{free}(a) & \text{iff } a \in \mathit{fn}(P) \\
P \models_S \neg\varphi & \text{iff } P \not\models_S \varphi \\
P \models_S \varphi \wedge \psi & \text{iff } P \models_S \varphi \text{ and } P \models_S \psi \\
P \models_S \text{res } a\ \varphi & \text{iff } \exists m \in \mathcal{A}\colon \exists P' \in \mathcal{P}\colon m \notin \mathit{fn}(P) \cup \mathit{fn}(\varphi) \\
 & \qquad \text{and } P \equiv \nu m.P' \text{ and } P' \models_S \varphi\{a \leftarrow m\} \\
P \models_S \varphi \between \psi & \text{iff } \exists P', P''\colon P \equiv P' \mid P'' \text{ and } P' \models_S \varphi \text{ and } P'' \models_S \psi
\end{array}$$

We define satisfaction and validity of formulas.

Definition 3.1.6 (Satisfaction and Validity)**.** A process P *satisfying* a formula φ is denoted by $P \models_S \varphi$. We write $\models_S \varphi$ if and only if all processes satisfy φ, i.e.,

$$\models_S \varphi \quad \text{iff} \quad \forall P \in \mathcal{P}\colon P \models_S \varphi.$$

We then say that φ is *valid*.

Example 3.1.4. Some examples for processes satisfying formulas would be

$$\begin{aligned}\nu b.\overline{b}\langle x\rangle &\models_S \text{res } a\ (\overline{a}\langle x\rangle)\\ \nu a.(\overline{a}\langle x\rangle \mid a(x).(\overline{x}\langle c\rangle + \overline{x}\langle b\rangle)) &\models_S \text{res } y\ (\overline{y}\langle x\rangle \between \top)\end{aligned}$$

An example for a valid formula is

$$\varphi \;=\; \text{res } a\ (\neg \mathit{free}(a) \wedge (\overline{b}\langle x\rangle \between \top)) \Rightarrow (\overline{b}\langle x\rangle \between \top)$$

Let P be a process satisfying res a $(\neg \mathit{free}(a) \wedge (\overline{b}\langle x\rangle \between \top))$ (otherwise P would trivially satisfy φ). Then

$$\begin{aligned}P \equiv \nu m.P' \quad &\text{and} \quad m \notin \mathit{fn}(P) \cup \{a, b, x\}\\ &\text{and} \quad P' \models_S (\neg \mathit{free}(a) \wedge (\overline{b}\langle x\rangle \between \top))\{a \leftarrow m\}\end{aligned}$$

Hence, $m \notin \mathit{fn}(P')$ and $P' \models_S \overline{b}\langle x\rangle \between \top$. The semantics of the parallel composition and the process atoms yields $P' \equiv \overline{b}\langle x\rangle \mid P''$. Furthermore, we get $P \equiv \nu m.P' \equiv P'$, since m is not free in P'. By transitivity of structural congruence, we have $P \equiv \overline{b}\langle x\rangle \mid P''$, i.e. $P \models_S \overline{b}\langle x\rangle \between \top$. Because P was arbitrary, we conclude that φ is valid.

While the definition of the consequence relation for the atomic formulas and the Boolean connectives is not surprising, the semantics of the restriction quantifier res a are more involved. But this definition is needed to establish a connection between the names in the process and the names in the formula under consideration. If we use a simpler definition by omitting the need for substituting a fresh name m, i.e., the definition is

$$P \models_S \text{res } a\ \varphi \qquad \text{iff } \exists m \in \mathcal{A}, P' \in \mathcal{P}\colon P \equiv \nu m.P' \text{ and } P' \models_S \varphi, \tag{3.1}$$

we end up with the result, that the set of processes satisfying a formula would depend on the choice of the restricted name a . Consider for example

$$\varphi = \text{res } x\ (\overline{x}\langle b\rangle.b(y))$$

together with the processes $P \equiv \nu a.(\overline{a}\langle b\rangle.b(y))$ and $Q \equiv \nu b.(\overline{a}\langle b\rangle.b(y))$. Intuitively, P should satisfy φ while Q should not. But with the simple restriction semantics given in (3.1), we have to conclude that both P and Q satisfy φ. If we examine P and φ, we end up with

$$P \models_S \text{res } x\ (\overline{x}\langle b\rangle.b(y)) \text{ iff } \quad \exists m \in \mathcal{A}, \exists P'\colon P \equiv \nu m.P' \text{ and } P' \models_S \overline{x}\langle b\rangle.b(y).$$

The only name bound by a restriction in P is a. If we use alpha-conversion to get $P \equiv P_1 = \nu x.(\overline{x}\langle b\rangle.b(y))$, we get $P_1' \equiv \overline{x}\langle b\rangle.b(y)$, i.e., $P \equiv \text{res } x\ (\overline{x}\langle b\rangle.b(y))$. Observe that we can not use alpha-conversion to achieve $Q \models_S \text{res } x\ (\overline{x}\langle b\rangle.b(y))$. But now consider

$$\varphi' = \text{res } a\ (\overline{a}\langle b\rangle.b(y)).$$

Now both $P \models_S \varphi'$ and $Q \models_S \varphi'$. The connection between the restricted name a and the use of a in $\overline{a}\langle b\rangle.b(y)$ is completely lost in the semantics.

The more complex definition above establishes this connection by demanding that m occurs free neither in P nor in φ and that m has to be substituted for x in φ. The expanded semantics of φ is

$$\begin{aligned} P \models_S \text{res } x\ (\overline{x}\langle b\rangle.b(y)) \qquad & \text{iff } \exists m \in \mathcal{A}, P' \in \mathcal{P}\colon P \equiv \nu m.P' \\ & \quad \text{and } m \notin \mathit{fn}(P) \cup \mathit{fn}(\overline{x}\langle b\rangle.b(y)) \\ & \quad \text{and } P' \models_S \overline{x}\langle b\rangle.b(y)\{x \leftarrow m\} \end{aligned}$$

The identification of m with a is possible, since $a \notin \mathit{fn}(P) \cup \mathit{fn}(\overline{x}\langle b\rangle.b(y))$. Furthermore, $P' \models_S \overline{x}\langle b\rangle.b(y)\{x \leftarrow a\}$. Hence, P satisfies φ. The process Q however does not satisfy φ. For $Q \models_S \varphi$, there has to be an Q' with $Q \equiv \nu m.Q'$ and $Q' \models_S \overline{x}\langle b\rangle.b(y)\{x \leftarrow m\}$. Since the only restriction in Q is on b, $Q' \equiv \overline{a}\langle b\rangle.b(y)$ is the only possibility. But for every $m \neq a$, $Q' \not\models_S \overline{x}\langle b\rangle.b(y)\{x \leftarrow m\}$. Since $a \in \mathit{fn}(Q)$, the choice $m = a$ is not allowed and hence $Q \not\models_S \varphi$. Some other properties of the restriction quantifier will be explored in Section 3.3.

Remark: Our semantics of the restriction quantifier follows the ideas for the hidden name quantifier ν of Caires and Cardelli [CC01, CG01], but we refrain from splitting our quantifier into a quantifier on fresh names and a revelation operator as Caires and Cardelli did. We think that such splitting is not helpful in our setting and only confuses the intuitive meaning of the quantifier. However, we lose expressiveness because of this decision. E.g. it is not possible to express that a process does not contain free names ◇

We give a notion of equivalence of formulas which we will call *structural equivalence*, since both formulas define essentially the same classes of processes.

Definition 3.1.7 (Structural Equivalence)**.** If for two formulas φ_1 and φ_2 and all processes $P \in \mathcal{P}$, $P \models_S \varphi_1$ iff $P \models_S \varphi_2$ holds, we write $\varphi_1 \mathrel{\hat{\equiv}} \varphi_2$ and call φ_1 *structurally equivalent* to φ_2 and vice versa.

There are many interesting properties, most of which are intuitively expected. For example, the formulas φ and ψ of Example 3.1.1 should be structurally equivalent due to the commutativity of the restriction quantifier and the parallel composition. And in fact, they are. But as shown above, the semantics of the restriction quantifier is rather unfamiliar. Hence we explicitly show the needed equivalences in Section 3.3, Lemma 3.3.5.

For the proofs of these and other useful equivalences, we will need the following lemma. It shows that our semantics is well-defined with respect to the substitution of free names.

Lemma 3.1.2 (Compatibility of Substitution with the Consequence Relation)**.** *Let φ be a structural formula, P be a process and $a, m \in \mathcal{A}$ such that $m \notin fn(P) \cup fn(\varphi)$. Then*

$$P \models_S \varphi \quad \textit{iff} \quad P\{m/a\} \models_S \varphi\{a \leftarrow m\}.$$

Proof *of Lemma 3.1.2*
By induction on the structure of formulas.
Base Case $P \models_S Q$:
Then $m \notin fn(P) \cup fn(Q)$ and $P \equiv Q$. Without loss of generalisation, we assume that $m \notin bn(P) \cup bn(Q)$, since otherwise we could employ alpha-conversion to achieve this. Hence $P\{m/a\} \equiv Q\{m/a\}$, i.e., $P\{m/a\} \models_S Q\{a \leftarrow m\}$. The other direction is similar.
Base Case $P \models_S free(a)$:
Then $a \in fn(P)$ and hence $m \in fn(P\{m/a\})$. Since $free(m) = free(a)\{a \leftarrow m\}$, $P\{m/a\} \models_S free(a)\{a \leftarrow m\}$.

Conversely, let $P\{m/a\} \models_S free(a)\{a \leftarrow m\}$, i.e., $m \in fn(P\{m/a\})$. Therefore $a \in fn(P)$, since $m \notin fn(P)$. Hence $P \models_S free(a)$.
Case $P \models_S \neg\varphi$:
This is equivalent to

$$\begin{array}{rll} & & P \not\models_S \varphi \\ \{\text{Induction Hypothesis}\} & \text{iff} & P\{m/a\} \not\models_S \varphi\{a \leftarrow m\} \\ \{\text{Semantics}\} & \text{iff} & P\{m/a\} \models_S \neg(\varphi\{a \leftarrow m\}) \\ \{\text{Application of Substitution}\} & \text{iff} & P\{m/a\} \models_S (\neg\varphi)\{a \leftarrow m\} \end{array}$$

Case $P \models_S \varphi \wedge \psi$:
The semantics of the conjunction yields

$$\begin{array}{rll} & & P \models_S \varphi \text{ and } P \models_S \psi \\ \{\text{Induction Hypothesis}\} & \text{iff} & P\{m/a\} \models_S \varphi\{a \leftarrow m\} \text{ and } P\{m/a\} \models_S \psi\{a \leftarrow m\} \\ \{\text{Semantics}\} & \text{iff} & P\{m/a\} \models_S \varphi\{a \leftarrow m\} \wedge \psi\{a \leftarrow m\} \\ \{\text{Application of Substitution}\} & \text{iff} & P\{m/a\} \models_S (\varphi \wedge \psi)\{a \leftarrow m\} \end{array}$$

Case $P \models_S$ res b φ:
This is equivalent to

$$P \equiv \nu n.P' \wedge n \notin fn(P) \cup fn(\varphi) \wedge P' \models_S \varphi\{b \leftarrow n\}$$

By the induction hypothesis, this is true if and only if

$$P'\{m/a\} \models_S \varphi\{b \leftarrow n\}\{a \leftarrow m\}.$$

We assume that $n \neq a$ and $n \neq m$, because otherwise we could use alpha-conversion to satisfy this condition. Since $b \in bn(\text{res } b\ \varphi)$, we conclude the same for b. Then we exploit the commutativity of substitutions (Lemma 3.1.1) to get the equivalence to

$$P'\{m/a\} \models_S \varphi\{a \leftarrow m\}\{b \leftarrow n\}.$$

Because $P\{m/a\} \equiv \nu n.(P'\{m/a\}$, the first statement holds if and only if

$$\begin{array}{c} P(\{m/a\}) \equiv \nu n.(P'\{m/a\}) \\ \text{and } n \notin fn(P) \cup fn(\varphi\{a \leftarrow m\}) \\ \text{and } P'\{m/a\} \models_S \varphi\{a \leftarrow m\}\{b \leftarrow n\} \end{array}$$

is true. Finally, the semantics yields $P\{m/a\} \models_S (\text{res } b\ \varphi)\{a \leftarrow m\}$.

Case $P \models_S \varphi \between \psi$:
The semantics yields the equivalence to

$$\begin{array}{rll}
 & & P \equiv P' \mid P'' \\
 & & \text{and } P' \models_S \varphi \\
 & & \text{and } P'' \models_S \psi \\
\{\text{Induction Hypothesis}\} & \text{iff} & P \equiv P' \mid P'' \\
 & & \text{and } P'\{m/a\} \models_S \varphi\{a \leftarrow m\} \\
 & & \text{and } P''\{m/a\} \models_S \psi\{a \leftarrow m\} \\
\{\text{Definition 2.3.2}\} & \text{iff} & P\{m/a\} \equiv P'\{m/a\} \mid P''\{m/a\} \\
 & & \text{and } P'\{m/a\} \models_S \varphi\{a \leftarrow m\} \\
 & & \text{and } P''\{m/a\} \models_S \psi\{a \leftarrow m\} \\
\{\text{Semantics}\} & \text{iff} & P\{m/a\} \models_S (\varphi \between \psi)\{a \leftarrow m\}
\end{array}$$

□

Often while specifying a property, one does not want to explicitly describe all restrictions on a process for the sake of readability. Therefore we introduce the operator res* as an abbreviation. First, we impose an arbitrary order on the free names of a formula. We then compute all subsequences $\tilde{n}$ of the free names and construct a disjunction of all possible restrictions, including no restriction. This definition considers all restrictions, since the restriction quantifier is commutative (refer to Lemma 3.3.5).

Definition 3.1.8 (Arbitrary Restriction Quantification)**.** For a formula φ, we use the following abbreviation.

$$\text{res}^* \varphi \overset{\text{def}}{=} \bigvee_{\tilde{n} \subseteq fn(\varphi)} \text{res } \tilde{n}\ \varphi$$

Example 3.1.5. Consider the formula res* $\neg\overline{c}\langle x\rangle$. Expanding res* yields

$$\text{res } c, x\ \neg\overline{c}\langle x\rangle \vee \text{res } c\ \neg\overline{c}\langle x\rangle \vee \text{res } x\ \neg\overline{c}\langle x\rangle \vee \neg\overline{c}\langle x\rangle.$$

With the help of this abbreviation we can define the formula $\diamondminus\varphi$ to express that *somewhere in* the process under consideration there is a process (possibly under some restrictions) which satisfies φ. The logically dual operator is $\boxminus$.

$$\begin{array}{rcl}
\diamondminus\varphi & \overset{\text{def}}{=} & \text{res}^*(\varphi \between \top) \\
\boxminus\varphi & \overset{\text{def}}{=} & \neg\diamondminus\neg\varphi
\end{array}$$

3.2 Reasoning about temporal progress

Syntax

As mentioned in the introduction, the formulas with temporal modalities are similar to standard LTL formulas, but the atomic formulas of PSTL are more complicated because they are structural formulas. This is formalised in Definition 3.2.1.

Definition 3.2.1 (Syntax)**.** The syntax of PSTL is defined inductively as follows

$$\varphi ::= (\varphi_S) \mid (\neg\varphi) \mid (\varphi \wedge \psi) \mid (\bigcirc\varphi) \mid (\Diamond\varphi),$$

where φ_S is a structural formula as defined in Section 3.1. We will denote the set of all PSTL formulas by $\mathcal{PSTL}$.

The formula $\bigcirc\varphi$ is true for a process P, if the process evolves within one reaction to a process P' satisfying φ. Similarly, P satisfies $\Diamond\varphi$, if there is a P' in all computations of P such that $P \to^* P'$ and $P' \models \varphi$. Like in Section 3.1, the remaining Boolean operators are defined as the usual abbreviations. We adopt the binding conventions of LTL to eliminate parenthesis, i.e., negation $\neg$, next $\bigcirc$ and finally $\Diamond$ bind stronger than conjunction $\wedge$. To clearly distinguish the structural from the temporal operators, we will parenthesize structural formulas φ_S if convenient.

Example 3.2.1. The formulas φ, ψ and χ are temporal formulas of PSTL.

$$\begin{aligned}
\varphi &= (a(x).\overline{x}\langle y\rangle \between \overline{a}\langle b\rangle) \Rightarrow \bigcirc\overline{b}\langle y\rangle \\
\psi &= \Box(\text{res } a\ (a(x) \between \mathit{free}(b)) \Rightarrow \Diamond \text{ res } a\ (\overline{b}\langle a\rangle \between \top)) \\
\chi &= \Box(\diamondminus(\text{res } c\ (c(y) \between (\nu d.(\overline{d}\langle x\rangle + \overline{a}\langle b\rangle.\overline{b}\langle x\rangle) \vee \overline{b}\langle x\rangle))) \vee (\mathbf{0} \mid \overline{a}\langle x\rangle)) \Rightarrow \Diamond\mathbf{0})
\end{aligned}$$

The formula χ shows, that the formulas of PSTL, although well-defined, can be confusing sometimes due to the huge set of operators. For a more concise specification, we introduce simpler atoms representing many sequential processes at once in Chapter 4. Furthermore, Lemma 3.3.4 in Section 3.3 will show that we can restrict the use of process to sequential ones. This means we can remove the parallel composition $\mid$ and the hidden name quantifier ν of the π-Calculus from our formulas without loss of expressiveness.

Semantics

To define the semantics of PSTL we need the notion of *process sequences* similar to the occurrence sequences of LTL. A process sequence models one sequence of reactions of a π-Calculus process.

Definition 3.2.2 (Process Sequence)**.** A *process sequence* π is an infinite sequence $\pi = P_0P_1P_2\ldots$, where $P_i \in \mathcal{P}$ and $P_i \rightarrow P_{i+1}$ for every $i \geq 0$. We will denote P_i by $\pi(i)$. If there is a k so that P_k has no reactions, we define $P_n = P_k$ for all $n \geq k$. We denote the set of all process sequences by $\mathcal{PC}$.

We proceed with the definition of the semantics of temporal formulas. It is similar to the semantics of LTL (refer to Definition 2.5.1 in Chapter 2).

Definition 3.2.3 (Semantics)**.** The semantics of temporal formulas is based on a process sequence π and an integer number x, with $x \geq 0$.

$$\begin{array}{ll}
\pi, x \models \varphi_S & \text{iff } \pi(x) \models_S \varphi_S \\
\pi, x \models \neg\varphi & \text{iff } \pi, x \not\models \varphi \\
\pi, x \models \varphi \wedge \psi & \text{iff } \pi, x \models \varphi \text{ and } \pi, x \models \psi \\
\pi, x \models \bigcirc\varphi & \text{iff } \pi, x+1 \models \varphi \\
\pi, x \models \Diamond\varphi & \text{iff } \exists y \geq x\colon \pi, y \models \varphi
\end{array}$$

Definition 3.2.4 (Satisfaction and Validity)**.** A process sequence π *satisfies* a PSTL formula φ, written $\pi \models \varphi$ if and only if $\pi, 0 \models \varphi$. A process P satisfies a formula φ, denoted by $P \models \varphi$, if and only if

$$\forall \pi \in \mathcal{PC}\colon \pi(0) = P \text{ implies } \pi \models \varphi.$$

We call a formula φ *valid*, if all processes $P \in \mathcal{P}$ satisfy φ, i.e.,

$$\models \varphi \quad \text{iff} \quad \forall P \in \mathcal{P}\colon P \models \varphi.$$

3.3 Properties of PSTL

In this section we examine some interesting properties of the structural fragment we will need for the translation of PSTL to LTL. We start with the observation that the set of processes satisfying a formula is closed under structural congruence.

Lemma 3.3.1. *Let $P \models_S \varphi$ and $P \equiv Q$. Then also $Q \models_S \varphi$.*

Proof *of Lemma 3.3.1*
By induction on the structure of φ. □

Now we will show that the purely structural formulas satisfy the Gabbay-Pitts-Property [GP99]. That is, if for a process P we can substitute a free name a of a formula φ with a fresh name m such that $P \models_S \varphi\{a \leftarrow m\}$, then we can choose *any* fresh name p instead of m and $P \models_S \varphi\{a \leftarrow p\}$ still holds. We will show that this property holds for all names m which are fresh with respect to a finite set N with $fn(P) \cup fn(\varphi) \subseteq N \subset \mathcal{A}$.

Lemma 3.3.2 (Gabbay-Pitts-Property)**.** *For all $P \in \mathcal{P}$, $\varphi \in \mathcal{PSL}$ and finite set $N \subset \mathcal{A}$ such that $fn(P) \cup fn(\varphi) \subseteq N$ and $a \in fn(\varphi)$, the following equality holds:*

$$\exists m \in \mathcal{A}\colon m \notin N \textit{ and } P \models_S \varphi\{a \leftarrow m\}$$

if and only if

$$\forall m \in \mathcal{A}\colon m \notin N \textit{ implies } P \models_S \varphi\{a \leftarrow m\}$$

Proof *of Lemma 3.3.2*
Let $P \in \mathcal{P}$, $\varphi \in \mathcal{PSL}$, $a \in \mathcal{A}$ and $N \subset \mathcal{A}$ such that N is finite, $fn(P) \cup fn(\varphi) \subseteq N$ and $a \in fn(\varphi)$.

First we assume that $\forall m \in \mathcal{A}\colon m \notin N$ implies $P \models_S \varphi\{a \leftarrow m\}$. Since N is finite and $\mathcal{A}$ is infinite, there is a name $p \notin N$. By the assumption, we get $P \models_S \varphi\{a \leftarrow m\}$. Hence $\exists p \in \mathcal{A}\colon p \notin N$ and $P \models_S \varphi\{a \leftarrow p\}$.

For the reverse direction we assume that there exists $m \in \mathcal{A}$ such that $m \notin N$ and $P \models_S \varphi\{a \leftarrow m\}$. Now we choose an arbitrary $p \in \mathcal{A}$ such that $p \notin N$.

1. If $p = m$ we get by the assumption that $P \models_S \varphi\{a \leftarrow p\}$.
2. If $p \neq m$ obviously $p \notin N \cup \{m\}$ is true. Since $fn(P) \cup fn(\varphi\{a \leftarrow m\}) \subseteq N$ also $p \notin fn(P) \cup fn(\varphi\{a \leftarrow m\})$, i.e., p is fresh with respect to P and $\varphi\{a \leftarrow m\}$. Hence by Lemma 3.1.2 $P\{p/m\} \models_S \varphi\{a \leftarrow m\}\{m \leftarrow p\}$. Because $m \notin N$, also $m \notin fn(P)$. We conclude $P \models_S \varphi\{a \leftarrow p\}$.

Since p was an arbitrary name, we get that for all $p \in \mathcal{A}$, $p \notin N$ implies $P \models_S \varphi\{a \leftarrow p\}$. □

The Gabbay-Pitts-Property formalises the intuition, that if a process P satisfies a formula φ, we can substitute free names of a formula, which do not occur free in P by fresh ones.

Example 3.3.1. Let $P \equiv \overline{a}\langle x\rangle$ and $\varphi = \neg\overline{b}\langle x\rangle$. Then we get, that $P \models_S \neg\overline{b}\langle x\rangle$ since $P \not\equiv \overline{b}\langle x\rangle$. Furthermore $fn(P) \cup fn(\varphi) = \{a, b, x\}$. Because also $P \models_S \neg\overline{m}\langle x\rangle$, i.e., $P \models_S \neg\overline{b}\langle x\rangle\{b \leftarrow m\}$ we get that $P \models_S \neg\overline{b}\langle x\rangle\{b \leftarrow m\}$ for all $m \notin \{a, b, x\}$. The constraint on m is a safe upper bound, since in this example we could choose $m = b$ or $m = x$ and still get the desired result that $P \models_S \neg\overline{b}\langle x\rangle\{b \leftarrow m\}$. However, if we would take $m = a$, P would not satisfy $\neg\overline{b}\langle x\rangle\{b \leftarrow a\}$.

On the other hand, consider $\varphi' = \overline{a}\langle x\rangle$. Then there is no $m \notin fn(P) \cup fn(\varphi')$ such that $P \models_S \overline{a}\langle x\rangle\{a \leftarrow m\}$.

Lemma 3.3.3 shows that the structural fragment of PSTL is νx-proper. This is a characterisation of well-definition of the restriction quantifier with respect to structural congruence as shown by Cardelli and Gordon [CG01]. The property can be described as follows. For a name n, a structural formula φ and a process P, there exists a process P' such that $P \equiv \nu n.P'$ and P' satisfies the formula φ if and only if n is not free in P and P satisfies the formula res a $(\varphi\{n \leftarrow a\})$, i.e., the formula φ, where every occurrence of n is substituted by a now restricted name a.

Lemma 3.3.3 (νx-proper)**.** *For a process $P \in \mathcal{P}$, the names $a, n \in \mathcal{A}$ and a formula $\varphi \in \mathcal{PSL}$, the following property holds.*

$$n \notin fn(P) \text{ and } P \models_S \text{res } a\ (\varphi\{n \leftarrow a\}) \quad \text{iff} \quad \exists P' \colon P \equiv \nu n.P' \text{ and } P' \models_S \varphi.$$

Proof *of Lemma 3.3.3*
Let $n \notin fn(P)$ and $P \models_S$ res a $(\varphi\{n \leftarrow a\})$. Then $P \equiv \nu m.Q$ where $m \notin fn(P) \cup fn(\varphi\{n \leftarrow a\})$ and $Q \models_S \varphi\{n \leftarrow a\}\{a \leftarrow m\}$, i.e., $Q \models_S \varphi\{n \leftarrow m\}$.

1. If $m = n$ then $Q \models_S \varphi$.

2. If $m \neq n$ then we get by Lemma 3.1.2 that $Q\{n/m\} \models_S \varphi$. Furthermore $P \equiv \nu m.Q \equiv \nu n.(Q\{n/m\})$.

Conversely assume $P \equiv \nu n.P'$ and $P' \models_S \varphi$. Now we choose two different arbitrary fresh names m and a, i.e., $m, a \notin fn(P) \cup fn(\varphi)$ and $m \neq a$. Then $P \equiv \nu n.P' \equiv \nu m.(P'\{m/n\})$. By Lemma 3.1.2 $P'\{m/n\} \models_S \varphi\{n \leftarrow m\}$. Hence $P'\{m/n\} \models_S \varphi\{n \leftarrow a\}\{a \leftarrow m\}$. Therefore $P \models_S$ res a $(\varphi\{n \leftarrow a\})$ and $n \notin fn(P)$. □

That this property is a good distinction between proper and flawed restriction quantifiers is not obvious. An example for a more natural characterisation would be

$$\nu n.P \models_S \text{res } a\ (\varphi\{n \leftarrow a\}) \quad \text{iff} \quad P \models_S \varphi. \tag{3.2}$$

Unfortunately, the direction from left to right would violate structural congruence. We give the example taken from Cardelli and Gordon, which clarifies this violation. Consider $\overline{b}\langle x\rangle \models_S \overline{b}\langle x\rangle$. By (3.2), we get that

$$\nu b.\overline{b}\langle x\rangle \models_S \text{res } a\ (\overline{b}\langle x\rangle\{b \leftarrow a\}).$$

Because $\nu b.\overline{b}\langle x\rangle \equiv \nu b.\nu b.\overline{b}\langle x\rangle$ and Lemma 3.3.1, we get

$$\nu b.\nu b.\overline{b}\langle x\rangle \models_S \text{res } a\ (\overline{b}\langle x\rangle\{b \leftarrow a\}).$$

Another application of (3.2) yields

$$\nu b.\overline{b}\langle x\rangle \models_S \overline{b}\langle x\rangle,$$

that is $\nu b.\overline{b}\langle x\rangle \equiv \overline{b}\langle x\rangle$, which is obviously wrong. The problem is the second application of 3.2, since the chosen process to satisfy $\overline{b}\langle x\rangle$ is not appropriate. Hence we use existential quantification in Lemma 3.3.3.

Now we concentrate our view on equivalences of formulas. Most of these equivalences work as expected, for example the commutativity of the parallel composition. Furthermore, we have found an equivalence resembling scope extrusion of restricted names (Lemma 3.3.6), which is unfortunately not as easy as we hoped, since we have to explicitly state that a name is not free with the help of the formula *free*(a). All equivalences are shown in Table 3.1.

The following equivalence shows that we can model all restrictions and parallel compositions of sequential processes with the operators of PSTL. By this result, we can restrict the atoms of the structural fragment to sequential processes without loss of expressiveness. These results will be used in Chapter 5.

Lemma 3.3.4. *Let $Q, R \in \mathcal{P}$. Then the following equivalences hold.*

(i) $Q \,|\, R \ \hat{\equiv}\ Q \between R$

(ii) $\nu a.Q \ \hat{\equiv}\ \text{res } a\ Q$

Proof *of Lemma 3.3.4*
(i) Let $P \models_S Q \,|\, R$. By the semantics this is equivalent to $P \equiv Q \,|\, R$, i.e. $P \equiv$

$P \mid Q$	$\hat{=}$	$P \between Q$	(seq-Par)
$\nu a.P$	$\hat{=}$	$\text{res } a\ P$	(seq-Res)
$\varphi \between \psi$	$\hat{=}$	$\psi \between \varphi$	(Com-Par$_\varphi$)
$(\varphi \between \psi) \between \chi$	$\hat{=}$	$\varphi \between (\psi \between \chi)$	(Ass-Par$_\varphi$)
$\text{res } a\ \text{ res } b\ \varphi$	$\hat{=}$	$\text{res } b\ \text{ res } a\ \varphi$	(Com-Res$_\varphi$)
$\text{res } a\ (\varphi \between \neg \mathit{free}(a) \wedge \psi)$	$\hat{=}$	$\text{res } a\ (\varphi) \between \psi$, where $a \notin \mathit{fn}(\psi)$	(Res-Scope)
$\neg \text{ res } a\ \varphi$	$\hat{=}$	$\text{res } a\ (\neg\varphi)$	(Neg-Res)
$\text{res } a\ (\varphi \between \top)$	$\hat{=}$	$\text{res } a\ (\varphi \between \top) \between \top$	(Par-True-Res)
$\text{res } a\ (\varphi \vee \psi)$	$\hat{=}$	$\text{res } a\ \varphi \vee \text{res } a\ \psi$	(Or-Res)
$(\varphi_1 \vee \varphi_2) \between \psi$	$\hat{=}$	$(\varphi_1 \between \psi) \vee (\varphi_2 \between \psi)$	(Or-Par)
$\neg \mathit{free}(a) \wedge (\varphi \between \psi)$	$\hat{=}$	$\neg \mathit{free}(a) \wedge \varphi \between \neg \mathit{free}(a) \wedge \psi$	(Neg-Free-Par)

Table 3.1: Structural Equivalences

$Q \mid R$. For every process $P' \in \mathcal{P}$, $P' \models_S P'$ and in particular $Q \models_S Q$ and $R \models_S R$. So, we can conclude that

$$P \models_S Q \mid R \text{ iff } P \equiv Q \mid R \text{ and } Q \models_S Q \text{ and } R \models_S R,$$

which is the definition of $P \models_S Q \between R$.
(ii) Let $P \models_S \nu a.Q$, which is equivalent to $P \equiv \nu a.Q$. By alpha-conversion we get $\nu a.Q \equiv \nu n.(Q\{n/a\})$, with $n \notin \mathit{fn}(Q)$. Since $\mathit{fn}(P) \subseteq \mathit{fn}(Q)$, we also have $n \notin \mathit{fn}(P) \cup \mathit{fn}(Q)$. Additionally, we know that $Q\{n/a\} \models_S Q\{a \leftarrow n\}$. In summary we have

$$\begin{aligned} P \models_S \nu a.Q \quad \text{iff} \quad & P \equiv \nu n.(Q\{n/a\}) \text{ and } n \notin \mathit{fn}(P) \cup \mathit{fn}(Q) \\ & \text{and } Q\{n/a\} \models_S Q\{a \leftarrow n\} \end{aligned}$$

This result concludes the proof, because with $Q\{n/a\} \equiv P'$, it is the definition of $P \models_S \text{res } a\ Q$. □

Our parallel composition operator of structural formulas is commutative and associative. The restriction quantifier is commutative. These properties are shown in Lemma 3.3.5.

Lemma 3.3.5 (Commutativity of $\between$ and res, Associativity of $\between$)**.** *Let φ, ψ and χ be structural formulas and $a, b \in \mathcal{A}$. Then*

(i) $\varphi \between \psi \mathrel{\hat{=}} \psi \between \varphi$

(ii) $(\varphi \between \psi) \between \chi \mathrel{\hat{=}} \varphi \between (\psi \between \chi)$

(iii) $\text{res } a \ \text{res } b\ \varphi \mathrel{\widehat{\equiv}} \text{res } b\ \text{res } a\ \varphi$

Proof *of Lemma 3.3.5*
(i) and (ii): Immediate by the semantics (Definition 3.1.5).
(iii) Let $P \models_S \text{res } a\ \text{res } b\ \varphi$, i.e., $P \equiv \nu m.\nu n.Q$, with $m \notin \mathit{fn}(P) \cup \mathit{fn}(\text{res } b\ \varphi)$, $n \notin \mathit{fn}(\nu n.Q) \cup \mathit{fn}(\varphi)$ and $Q \models_S \varphi\{a \leftarrow m\}\{b \leftarrow n\}$. Without loss of generalisation, we can choose m and n such that $a \neq m$ and $b \neq n$. Since $\mathit{fn}(\nu m.\nu n.Q) \subseteq \mathit{fn}(\nu n.Q)$ (Definition 2.3.1) and $\mathit{fn}(\text{res } a\ \varphi) \subseteq \mathit{fn}(\varphi)$ (Definition 3.1.3), we also have $n \notin \mathit{fn}(P) \cup \mathit{fn}(\text{res } a\ \varphi)$. Furthermore, we know that $P \equiv \nu m.\nu n.Q \equiv \nu n.\nu m.Q$.

Now we choose a fresh $m' \in \mathcal{A}$ to rename m by alpha-conversion, i.e. $m' \notin \mathit{fn}(Q) \cup \mathit{fn}(\varphi)$ and $P \equiv \nu n.\nu m'.Q\{m'/m\}$. Obviously $m' \notin \mathit{fn}(\nu m'.Q\{m'/m\}) \cup \mathit{fn}(\varphi)$ is true. Then

$$\begin{array}{rll} \{\text{Lemma 3.1.2}\} & Q\{m'/m\} & \models_S \varphi\{a \leftarrow m\}\{b \leftarrow n\}\{m \leftarrow m'\} \\ \Rightarrow & Q\{m'/m\} & \models_S \varphi\{a \leftarrow m'\}\{b \leftarrow n\} \\ \{\text{Lemma 3.1.1}\} \Rightarrow & Q\{m'/m\} & \models_S \varphi\{b \leftarrow n\}\{a \leftarrow m'\} \end{array}$$

By $Q\{m'/m\} \equiv Q'$, we get that $P \equiv \nu n.\nu m'.Q'$ with $n \notin \mathit{fn}(P) \cup \mathit{fn}(\text{res } a\ \varphi)$, $m' \notin \mathit{fn}(\nu m'.Q') \cup \mathit{fn}(\varphi)$ and $Q' \models_S \varphi\{b \leftarrow n\}\{a \leftarrow m'\}$, i.e., $P \models_S \text{res } b\ \text{res } a\ \varphi$.
The other direction is similar. □

Lemma 3.3.6 states an equivalent to scope extrusion of the structural congruence relation, even though it looks more complicated. The conjunction of the formula ψ which does not contain the name a with the assertion $\neg \mathit{free}(a)$ looks redundant, but is necessary for the equivalence. This is due to the observation that we need to ensure that neither the elements of φ contain the name a nor the process P contains the semantic counterpart to a, which we normally call m. Consider for example the formulas

$$\begin{array}{rcl} \varphi & = & \text{res } a\ (\overline{a}\langle x\rangle \between \neg\overline{b}\langle x\rangle) \\ \psi & = & \text{res } a\ \overline{a}\langle x\rangle \between \neg\overline{b}\langle x\rangle \end{array}$$

and the process

$$P \equiv \nu m.(\overline{m}\langle x\rangle \mid \overline{m}\langle x\rangle).$$

Clearly $a \notin \mathit{fn}(\neg\overline{b}\langle x\rangle)$. Furthermore $P \models_S \varphi$ but $P \not\models_S \psi$. Now consider

$$\varphi' = \text{res } a\ (\overline{a}\langle x\rangle \between \neg \mathit{free}(a) \wedge \neg\overline{b}\langle x\rangle).$$

Then we get that $P \not\models_S \varphi'$.

Lemma 3.3.6. *Let φ and ψ be structural formulas of PSTL. Furthermore let $a \in \mathcal{A}$ such that $a \notin fn(\psi)$. Then*

$$\text{res } a\ (\varphi \between \neg \mathit{free}(a) \wedge \psi) \quad \widehat{\equiv} \quad \text{res } a\ \varphi \between \psi$$

Proof *of Lemma 3.3.6*
Let $P \models_S \text{res } a\ (\varphi \wedge \neg \mathit{free}(a) \wedge \psi$. Then

$$P \equiv \nu m.P' \text{ and } m \notin fn(P) \cup fn(\varphi) \text{ and } P' \models_S (\varphi \between \neg \mathit{free}(a) \wedge \psi)\{a \leftarrow m\}.$$

Definition 3.1.4 and the semantics of the parallel composition yield

$$P' \equiv Q \mid R \text{ and } Q \models_S \varphi\{a \leftarrow m\} \text{ and } R \models_S (\neg \mathit{free}(a) \wedge \psi)\{a \leftarrow m\}.$$

That $R \models_S \neg \mathit{free}(a)\{a \leftarrow m\}$ is equivalent to $R \models_S \neg \mathit{free}(m)$, i.e. $m \notin fn(R)$ and since $a \notin fn(\psi)$, $R \models_S \psi\{a \leftarrow m\}$ can be simplified to $R \models_S \psi$. Now we can apply (Scope-Extr) to get

$$P \equiv \nu m.(Q \mid R) \equiv \nu m.Q \mid R.$$

By the semantics of the restriction quantifier and the parallel composition, we conclude

$$P \models_S \text{res } a\ \varphi \between \psi.$$

Conversely, let $P \models_S \text{res } a\ \varphi \between \psi$, i.e., $P \equiv Q \mid R$ where $Q \models_S \text{res } a\ \varphi$ and $R \models_S \psi$. Then $Q \equiv \nu m.Q'$ with $m \notin fn(Q) \cup fn(\varphi)$ and $Q \models_S \varphi\{a \leftarrow m\}$. We choose m such that $m \notin fn(Q) \cup fn(\varphi) \cup fn(\psi) \cup fn(R)$, which is possible due to alpha-conversion and the finiteness of $fn(\psi) \cup fn(R)$. Then by (Scope-Extr)

$$P \equiv \nu m.Q' \mid R \equiv \nu m.(Q' \mid R).$$

Because $a \notin fn(\psi)$, $R \models_S \psi\{a \leftarrow m\}$ and $R \models_S \neg \mathit{free}(m)$, i.e., $R \models_S (\neg \mathit{free}(a))\{a \leftarrow m\}$. Hence $R \models_S (\neg \mathit{free}(a) \wedge \psi)\{a \leftarrow m\}$ and

$$Q' \mid R \models_S \varphi\{a \leftarrow m\} \between (\neg \mathit{free}(a) \wedge \psi)\{a \leftarrow m\}.$$

Definition 3.1.4 and the semantics of the restriction operator yield

$$P \models_S \text{res } a\ (\varphi \between \neg \mathit{free}(a) \wedge \psi).$$

□

Furthermore, we have identified three distributive laws. The first is distribution of the restriction quantifier over disjunction. The other laws concern the parallel composition operator. That is, disjunction and the assertion that a name occurs not free are distributive over parallel composition. The laws are shown in Lemma 3.3.7.

Lemma 3.3.7 (Distributive Laws)**.** *For the formulas φ, ψ, χ and a name $a \in \mathcal{A}$, the following structural equivalences hold.*

(i) $\text{res } a\ (\varphi \vee \psi) \mathrel{\widehat{\equiv}} \text{res } a\ \varphi \vee \text{res } a\ \psi$

(ii) $\varphi \vee \psi \between \chi \mathrel{\widehat{\equiv}} (\varphi \between \chi) \vee (\psi \between \chi)$

(iii) $\neg \mathit{free}(a) \wedge (\varphi \between \psi) \mathrel{\widehat{\equiv}} \neg \mathit{free}(a) \wedge \varphi \between \neg \mathit{free}(a) \wedge \psi$

Proof *of Lemma 3.3.7*
(i) Let $P \models_S \text{res } a\ (\varphi \vee \psi)$. Then

$$P \equiv \nu m.P' \text{ and } m \notin \mathit{fn}(P) \cup \mathit{fn}(\varphi \vee \psi) \text{ and } P' \models_S (\varphi \vee \psi)\{a \leftarrow m\},$$

i.e., $P' \models_S \varphi\{a \leftarrow m\}$ or $P' \models_S \psi\{a \leftarrow m\}$. If $m \notin \mathit{fn}(\varphi \vee \psi)$ then $m \notin \mathit{fn}(\varphi)$ and $m \notin \mathit{fn}(\psi)$. Hence

$$P \equiv \nu m.P' \text{ and } m \notin \mathit{fn}(P) \cup \mathit{fn}(\varphi) \text{ and } P' \models_S \varphi\{a \leftarrow m\}$$

or

$$P \equiv \nu m.P' \text{ and } m \notin \mathit{fn}(P) \cup \mathit{fn}(\psi) \text{ and } P' \models_S \psi\{a \leftarrow m\}.$$

Therefore $P \models_S \text{res } a\ \varphi$ or $P \models_S \text{res } a\ \psi$, i.e., $P \models_S \text{res } a\ \varphi \vee \text{res } a\ \psi$.
Now let $P \models_S \text{res } a\ \varphi \vee \text{res } a\ \psi$. Then

$$P \equiv \nu m.Q \text{ and } m \notin \mathit{fn}(P) \cup \mathit{fn}(\varphi) \text{ and } Q \models_S \varphi\{a \leftarrow m\}$$

or

$$P \equiv \nu n.R \text{ and } n \notin \mathit{fn}(P) \cup \mathit{fn}(\psi) \text{ and } R \models_S \psi\{a \leftarrow n\}.$$

Now we choose a new name m' such that $m' \notin \mathit{fn}(P) \cup \mathit{fn}(\varphi) \cup \mathit{fn}(\psi)$, i.e. $m \notin \mathit{fn}(P) \cup \mathit{fn}(\varphi \vee \psi)$. Then with the application of alpha-conversion and Lemma 3.1.2

$$P \equiv \nu m'.(Q\{m'/m\}) \text{ and } m' \notin \mathit{fn}(P) \cup \mathit{fn}(\varphi \vee \psi) \text{ and } Q\{m'/m\} \models_S \varphi\{a \leftarrow m\}\{m \leftarrow m'\}$$

or

$$P \equiv \nu m'.(R\{m'/n\}) \text{ and } m' \notin \mathit{fn}(P) \cup \mathit{fn}(\varphi \vee \psi) \text{ and } R\{m'/n\} \models_S \psi\{a \leftarrow n\}\{n \leftarrow m'\}.$$

Hence

$$P \equiv \nu m'.P' \text{ and } m' \notin \mathit{fn}(P) \cup \mathit{fn}(\varphi \vee \psi) \text{ and } P' \models_S \varphi\{a \leftarrow m'\} \text{ or } P' \models_S \psi\{a \leftarrow m'\}.$$

The semantics and Definition 3.1.4 yield $P' \models_S (\varphi \vee \psi)\{a \leftarrow m'\}$. So we finally conclude $P \models_S \text{res } a\ (\varphi \vee \psi)$.
(ii) Let $P \models_S \varphi \vee \psi \between \chi$. By the semantics this is equivalent to

$$P \equiv Q \,|\, R \text{ and } Q \models_S \varphi \vee \psi \text{ and } R \models_S \chi \tag{3.3}$$

This statement is true, if and only if

$$Q \models_S \varphi \text{ or } Q \models_S \psi. \tag{3.4}$$

Rearranging (3.3) and (3.4) yields

$$P \equiv Q \,|\, R \text{ and } Q \models_S \varphi \text{ and } R \models_S \chi$$

or

$$P \equiv Q \,|\, R \text{ and } Q \models_S \psi \text{ and } R \models_S \chi.$$

Hence the equivalence to $P \models_S (\varphi \between \chi) \vee (\psi \between \chi)$.
(iii) Let $P \models_S \neg \mathit{free}(a) \wedge (\varphi \between \psi)$. This is equivalent to $a \notin \mathit{fn}(P)$ and $P \models_S \varphi \between \psi$, which again is equivalent to

$$P \equiv Q \,|\, R \text{ and } Q \models_S \varphi \text{ and } R \models_S \psi.$$

Because $\mathit{fn}(P) = \mathit{fn}(Q) \cup \mathit{fn}(R)$, $a \notin \mathit{fn}(P)$ if and only if a is neither free in Q nor in R. Hence $Q \models_S \neg \mathit{free}(a) \wedge \varphi$ and $R \models_S \neg \mathit{free}(a) \wedge \psi$ which is by Definition 3.1 equivalent to $P \models_S \neg \mathit{free}(a) \wedge \varphi \between \neg \mathit{free}(a) \wedge \psi$. □

A rather interesting property is the independence of negation and the restriction quantifier. That is, if we consider a formula of the form $\neg \text{res } a\ \varphi$, we can swap the negation and quantification.

Lemma 3.3.8.

$$\neg \text{res } a\ \varphi \mathrel{\widehat{\equiv}} \text{res } a\ \neg\varphi$$

Proof *of Lemma 3.3.8*
Let $P \models_S \neg \text{res } a$. By the semantics this is equivalent to $P \not\models_S \text{res } a\ \varphi$, i.e., $P \not\equiv \nu m.P'$ or $m \notin \mathit{fn}(P) \cup \mathit{fn}(\varphi)$ or $P' \neg \models_S \varphi\{a \leftarrow m\}$. Observe that due to the finiteness of $\mathit{fn}(P) \cup \mathit{fn}(\varphi)$ we can always choose an appropriate m and in this

case also $P \equiv \nu m.P'$ holds. Hence the only possibility for $P \models_S \neg\, \text{res}\ a\ \varphi$ to hold is $P' \not\models_S \varphi\{a \leftarrow m\}$ which is by the semantics equivalent to $P' \models_S \neg\varphi\{a \leftarrow m\}$. By the definition of substitution application, this is true if and only if $P' \models_S (\neg\varphi)\{a \leftarrow m\}$. Hence the equivalence to $P \models_S \text{res}\ a\ (\neg\varphi)$. $\square$

The interaction of $\top$ with quantification of restricted names is shown in Lemma 3.3.9. If we have a restriction quantification over a parallel composition with $\top$, we can compose this formula in parallel with $\top$ and the other way round. Intuitively, this holds, since the processes satisfying $\top$ can contain the restricted name free or not, and we can extrude the scope of the restriction appropriately.

Lemma 3.3.9. *Let φ be a PSTL-formula and $a \in \mathcal{A}$. Then*

$$\text{res}\ a\ (\varphi \between \top) \quad \widehat{=} \quad \text{res}\ a\ (\varphi \between \top) \between \top.$$

Proof *of Lemma 3.3.9*
Let $P \models_S \text{res}\ a\ (\varphi \between \top)$, i.e.,

$$P \equiv \nu m.P' \text{ and } m \notin \mathit{fn}(P) \cup \mathit{fn}(\varphi \between \top) \text{ and } P' \models_S (\varphi \between \top)\{a \leftarrow m\}.$$

Hence $P' \equiv Q \,|\, R$ with $Q \models_S \varphi\{a \leftarrow m\}$ and $R \models_S \top$.

1. If $m \notin \mathit{fn}(R)$, we get by (Scope-Extr) that

$$P \equiv \nu m.Q \,|\, R \equiv \nu m.(Q \,|\, \mathbf{0}) \,|\, R.$$

   Now let $Q' \equiv \nu m.(Q \,|\, \mathbf{0})$. Then $m \notin \mathit{fn}(Q')$ since $\mathit{fn}(Q') \subseteq \mathit{fn}(P)$, consequently $Q' \models_S \text{res}\ a\ (\varphi \between \top)$. Therefore $P \models_S \text{res}\ a\ (\varphi \between \top) \between \top$.

2. If $m \in \mathit{fn}(R)$, we can use (Neutr-Par) to get $P \equiv P \,|\, \mathbf{0}$. Since $\mathbf{0} \models_S \top$, and $P \models_S \text{res}\ a\ (\varphi \between \top)$ we conclude $P \models_S \text{res}\ a\ (\varphi \between \top) \between \top$.

Conversely let $P \models_S \text{res}\ a\ (\varphi \between \top) \between \top$. Then $P \equiv Q \,|\, R$ such that $Q \models_S \text{res}\ a\ (\varphi \between \top)$ and $R \models_S \top$. Hence $Q \equiv \nu m.(Q' \,|\, Q'')$ where $m \notin \mathit{fn}(Q) \cup \mathit{fn}(\varphi \between \top)$, $Q' \models_S \varphi\{a \leftarrow m\}$ and $Q'' \models_S \top$.

1. Let $m \notin \mathit{fn}(R)$. Then by (Scope-Extr) $P \nu m.(Q' \,|\, Q'' \,|\, R)$. Since $Q'' \,|\, R \models_S \top$, we get $P \models_S \text{res}\ a\ (\varphi \between \top)$.

2. If $m \in fn(R)$, we choose an $m' \notin fn(R)$ and use alpha-conversion which yields

$$P \equiv \nu m'.(Q'\{m'/m\} \mid Q''\{m'/m\}) \mid R.$$

Now we can apply (Scope-Extr) to get

$$P \equiv \nu m'.(Q'\{m'/m\} \mid Q''\{m'/m\} \mid R).$$

By Lemma 3.1.2, we get that $Q'\{m'/m\} \models_S \varphi\{a \leftarrow m\}\{m \leftarrow m'\}$ which is equivalent to $Q'\{m'/m\} \models_S \varphi\{a \leftarrow m'\}$. Because $Q''\{m'/m\} \mid R \models_S \top$, we conclude $P \models_S \text{res } a\ (\varphi \between \top)$.

$\square$

4 Complex atoms for simple reasoning

In this chapter, we present a new type of structural atoms, *identificators*, which will simplify the reasoning with PSTL. We denote the identificators like process identifiers with capital letters $K, L, \ldots$. The general idea behind these identificators is that a process P satisfies the identificator K if and only if there is a process equation $K(\tilde{x}) := P_K$ and P is derived from P_K. The intuition would be that K is the name of a procedure in a program, P_K is the program code and P is the actual location within the procedure during its execution.

To achieve this result, we compute a new process P' bisimilar to P, which contains information about its caller. Therefore, we have to make some assumptions about the initially defined processes.

Assumption: Every process is either structurally congruent to $\mathbf{0}$, or does not contain $\mathbf{0}$. This is no restriction on the possible process. E.g. consider the process $x(y).\mathbf{0}$. We introduce a new process identifier D with the defining equation $D := \mathbf{0}$ and transform the initial process to $x(y).D$. This transformation can be applied to all processes.

Assumption: The evolution of the process under consideration starts with a special process identifier, to which we refer as SYS.

Assumption: The process identifiers of all reachable processes constitute the set $\mathcal{I}$.

4.1 Tagging Process Calls

With the assumptions given in the introduction of this chapter, we construct a new system of process equations by adding a superscript that refers to the caller of the equation to every process identifier occurring in process equation definitions.

We therefore employ a renaming function $ren_K \colon \mathcal{P} \to \mathcal{P}$, such that $ren_K(P)$ adds K as a superscript to every process identifier occurring in P.

Definition 4.1.1. For every process identifier K, the function $ren_K \colon \mathcal{P} \to \mathcal{P}$ is defined by

$$\begin{array}{rcl} ren_K(L) & = & L^K \\ ren_K(L\lfloor\tilde{a}\rfloor) & = & ren_K(L)\lfloor\tilde{a}\rfloor \\ ren_K(\pi.P) & = & \pi.ren_K(P) \\ ren_K(M+N) & = & ren_K(M)+ren_K(N) \\ ren_K(P\,|\,Q) & = & ren_K(P)\,|\,ren_K(Q) \\ ren_K(\nu a.P) & = & \nu a.ren_K(P) \end{array}$$

Lemma 4.1.1 shows the independence of renaming and application of substitution on processes, i.e., the order of renaming and substitution is unimportant. We will need this result for proving the bisimilarity of the original and the renamed processes stated in Proposition 4.1.1.

Lemma 4.1.1. *The renaming function is compatible with the application of substitutions, i.e., for all substitutions σ, $P \in \mathcal{P}$ and process identifiers K,*

$$ren_K(P)\sigma = ren_K(P\sigma).$$

Proof *of Lemma 4.1.1*
We use induction according to the structure of processes.
Base Case $L\lfloor\tilde{a}\rfloor$:

$$\begin{array}{rcl} & & ren_K(L\lfloor\tilde{a}\rfloor)\sigma \\ \{\text{Definition of } ren_K\} & = & ren_K(L)\lfloor\tilde{a}\rfloor\sigma \\ \{\text{Definition of } ren_K\} & = & L^K\lfloor\tilde{a}\rfloor\sigma \\ \{\text{Application of substitution}\} & = & L^K\lfloor\sigma(\tilde{a})\rfloor \\ \{\text{Definition of } ren_K\} & = & ren_K(L)\lfloor\sigma(\tilde{a})\rfloor \\ \{\text{Definition of } ren_K\} & = & ren_K(L\lfloor\sigma(\tilde{a})\rfloor) \end{array}$$

For the induction step, we assume the lemma holds for the processes P and Q and the sums M and N.
Case $\pi.P$:
We examine this case for $\pi = \overline{x}\langle z\rangle$, since the cases $\pi = x(y)$ and $\pi = \tau$ can be

handled similarly.

$$\begin{array}{rcl}
 & & ren_K(\overline{x}\langle z\rangle.P)\sigma \\
\{\text{Definition of } ren_K\} & = & (\overline{x}\langle z\rangle.ren_K(P))\sigma \\
\{\text{Application of substitution}\} & = & \overline{\sigma(x)}\langle\sigma(z\rangle.(ren_K(P)\sigma) \\
\{\text{Induction Hypothesis}\} & = & \overline{\sigma(x)}\langle\sigma(z)\rangle.ren_K(P\sigma) \\
\{\text{Definition of } ren_K\} & = & ren_K(\overline{\sigma(x)}\langle\sigma(z)\rangle.(P\sigma)) \\
\{\text{Application of substitution}\} & = & ren_K((\overline{x}\langle z\rangle.P)\sigma)
\end{array}$$

Case $M + N$ **and** $P \mid Q$**:**
Since ren_K and the application of substitution are defined similar for $M + N$ and $P \mid Q$, it is sufficient to examine one of these cases. We show the lemma for $M+N$.

$$\begin{array}{rcl}
 & & ren_K(M+N)\sigma \\
\{\text{Definition of } ren_K\} & = & (ren_K(M) + ren_K(N))\sigma \\
\{\text{Application of substitution}\} & = & ren_K(M)\sigma + ren_K(N)\sigma \\
\{\text{Induction hypothesis}\} & = & ren_K(M\sigma) + ren_K(N\sigma) \\
\{\text{Definition of } ren_K\} & = & ren_K(M\sigma + N\sigma) \\
\{\text{Application of substitution}\} & = & ren_K((M+N)\sigma)
\end{array}$$

Case $\nu a.P$**:**

$$\begin{array}{rcl}
 & & ren_K(\nu a.P)\sigma \\
\{\text{Definition of } ren_K\} & = & (\nu a.ren_K(P))\sigma \\
\{\text{Application of substitution}\} & = & \nu a.(ren_K(P))\sigma \\
\{\text{Induction Hypothesis}\} & = & \nu a.ren_K(P\sigma) \\
\{\text{Definition of } ren_K\} & = & ren_K(\nu a.(P\sigma)) \\
\{\text{Application of substitution}\} & = & ren_K((\nu a.P)\sigma)
\end{array}$$

□

That renaming preserves the internal structure of a process is formulated in Lemma 4.1.2. It shows that two structural congruent processes remain congruent after the application of ren_K to both processes. This result is not surprising, since renaming is homomorphic on the operators of the π-Calculus. Nevertheless we explicitly prove the statement because we need it for the proof of Proposition 4.1.1.

Lemma 4.1.2. *The renaming function is compatible with structural congruence. For all processes $P, Q \in \mathcal{P}$ and all process identifiers K, the following equivalence holds:*

$$P \equiv Q \textit{ iff } ren_K(P) \equiv ren_K(Q).$$

Proof *of Lemma 4.1.2*
The right to left direction is immediate by dropping all superscripts of the process identifiers of $ren_K(P)$ and $ren_K(Q)$.

For the other direction, we proceed by induction on the derivatives of structural congruence. Most of the base cases are simple applications of the definition of ren_K.

Base Case $P \equiv P \,|\, \mathbf{0}$ and $\nu a.\mathbf{0} \equiv \mathbf{0}$:
The proofs for the laws concerning the process $\mathbf{0}$ are straightforward applications of Definition 4.1.1.

$$\begin{aligned} &ren_K(P \,|\, \mathbf{0}) = ren_K(P) \,|\, ren_K(\mathbf{0}) = ren_K(P) \,|\, \mathbf{0} \equiv ren_K(P) \\ &ren_K(\nu a.\mathbf{0}) = \nu a.ren_K(\mathbf{0}) = \nu a.\mathbf{0} \equiv \mathbf{0} = ren_K(\mathbf{0}) \end{aligned}$$

Base Case $P \,|\, Q \equiv Q \,|\, P$ and $M + N \equiv N + M$:
We only consider the law $P \,|\, Q \equiv Q \,|\, P$, since the proof for the commutative law of the choice operator is similar.

$$ren_K(P \,|\, Q) = ren_K(P) \,|\, ren_K(Q) \equiv ren_K(Q) \,|\, ren_K(P) = ren_K(Q \,|\, P)$$

Base Case $(P \,|\, Q) \,|\, R \equiv P \,|\, (Q \,|\, R)$ and $(M + N) + O \equiv M + (N + O)$:
Again we refrain from proving the law for the choice operator due to its similarity to the proof given below.

$$\begin{aligned} & ren_K((P \,|\, Q) \,|\, R) \\ = \;& ren_K((P \,|\, Q)) \,|\, ren_K(R) \\ = \;& (ren_K(P) \,|\, ren_K(Q)) \,|\, ren_K(R) \\ \equiv \;& ren_K(P) \,|\, (ren_K(Q) \,|\, ren_K(R)) \\ = \;& ren_K(P) \,|\, ren_K(Q \,|\, R) \\ = \;& ren_K(P \,|\, (Q \,|\, R)) \end{aligned}$$

Base Case $\nu a.(P \,|\, Q) \equiv P \,|\, \nu a.Q$, if $a \notin fn(P)$:
For this law, we have to note, that $fn(P) = fn(ren_K(P))$. This is true, since ren_K

does not change any names of the renamed process.

$$\begin{array}{rcl} & & ren_K(\nu a.(P \mid Q)) \\ & = & \nu a.ren_K(P \mid Q) \\ & = & \nu a.(ren_K(P) \mid ren_K(Q)) \\ \{fn(P) = fn(ren_K(P))\} & \equiv & ren_K(P) \mid \nu a.ren_K(Q) \\ & = & ren_K(P) \mid ren_K(\nu a.Q) \\ & = & ren_K(P \mid \nu a.Q) \end{array}$$

Base Case $\nu a.\nu b.P \equiv \nu b.\nu a.P$:
This proof is again a straightforward application of Definition 4.1.1 and the standard laws of structural congruence.

$$\begin{array}{rl} & ren_K(\nu a.\nu b.P) \\ = & \nu a.ren_K(\nu b.P) \\ = & \nu a.\nu b.ren_K(P) \\ \equiv & \nu b.\nu a.ren_K(P) \\ = & \nu b.ren_K(\nu a.P) \\ = & ren_K(\nu b.\nu a.P) \end{array}$$

For the induction step, assume the Lemma holds for the processes $P, Q \in \mathcal{P}$. We will not show the proofs for reflexivity, symmetry and transitivity.
Case $P \equiv Q$ implies $\pi.P + M \equiv \pi.Q + M$:

$$\begin{array}{rcl} & & ren_K(\pi.P + M) \\ & = & ren_K(\pi.P) + ren_K(M) \\ & = & \pi.ren_K(P) + ren_K(M) \\ \{\text{Induction Hypothesis}\} & \equiv & \pi.ren_K(Q) + ren_K(M) \\ & = & ren_K(\pi.Q) + ren_K(M) \\ & = & ren_K(\pi.P + M) \end{array}$$

Case $P \equiv Q$ implies $P \mid R \equiv Q \mid R$:

$$\begin{array}{rcl} & & ren_K(P \mid R) \\ & = & ren_K(P) \mid ren_K(R) \\ \{\text{Induction Hypothesis}\} & \equiv & ren_K(Q) \mid ren_K(R) \\ & = & ren_K(P \mid Q) \end{array}$$

Case $P \equiv Q$ implies $\nu a.P \equiv \nu a.Q$:

$$\begin{array}{rcl} & & ren_K(\nu a.P) \\ & = & \nu a.ren_K(P) \\ \{\text{Induction Hypothesis}\} & \equiv & \nu a.ren_K(Q) \\ & = & ren_K(\nu a.Q) \end{array}$$

In summary, we have shown that $P \equiv Q$ iff $ren_K(P) \equiv ren_K(Q)$. □

Now we define, how a given set of process equations shall be renamed in order to use identificators as formulas. For a process P with the defining equations $K_i(\tilde{x}) := P_i$ where $1 \leq i \leq n$ and $K_i \in \mathcal{I}$, we create new process equations for every process identifier K_i and $L \in \mathcal{I}$ by simultaneously adding L as a superscript to $K_i(\tilde{x})$ and renaming P_i by K_i. This models that the process P_i has been called by K_i.

Definition 4.1.2 (Tagged Process System)**.** Let $E ::= \{K_i(\tilde{x}) := P_i \mid K_i \in \mathcal{I}\}$. The *tagged process system* is defined by

$$E' ::= \{{K_i}^L(\tilde{x}) := ren_{K_i}(P_i) \mid K_i(\tilde{x}) := P_i \in E \text{ and } K_i, L \in \mathcal{I}\}$$

The initial process of the tagged process system is $\mathrm{SYS}^{\mathrm{SYS}}$.

Tagging a process system increases the number of process equation significantly. If the number of process equations in E is n, then the size of E' is n^n. But normally many of the created equations will not be reachable from the initial process.

To show that tagging a system of processes does not change the behaviour of the processes, we define a suitable bisimulation $\mathcal{R}$.

Proposition 4.1.1. *For $m, n \in \mathbb{N}$ and $K_j \in \mathcal{I}$ for $1 \leq j \leq m$, the relation $\mathcal{R}$ defined by*

$$(P, Q) \in \mathcal{R} \text{ iff } \left\{ \begin{array}{lcl} P & \equiv & \nu\tilde{a}.(P_1^{\neq\nu} \mid \ldots \mid P_n^{\neq\nu}) \\ Q & \equiv & \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \mid \ldots \mid ren_{K_m}(P_n^{\neq\nu})) \end{array} \right.$$

is a bisimulation.

Proof *of Proposition 4.1.1*
We use that P is structurally congruent to its standard form P^{sf}. Because of Lemma 4.1.2, we only consider the possible reactions of P^{sf}. Furthermore, with reaction rule (Struct), the reactions of P are the same as the reactions of P^{sf} and by Meyer [Mey08, Proposition 2.1.2] we can cut the possibilities for reactions $P^{sf} \to P'$ down to three.

1. A process in P^{sf} consumes a τ prefix.
2. Two process in P^{sf} communicate with each other.
3. A process call in P^{sf} is expanded.

We only show the first case, since the other two are similar.

$$\begin{array}{lcl} P & \equiv & \nu\tilde{a}.(P_1^{\neq\nu} \mid \ldots \mid M + \tau.P_i' + N \mid \ldots \mid P_n^{\neq\nu}) \\ & \to & \nu\tilde{a}.(P_1^{\neq\nu} \mid \ldots \mid P_i' \mid \ldots \mid P_n^{\neq\nu}) \\ & \equiv & P' \end{array}$$

The process P_i' is congruent to a process in standard form.

$$P_i' \equiv \nu\tilde{b}.(R_1^{\neq\nu} \mid \ldots \mid R_k^{\neq\nu})$$

Since all bound names of P are different and $bn(P) \cap fn(P) = \emptyset$, we can apply scope extrusion.

$$\begin{array}{lcl} P' & \equiv & \nu\tilde{a}.(P_1^{\neq\nu} \mid \ldots \mid P_i' \mid \ldots \mid P_n^{\neq\nu}) \\ & \equiv & \nu\tilde{a}.(P_1^{\neq\nu} \mid \ldots \mid \nu\tilde{b}.(R_1^{\neq\nu} \mid \ldots \mid R_k^{\neq\nu}) \mid \ldots \mid P_n^{\neq\nu}) \\ & \equiv & \nu\tilde{a}\tilde{b}.(P_1^{\neq\nu} \mid \ldots \mid R_1^{\neq\nu} \mid \ldots \mid R_k^{\neq\nu} \mid \ldots \mid P_n^{\neq\nu}) \\ & \equiv & \nu\tilde{a}\tilde{b}.(P_1^{\neq\nu} \mid \ldots \mid \Pi_{j=1}^{k} R_j^{\neq\nu} \mid \ldots \mid P_n^{\neq\nu}) \end{array}$$

Now we consider Q with $(P,Q) \in \mathcal{R}$, i.e.

$$\begin{aligned}
Q &\equiv \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \,|\ldots|\, ren_{K_i}(M + \tau.P_i' + N) \,|\ldots|\, ren_{K_m}(P_n^{\neq\nu}))\\
&= \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \,|\ldots|\, ren_{K_i}(M) + \tau.ren_{K_i}(P_i') + ren_{K_i}(N) \,|\ldots|\, ren_{K_m}(P_n^{\neq\nu}))\\
&\to \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \,|\ldots|\, ren_{K_i}(P_i') \,|\ldots|\, ren_{K_m}(P_n^{\neq\nu}))\\
&\equiv Q'
\end{aligned}$$

We examine $ren_{K_i}(P_i')$.

$$\begin{aligned}
ren_{K_i}(P_i') &\equiv ren_{K_i}(\nu\tilde{b}.(R_1^{\neq\nu} \mid \ldots \mid R_k^{\neq\nu}))\\
&= \nu\tilde{b}.(ren_{K_i}(R_1^{\neq\nu} \mid \ldots \mid R_k^{\neq\nu}))\\
&= \nu\tilde{b}.(ren_{K_i}(R_1^{\neq\nu}) \mid \ldots \mid ren_{K_i}(R_k^{\neq\nu}))\\
&= \nu\tilde{b}.(\Pi_{j=1}^{k} ren_{K_i}(R_j^{\neq\nu}))
\end{aligned}$$

Hence, we get that

$$\begin{aligned}
Q' &\equiv \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \mid \ldots \mid ren_{K_i}(P_i') \mid \ldots \mid ren_{K_m}(P_n^{\neq\nu}))\\
&\equiv \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \mid \ldots \mid \nu\tilde{b}.(\Pi_{j=1}^{k} ren_{K_i}(R_j^{\neq\nu})) \mid \ldots \mid ren_{K_m}(P_n^{\neq\nu}))
\end{aligned}$$

By application of scope extrusion we get

$$\begin{aligned}
Q' &\equiv \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \mid \ldots \mid \nu\tilde{b}.(\Pi_{j=1}^{k} ren_{K_i}(R_j^{\neq\nu})) \mid \ldots \mid ren_{K_m}(P_n^{\neq\nu}))\\
&\equiv \nu\tilde{a}\tilde{b}.(ren_{K_1}(P_1^{\neq\nu}) \mid \ldots \mid \Pi_{j=1}^{k} ren_{K_i}(R_j^{\neq\nu})) \mid \ldots \mid ren_{K_m}(P_n^{\neq\nu})).
\end{aligned}$$

Hence $(P', Q') \in \mathcal{R}$.

Now we show that $Q \to Q'$ and $(P,Q) \in \mathcal{R}$ implies the existence of a process P' with $P \to P'$ and $(P', Q') \in \mathcal{R}$. Again we show the case for a reaction due to a τ prefix.

$$\begin{aligned}
Q &\equiv \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \,|\ldots|\, ren_{K_i}(M) + \tau.ren_{+}(P_i')K_i ren_{K_i}(N) \,|\ldots|\, ren_{K_m}(P_n^{\neq\nu}))\\
&\to \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \,|\ldots|\, ren_{K_i}(P_i') \,|\ldots|\, ren_{K_m}(P_n^{\neq\nu}))\\
&\equiv Q'
\end{aligned}$$

Because

$$\begin{aligned}
Q &\equiv \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \,|\ldots|\, ren_{K_i}(M) + \tau.ren_{K_i}(P_i') + ren_{K_i}(N) \,|\ldots|\, ren_{K_m}(P_n^{\neq\nu}))\\
&= \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \,|\ldots|\, ren_{K_i}(M + \tau.P_i' + N) \,|\ldots|\, ren_{K_m}(P_n^{\neq\nu})),
\end{aligned}$$

we get that

$$\begin{aligned} P &\equiv \nu\tilde{a}.(P_1^{\neq\nu} \mid \ldots \mid M + \tau.P_i' + N \mid \ldots \mid P_n^{\neq\nu}) \\ &\rightarrow \nu\tilde{a}.(P_1^{\neq\nu} \mid \ldots \mid P_i' \mid \ldots \mid P_n^{\neq\nu}) \\ &\equiv P' \end{aligned}$$

To show that $(P', Q') \in \mathcal{R}$, we apply similar arguments as above. The process Q' is structurally congruent to its standard form, i.e.

$$\begin{aligned} Q' &\equiv \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \mid \ldots \mid ren_{K_i}(P_i') \mid \ldots \mid ren_{K_m}(P_n^{\neq\nu})) \\ &\equiv \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \mid \ldots \mid ren_{K_i}(\nu\tilde{b}.(\Pi_{j=1}^{k} R_j^{\neq\nu})) \mid \ldots \mid ren_{K_m}(P_n^{\neq\nu})) \\ &= \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \mid \ldots \mid \nu\tilde{b}.ren_{K_i}(\Pi_{j=1}^{k} R_j^{\neq\nu}) \mid \ldots \mid ren_{K_m}(P_n^{\neq\nu})) \\ &= \nu\tilde{a}.(ren_{K_1}(P_1^{\neq\nu}) \mid \ldots \mid \nu\tilde{b}.\Pi_{j=1}^{k} ren_{K_i}(R_j^{\neq\nu}) \mid \ldots \mid ren_{K_m}(P_n^{\neq\nu})) \\ &\equiv \nu\tilde{a}\tilde{b}.(ren_{K_1}(P_1^{\neq\nu}) \mid \ldots \mid \Pi_{j=1}^{k} ren_{K_i}(R_j^{\neq\nu}) \mid \ldots \mid ren_{K_m}(P_n^{\neq\nu})), \end{aligned}$$

where the standard form of P_i' is

$$P_i'^{sf} = \nu\tilde{b}.(\Pi_{j=1}^{k} R_j^{\neq\nu}).$$

Since all bound names of P' are different and $bn(P') \cap fn(P') = \emptyset$ we can again apply scope extrusion.

$$\begin{aligned} P' &\equiv \nu\tilde{a}.(P_1^{\neq\nu} \mid \ldots \mid P_i' \mid \ldots \mid P_n^{\neq\nu}) \\ &\equiv \nu\tilde{a}.(P_1^{\neq\nu} \mid \ldots \mid \nu\tilde{b}.(\Pi_{j=1}^{k} R_j^{\neq\nu}) \mid \ldots \mid P_n^{\neq\nu}) \\ &\equiv \nu\tilde{a}\tilde{b}.(P_1^{\neq\nu} \mid \ldots \mid \Pi_{j=1}^{k} R_j^{\neq\nu} \mid \ldots \mid P_n^{\neq\nu}) \end{aligned}$$

We conclude with $(P', Q') \in \mathcal{R}$.

The other cases are similar except for an additional application of Lemma 4.1.1.

Hence $\mathcal{R}$ is a bisimulation. In particular $(\mathrm{SYS}, ren_{\mathrm{SYS}}(\mathrm{SYS})) \in \mathcal{R}$, i.e., the original process SYS and the new initial process $ren_{\mathrm{SYS}}(\mathrm{SYS})$ are bisimilar. $\square$

Example 4.1.1. Consider the Client/Server-example of Section 2.3 (Example 2.3.1) together with the initial process $\mathrm{SYS} := C\lfloor url \rfloor \mid C\lfloor url \rfloor \mid S\lfloor url \rfloor$. The set of identifiers is $\mathcal{I} = \{\mathrm{SYS}, C, S\}$, so the tagged process system is

$$\begin{array}{rcl}
\mathrm{SYS}^{\mathrm{SYS}} & := & C^{\mathrm{SYS}}\lfloor url \rfloor \mid C^{\mathrm{SYS}}\lfloor url \rfloor \mid S^{\mathrm{SYS}}\lfloor url \rfloor \\
\mathrm{SYS}^{C} & := & C^{\mathrm{SYS}}\lfloor url \rfloor \mid C^{\mathrm{SYS}}\lfloor url \rfloor \mid S^{\mathrm{SYS}}\lfloor url \rfloor \\
\mathrm{SYS}^{S} & := & C^{\mathrm{SYS}}\lfloor url \rfloor \mid C^{\mathrm{SYS}}\lfloor url \rfloor \mid S^{\mathrm{SYS}}\lfloor url \rfloor \\
C^{\mathrm{SYS}}(url) & := & \nu ip.\overline{url}\langle ip \rangle.ip(s).s(x).C^{C}\lfloor url \rfloor \\
C^{C}(url) & := & \nu ip.\overline{url}\langle ip \rangle.ip(s).s(x).C^{C}\lfloor url \rfloor \\
C^{S}(url) & := & \nu ip.\overline{url}\langle ip \rangle.ip(s).s(x).C^{C}\lfloor url \rfloor \\
S^{\mathrm{SYS}}(url) & := & url(y).\nu ses.\overline{y}\langle ses \rangle.\overline{ses}\langle ses \rangle.S^{S}\lfloor url \rfloor \\
S^{C}(url) & := & url(y).\nu ses.\overline{y}\langle ses \rangle.\overline{ses}\langle ses \rangle.S^{S}\lfloor url \rfloor \\
S^{S}(url) & := & url(y).\nu ses.\overline{y}\langle ses \rangle.\overline{ses}\langle ses \rangle.S^{S}\lfloor url \rfloor
\end{array}$$

On closer examination, we see that no call to $\mathrm{SYS}^{C}, \mathrm{SYS}^{S}, C^{S}$ and S^{C} ever occurs, so these equations can be safely omitted.

4.2 Semantics

To define the semantics of identificators, we employ the function $caller\colon \mathcal{P} \to \mathbb{P}(\mathcal{I})$. It collects all process identifiers that called a process, i.e., that are superscripts.

Definition 4.2.1 (Calling Processes)**.** The function $caller\colon \mathcal{P} \to \mathbb{P}(\mathcal{I})$ returning for a process P the identifier K with $K\lfloor \tilde{a} \rfloor \to P_K\{\tilde{a}/\tilde{x}\} \to^* P$, such that no other process call occurs in the reactions of P' to P, is defined as follows.

$$\begin{array}{rclrcl}
caller(L^K) & = & \{K\} & caller(L^K\lfloor \tilde{a} \rfloor) & = & caller(L^K) \\
caller(\pi.P) & = & caller(P) & caller(\nu a.P) & = & caller(P) \\
caller(M + N) & = & caller(M) \cup caller(N) & & & \\
caller(P \mid Q) & = & caller(P) \cup caller(Q) & & &
\end{array}$$

With the help of this function, the semantics of a process identificator K can be defined. Observe that we restrict the process satisfying an identificator to sequential processes. We exploit this while proving the soundness of the translation of structural formulas, in particular in the proof of Lemma 5.1.2.

Definition 4.2.2 (Semantics)**.** For a process P and an identificator K, the consequence relation is given by

$$P \models_S \quad K \quad \text{iff } caller(P) = \{K\} \text{ and } P \in \mathcal{P}^{\mathrm{seq}}$$

Example 4.2.1. Consider the tagged process system of the client/server example (see Example 4.1.1). Then

$$\nu ip.\overline{url}\langle ip\rangle.ip(s).s(x).C^{C}\lfloor url\rfloor \models_S C \text{ and } s(x).C^{C}\lfloor url\rfloor \models_S C$$

since $caller(\nu ip.\overline{url}\langle ip\rangle.ip(s).s(x).C^{C}\lfloor url\rfloor) = \{C\}$.

5 Translation to LTL

The aim of this chapter is to define a translation from a subset of PSTL to LTL on Petri nets. We present a type of formulas resembling fragments and processes in restricted form (see Definition 2.4.1). The translation is defined relative to the process P of interest and is done on two levels:

For structural formulas φ_S, we collect the congruence classes of all minimal processes satisfying φ_S in a set $[\![\varphi_S]\!]_P$. Then we consider all congruence classes $[Q] \in [\![\varphi_S]\!]_P$ and count the occurrences c of each fragment F of a representative process $Q_{rf} \in [Q]$ via the decomposition function. Out of these occurrences, we build a conjunction of atomic formulas of LTL, i.e., a formula $\psi_Q = \bigwedge [F_i] = c_i$ or $\psi_Q = \bigwedge [F_i] \geq c_i$. The complete translated formula is the disjunction of all the ψ_Q.

That we only collect the minimal processes satisfying φ_S is due to the possibility that $\top$ may be contained in φ_S. In this case, there is no limit for the size of the processes satisfying φ_S, and $[\![\varphi_S]\!]_P$ would be infinite, hence the translated formula would also be infinite.

The translation of temporal formulas is considerably simpler, since the temporal part of PSTL imitates LTL. The atoms, i.e., the structural formulas are translated as described above, the Boolean connectors and the temporal modalities are translated into their counterpart of LTL.

The relations between the structural and temporal translation function θ_S resp. θ are shown in Figure 5.1.

$$\begin{tikzcd}
\mathcal{PSL} \ni \Phi_{\mathcal{S}} \arrow[r, "\theta_S"', ""{name=A}] & \text{DNF on } p_i \sim c_i \\
\mathcal{PSTL} \ni \Phi_{\mathcal{T}} \arrow[r, "\theta", ""'{name=B}] & \text{Temporal Formula} \arrow[u, dashed, "\text{based on}"']
\arrow[from=B, to=A, dashed, "\text{uses}"']
\end{tikzcd}$$

Figure 5.1: Connections between the translation functions θ_S and θ. The relation $\sim$ is either equality $=$ or the inequality $\geq$.

5.1 Translating structural modalities to LTL

We start with the translation of formulas without temporal modalities. The translated formulas will only specify a single marking M, i.e. a certain state of the Petri nets.

The formulas which are translatable are rather strongly restricted. For example, we do not allow for any occurrence of negation in the formulas. Since we do allow for disjunction, we are not able to treat this Boolean operator as an abbreviation any more. Hence for giving a complete semantics to the translatable formulas, we define the consequence relation for the disjunction.

Definition 5.1.1 (Semantics of Disjunction)**.** For two structural formulas φ_1, φ_2 and a process P, the consequence relation for the disjunction of φ_1 and φ_2 is given by

$$P \models_S \varphi_1 \vee \varphi_2 \quad \text{iff } P \models_S \varphi_1 \text{ or } P \models_S \varphi_2$$

To ensure that we essentially specify fragments with the formulas of Definition 5.1.3, we define a subset of the free names of a formula, the *ensured free names*. This set only contains free names that occur in *all* parts of a conjunction (and disjunction) of such formulas, i.e., different from the union of free names of conjuncts (resp. disjuncts) it returns the intersection of the sets of free names.

Definition 5.1.2 (Ensured Free Names)**.** The *ensured free names* of a fragmentary formula (see Definition 5.1.3) is defined recursively by

$$\begin{aligned}
\mathit{efn}(P) &= \mathit{fn}(P)\\
\mathit{efn}(K \wedge \mathit{free}(\tilde{a})) &= \{\tilde{a}\}\\
\mathit{efn}(\tau_1 \vee \tau_2) &= \mathit{efn}(\tau_1) \cap \mathit{efn}(\tau_2)\\
\mathit{efn}(\text{res } a\ (\phi_1 \between \cdots \between \phi_n)) &= (\mathit{efn}(\phi_1) \cup \cdots \cup \mathit{efn}(\phi_n)) \setminus \{a\}\\
\mathit{efn}(\text{res } a\ (\phi_1 \between \cdots \between \phi_n \between \top)) &= \mathit{efn}(\text{res } a\ (\phi_1 \between \cdots \between \phi_n))\\
\mathit{efn}(\phi_1) \vee \mathit{efn}(\phi_2) &= \mathit{efn}(\phi_1) \cap \mathit{efn}(\phi_2)
\end{aligned}$$

Remark: Obviously, $\mathit{efn}(\phi) \subseteq \mathit{fn}(\phi)$. $\diamond$

Now we give the definition of fragmentary formulas. A fragmentary formula ϕ specifies a single fragment if it does not contain $\top$, because we only allow for restrictions on names that occur in all subformulas of ϕ. However, if a fragmentary formula ϕ contains $\top$ we can at least ensure that every process that satisfies ϕ also contains a single fragment satisfying ϕ.

Definition 5.1.3 (Sequential and Fragmentary Formulas)**.**

$$\begin{aligned}
\tau &::= Q \mid K \wedge \mathit{free}(\tilde{a}) \mid \tau_1 \vee \tau_2 \\
\phi &::= \tau \mid \mathrm{res}\ a\ (\phi_1 \between \cdots \between \phi_n) \mid \mathrm{res}\ a\ (\phi_1 \between \cdots \between \phi_n \between \top) \mid \phi_1 \vee \phi_2
\end{aligned}$$

where $a \in \mathit{efn}(\phi_i)$ and $Q \in \mathcal{P}^{\mathrm{seq}}$. We call the formulas τ *sequential formulas* and the formulas ϕ *fragmentary formulas.*

Remark: The abbreviation res* given in Definition 3.1.8 is based on the free names of a formula. Since we only allow for restrictions on ensured names, using the abbreviation would create formulas not well-formed with respect to Definition 5.1.3. Therefore, we further restrict the abbreviation to work only on the ensured free names and mark this restriction by the index *rf*, i.e.

$$\mathrm{res}^*_{\mathit{rf}}\ \phi \overset{\mathrm{def}}{=} \bigvee_{\tilde{n} \subseteq \mathit{efn}(\phi)} \mathrm{res}\ \tilde{n}\ \phi$$

Furthermore, we define the *restricted somewhere* operator, which relies on $\mathrm{res}^*_{\mathit{rf}}$ as

$$\Diamond_{\mathit{rf}}\phi \overset{\mathrm{def}}{=} \mathrm{res}^*_{\mathit{rf}}(\phi \between \top).$$

We can not define a restricted operator corresponding to $\boxminus$, since we do not allow for negation in the formulas. $\Diamond$

The following Lemma shows, that *ensured free names* is indeed a reasonable name for this set, since every name contained in $\mathit{efn}(\phi)$ is also a free name of the processes satisfying ϕ.

Lemma 5.1.1. *Let P be a process, ϕ a fragmentary formula and x a name so that $P \models_S \phi$ and $x \in \mathit{efn}(\phi)$. Then $x \in \mathit{fn}(P)$.*

Proof *of Lemma 5.1.1*
We prove the implication by induction on the structure of fragmentary formulas.
Base Case $P \models_S Q$ with $Q \in \mathcal{P}^{\mathbf{seq}}$ and $x \in \mathit{efn}(Q)$:
Then $P \equiv Q$ and by Definition 5.1.2 $x \in \mathit{fn}(Q)$. Since structural congruence preserves free names, also $x \in \mathit{fn}(P)$.

Base Case $P \models_S K \wedge \mathit{free}(\tilde{a})$ **with** $x \in \mathit{efn}(K \wedge \mathit{free}(\tilde{a}))$**:**
Then $P \models_S K$ and $P \models_S \mathit{free}(\tilde{a})$, i.e., $\tilde{a} \subseteq \mathit{fn}(P)$. By Definition 5.1.2 $x \in \{\tilde{a}\}$ and hence $x \in \mathit{fn}(P)$.
Case $P \models_S \tau_1 \vee \tau_2$ **and** $x \in \mathit{efn}(\tau_1 \vee \tau_2)$**:**
Then $P \models_S \tau_1$ or $P \models_S \tau_2$ and $x \in \mathit{efn}(\tau_1) \cap \mathit{efn}(\tau_2)$, i.e., $x \in \mathit{efn}(\tau_1)$ and $x \in \mathit{efn}(\tau_2)$. Now for $i \in \{1,2\}$, if $P \models_S \tau_i$, we additionally have $x \in \mathit{efn}(\tau_i)$ and hence by induction hypothesis $x \in \mathit{fn}(P)$.
Case $P \models_S \mathrm{res}\ a\ (\phi_1 \between \cdots \between \phi_n)$**, where** $a \in \mathit{efn}(\phi_i)$ $(1 \leq i \leq n)$ **and** $x \in \mathit{efn}(\mathrm{res}\ a\ (\phi_1 \between \cdots \between \phi_n))$**:**
By Definition 5.1.2 we get $x \in (\mathit{efn}(\phi_1) \cup \cdots \cup \mathit{efn}(\phi_n)) \setminus \{a\}$, i.e. $x \neq a$. The semantics of the restriction operator and the parallel composition yields

$$P \equiv \nu m.(P_1 \mid \ldots \mid P_n),$$

where $m \notin \mathit{fn}(P) \cup \mathit{fn}(\phi_1) \cup \cdots \cup \mathit{fn}(\phi_n)$ and

$$P_1 \mid \ldots \mid P_n \models_S (\phi_1 \between \cdots \between \phi_n)\{a \leftarrow m\}.$$

Since $\mathit{efn}(\phi_i) \subseteq \mathit{fn}(\phi_i)$, we get $x \neq m$. Furthermore, by the semantics of the parallel composition and the definition of substitution,

$$P_1 \models_S \phi_1\{a \leftarrow m\} \ldots P_n \models_S \phi_n\{a \leftarrow m\}.$$

Because $x \in \mathit{efn}(\phi_1) \cup \cdots \cup \mathit{efn}(\phi_n)$, there is an $j \in \{1, \ldots, n\}$, such that $x \in \mathit{efn}(\phi_j)$. By the induction hypothesis, we get $x \in \mathit{fn}(P_j)$. And since $x \neq m$, we conclude $x \in \mathit{fn}(P)$.
Case $P \models_S \mathrm{res}\ a\ (\phi_1 \between \cdots \between \phi_n \between \top)$**:**
Similar to the previous case.
Case $P \models_S \phi_1 \vee \phi_2$**:**
Similar to the case $P \models_S \tau_1 \vee \tau_2$. $\square$

As mentioned above, fragmentary formulas specify single fragments. Furthermore, sequential formulas τ specify sequential processes. This is formalised in Lemma 5.1.2. We only consider processes in restricted form to remove parallel compositions with **0**. This is no restriction on the processes since for every P, $\mathit{rf}(P) = P_{\mathit{rf}} \equiv P$ and the consequence relation is compatible with structural congruence (refer to Lemma 2.4.1 and Lemma 3.3.1).

Lemma 5.1.2. *Let τ be a sequential formula and ϕ be a fragmentary formula, where $\top \notin \mathit{sub}(\phi)$.*

(i) Every $P_{rf} \in \mathcal{P}$ with $P_{rf} \models_S \tau$ is a sequential process.

(ii) Every $P_{rf} \in \mathcal{P}$ with $P_{rf} \models_S \phi$ is a fragment.

Proof *of Lemma 5.1.2*
We prove both statements by induction on the structure of resp. formulas.
(i)
Base Case $P_{rf} \models_S Q$ **where** $Q \in \mathcal{P}^{\mathbf{seq}}$**:**
Then $P_{rf} \equiv Q$ and hence $P_{rf} \in \mathcal{P}^{\mathrm{seq}}$.
Base Case $P_{rf} \models_S K \wedge \mathit{free}(\tilde{a})$**:**
Then $P_{rf} \models_S \mathit{free}(\tilde{a})$ and $P_{rf} \models_S K$, hence $\mathit{caller}(P_{rf}) = \{K\}$ and $P_{rf} \in \mathcal{P}^{\mathrm{seq}}$.
Case $P_{rf} \models_S \tau_1 \vee \tau_2$**:**
Then $P_{rf} \models_S \tau_1$ or $P_{rf} \models_S \tau_2$. If $P_{rf} \models_S \tau_1$, we get by induction hypothesis that $P_{rf} \in \mathcal{P}^{\mathrm{seq}}$. The case for $P_{rf} \models_S \tau_2$ is similar.
(ii)
Base Case $P_{rf} \models_S \tau$**:**
Then by (i) $P_{rf} \in \mathcal{P}^{\mathrm{seq}}$ and since $\mathcal{P}^{\mathrm{seq}} \subseteq \mathcal{P}_{\mathcal{F}}$ also $P_{rf} \in \mathcal{P}_{\mathcal{F}}$.
Case $P_{rf} \models_S \mathrm{res}\ a\ (\phi_1 \between \cdots \between \phi_n)$**, where** $a \in \mathit{efn}(\phi_i)$ **for** $1 \leq i \leq n$**:**
Then $P_{rf} \equiv \nu m.(P_1 \mid \ldots \mid P_n)$, where $m \notin \mathit{fn}(P_{rf}) \cup \mathit{fn}(\phi_1) \cup \cdots \cup \mathit{fn}(\phi_n)$ and $P_1 \mid \ldots \mid P_n \models_S (\phi_1 \between \cdots \between \phi_n)\{a \leftarrow m\}$. Hence $P_i \models_S \phi_i\{a \leftarrow m\}$ for $1 \leq i \leq n$. Since $a \in \mathit{efn}(\phi_i)$, $m \in \mathit{efn}(\phi_i\{a \leftarrow m\})$. By Lemma 5.1.1, this implies $m \in \mathit{fn}(P_i)$. Because P_{rf} is in restricted form, $P_i = P_{irf}$ and by the induction hypothesis, $P_{irf} \in \mathcal{P}_{\mathcal{F}}$. Hence we get $P_{rf} \in \mathcal{P}_{\mathcal{F}}$.
Case $P_{rf} \models_S \phi_1 \vee \phi_2$**:**
Then $P_{rf} \models_S \phi_1$ or $P_{rf} \models_S \phi_2$. If $P_{rf} \models_S \phi_i$ for $i \in \{1, 2\}$, we get by induction hypothesis that $P_{rf} \in \mathcal{P}_{\mathcal{F}}$. □

The following Lemma is weaker than Lemma 5.1.2 (ii), since it just asserts that a process with $P_{rf} \models_S \phi$ contains a fragment that satisfies ϕ. We will need both statements for the proof of Theorem 5.1.1, i.e. the soundness of the structural translation.

Lemma 5.1.3. *Let ϕ be a fragmentary formula possibly containing $\top$. Then*

$$P_{rf} \models_S \phi \quad \textit{implies} \quad P_{rf} \equiv F \mid P' \textit{ and } F \models_S \phi$$

where F is a fragment and $P' \in \mathcal{P}$.

Proof *of Lemma 5.1.3*
By induction on the structure of ϕ.

Base Case $P_{rf} \models_S \tau$:
Then we get by Lemma 5.1.2 (i) that $P_{rf} \in \mathcal{P}^{\text{seq}}$. Since $\mathcal{P}^{\text{seq}} \subseteq \mathcal{P}_{\mathcal{F}}$ and $P_{rf} \equiv P_{rf} \,|\, \mathbf{0}$, we get the desired result.
Case $P_{rf} \models_S$ res a $(\phi_1 \between \cdots \between \phi_n \between \top)$, where $a \in \mathit{efn}(\phi_i)$ for $1 \leq i \leq n$:
Then $P_{rf} \equiv \nu m.(P_1 \,|\, \ldots \,|\, P_n)$, with $m \notin \mathit{fn}(P) \cup \mathit{fn}(\phi_1) \cup \cdots \cup \mathit{fn}(\phi_n)$ and $P_i \models_S \phi_i\{a \leftarrow m\}$ for $1 \leq i \leq n$. Recall that $P_i \equiv P_{i\,rf}$. Then the induction hypothesis yields $P_{i\,rf} \equiv F_i \,|\, P_i'$, i.e.

$$\begin{aligned} P_{rf} &\equiv \nu m.(F_1 \,|\, P_1' \,|\, \ldots F_n \,|\, P_n') \\ &\equiv \nu m.(F_1 \,|\, \ldots \,|\, F_n \,|\, P_1' \,|\, \ldots \,|\, P_n'), \end{aligned}$$

and $F_i \models_S \phi_i\{a \leftarrow m\}$. By Lemma 5.1.1 $m \in \mathit{fn}(F_i)$ for $1 \leq i \leq n$.
Now consider the set $I_m \subseteq \{1, \ldots, n\}$, defined by

$$i \in I_m \quad \text{iff} \quad m \in \mathit{fn}(P_i').$$

Then by (Scope-Extr)

$$\begin{aligned} P &\equiv \nu m.(\Pi_{i=1}^{n} F_i \,|\, \Pi_{i \in I_m} P_i') \,|\, \Pi_{i \in \{1,\ldots,n\} \setminus I_m} P_i' \\ &\equiv \nu m.(\Pi_{i=1}^{n} F_i \,|\, \mathit{rf}(\Pi_{i \in I_m} P_i')) \,|\, \Pi_{i \in \{1,\ldots,n\} \setminus I_m} P_i'. \end{aligned}$$

The process $F \equiv \nu m.(\Pi_{i=1}^{n} F_i \,|\, \mathit{rf}(\Pi_{i \in I_m} P_i'))$ is a fragment and trivially $\mathit{rf}(\Pi_{i \in I_m} P_i') \models_S \top$, i.e., $F \models_S$ res a $(\phi_1 \between \cdots \between \phi_n \between \top)$ and $P \equiv F \,|\, \Pi_{i \in \{1,\ldots,n\} \setminus I_m} P_i'$.
Case $P_{rf} \models_S$ res a $(\phi_1 \between \cdots \between \phi_n)$:
For every fragmentary formula ϕ_i, we have $\top \notin \mathit{sub}(\phi_i)$, because otherwise, if $\phi_i =$ res b $(\phi_{i_1} \between \cdots \between \phi_{i_m} \between \top)$, we would get the previous case, since

$$\begin{aligned} & \text{res } a\ (\phi_1 \between \cdots \between \text{res } b\ (\phi_{i_1} \between \cdots \between \phi_{i_m} \between \top) \between \cdots \between \phi_n) \\ \widehat{\equiv}\ & \text{res } a\ (\phi_1 \between \cdots \between \text{res } b\ (\phi_{i_1} \between \cdots \between \phi_{i_m} \between \top) \between \top \between \cdots \between \phi_n). \end{aligned}$$

Lemma 5.1.2 (ii) yields that $P_{rf} \in \mathcal{P}_{\mathcal{F}}$, and by $P_{rf} \equiv P_{rf} \,|\, \mathbf{0}$, we conclude this case.
Case $P_{rf} \models_S \phi_1 \vee \phi_2$:
. Then $P_{rf} \models_S \phi_1$ or $P_{rf} \models_S \phi_2$. Let $P_{rf} \models_S \phi_1$. Then by induction hypothesis $P_{rf} \equiv F \,|\, P'$, where $F \in \mathcal{P}_{\mathcal{F}}$ and $F \models_S \phi_1$. Furthermore $F \models_S \phi_1 \vee \phi_2$. The other case is similar. □

The set of translatable formulas is based on the *restriction formulas* given in Definition 5.1.4. They resemble processes in restricted form, since they consist of parallel compositions of fragmentary formulas. Because every process satisfying a fragmentary formula, i.e., $P \models_S \phi$ is structurally congruent to a process in restricted form containing a single fragment F with $F \models_S \phi$ (Lemma 5.1.3), we get that every process $P \models_S \varrho$ with $\varrho = \phi_1 \between \cdots \between \phi_n$ is structurally congruent to $F_1 \mid \ldots \mid F_n \mid P'$, where the F_i are fragments.

Definition 5.1.4 (Restriction Formulas)**.** We call formulas ϱ defined by

$$\varrho ::= \phi_1 \between \cdots \between \phi_n$$

restriction formulas.

The fragmentary semantics defined below is the set of congruence classes of the smallest processes satisfying a restriction formula. If we choose $[Q] \in [\![\varrho]\!]_P$ and a representative in restricted form Q_{rf}, then each fragment of Q_{rf} satisfies exactly one fragmentary subformula of ϱ.

Definition 5.1.5 (Fragmentary Semantics)**.** For every restriction formula ϱ, the *fragmentary semantics* with respect to a structural stationary process P is

$$\begin{array}{rcl} [\![\top]\!]_P & = & \emptyset \\ [\![\phi]\!]_P & = & \{[F] \mid [F] \in fg(rf(Reach(P)))_{/\equiv} \text{ and } F \models_S \phi\} \\ [\![\phi_1 \between \cdots \between \phi_n]\!]_P & = & \{[F_1 \mid \ldots \mid F_n] \mid [F_i] \in [\![\phi_i]\!]_P\} \end{array}$$

That indeed every process Q of a process congruence class in the fragmentary semantics of a restriction formula ϱ satisfies this formula is formulated in Lemma 5.1.4. The proof is rather obvious by Definition 5.1.5.

Lemma 5.1.4. *Let ϱ be a restriction formula and $P, Q \in \mathcal{P}$ be processes such that $[Q] \in [\![\varrho]\!]_P$. Then $Q \models_S \varrho$.*

Proof *of Lemma 5.1.4*
We consider the three possibilities.
Case $[Q] \in [\![\top]\!]_P$**:**
Trivial.
Case $[Q] \in [\![\phi]\!]_P$**:**
Then by Definition 5.1.5 $Q \models_S \phi$.
Case $[Q] \in [\![\phi_1 \between \cdots \between \phi_n]\!]_P$**:**
Then $[Q] = [F_1 \mid \ldots \mid F_n]$ where $[F_i] \in [\![\phi_i]\!]_P$ $(1 \leq i \leq n)$. By induction hypothesis $F_i \models_S \phi_i$ and by the semantics of the parallel composition operator $Q \models_S \phi_1 \between \cdots \between \phi_n$. □

Furthermore, for two processes P and P', where P can react to P' the fragmentary semantics with respect to P' is a subset of the fragmentary semantics with respect to P. This Lemma is needed for the soundness of the translation (Theorem 5.1.1), i.e., to show that if the structural semantics of a process P satisfy a translated formula, then P satisfies the original formula.

Lemma 5.1.5. *Let $P, P' \in \mathcal{P}$ with $P \rightarrow^* P'$ and ϱ be a restriction formula. Then*

$$[\![\varrho]\!]_{P'} \subseteq [\![\varrho]\!]_P.$$

Proof *of Lemma 5.1.5*
Obvious by the definition of the fragmentary semantics, because

$$fg(Reach(P'))_{/\equiv} \subseteq fg(Reach(P))_{/\equiv}.$$

□

Now we can finally define the set of translatable formulas, which we will call *structural formulas*. They consist of conjunctions and disjunctions of restriction formulas.

Definition 5.1.6 (Structural Formulas)**.** The *structural formulas* are defined by the following BNF.

$$\Phi_{\mathcal{S}} ::= \varrho \mid \Phi_{\mathcal{S}1} \vee \Phi_{\mathcal{S}2} \mid \Phi_{\mathcal{S}1} \wedge \Phi_{\mathcal{S}2}$$

The main part of the translation of structural formulas is the translation of restriction formulas. We have to distinguish restriction formulas containing $\top$ from formulas without $\top$, since for the former, the translated formula has to consist of formulas $[F] = c$ where $c \in \mathbb{N}$. This distinction is necessary, because the restricted form of a process P satisfying $\varrho = \phi_1 \between \cdots \between \phi_n$ without $\top$ should be of the form $rf(P) \equiv_{rf} F_1 \mid \dots F_n$ where each F_i satisfies exactly one ϕ_i. If we would translate such a formula with atoms of the form $[F] \geq c$, also $rf(P') \equiv_{rf} rf(P) \mid F_1$ would satisfy the translated formula. Since $P' \not\models_S \varrho$, our translation would be unsound.

Definition 5.1.7 (Translation of Structural Formulas)**.** Let $\Phi_{\mathcal{S}}$ be a structural formula and P be a structural stationary process. The translation $\theta_S \colon \mathcal{PSL} \times \mathcal{P} \rightarrow \mathcal{LTL}$ of $\Phi_{\mathcal{S}}$ is defined by

$$\theta_S(\varrho, P) \quad = \quad \begin{cases} \bigvee_{[Q] \in [\![\varrho]\!]_P} \bigwedge_{[F] \in fg(rf(Reach(P)))_{/\equiv}} [F] = dec(Q)([F]) & \top \notin sub(\varrho) \\ \bigvee_{[Q] \in [\![\varrho]\!]_P} \bigwedge_{[F] \in fg(rf(Reach(P)))_{/\equiv}} [F] \geq dec(Q)([F]) & \text{else} \end{cases}$$

$$\begin{array}{rcl}\theta_S(\Phi_{\mathcal{S}1} \wedge \Phi_{\mathcal{S}2}, P) & = & \theta_S(\Phi_{\mathcal{S}1}, P) \wedge \theta_S(\Phi_{\mathcal{S}2}, P) \\ \theta_S(\Phi_{\mathcal{S}1} \vee \Phi_{\mathcal{S}2}, P) & = & \theta_S(\Phi_{\mathcal{S}1}, P) \vee \theta_S(\Phi_{\mathcal{S}2}, P)\end{array}$$

The following theorem is the most important result of this section as it ensures the soundness of the translation. That is, for every structural stationary process P (see Definition 2.4.6) and every structural formula $\Phi_{\mathcal{S}}$ (Definition 5.1.6), P satisfies $\Phi_{\mathcal{S}}$ if and only if the structural semantics of P satisfy the translated formula with respect to P.

Theorem 5.1.1. *A structural stationary process P satisfies a structural formula $\Phi_{\mathcal{S}}$ if and only if the structural semantics of P satisfies the translated formula with respect to P, i.e.,*

$$P \models_S \Phi_{\mathcal{S}} \quad \textit{iff} \quad \mathcal{N}[\![P]\!] \models_S \theta_S(\Phi_{\mathcal{S}}, P).$$

Proof *of Theorem 5.1.1*
By induction on the structure of $\Phi_{\mathcal{S}}$. For the sake of readability, we will write k for the size of the set of reachable fragments of P up to structural congruence, i.e.,

$$|fg(rf(Reach(P)))_{/\equiv}| = k.$$

Base Case $\Phi_{\mathcal{S}} = \phi_1 \between \cdots \between \phi_n$**, where** $\top \notin sub(\phi_i)$ **for all** $1 \leq i \leq n$**:**
Let $P \models_S \phi_1 \between \cdots \between \phi_n$, i.e.,

$$P \equiv P_1 \mid \ldots \mid P_n \text{ and } P_i \models_S \phi_i \text{ for } 1 \leq i \leq n.$$

By Lemma 5.1.2 we get that $P_i \in \mathcal{P}_{\mathcal{F}}$, i.e., $P_1 \mid \ldots \mid P_n$ is in restricted form and therefore $rf(P) \equiv_{rf} P_1 \mid \ldots \mid P_n$. Furthermore $[P_i] \in [\![\phi_i]\!]_P$. So,

$$[P_1 \mid \ldots \mid P_n] \in [\![\phi_1 \between \cdots \between \phi_n]\!]_P.$$

By the structure of $\theta_S(\phi_1 \between \cdots \between \phi_n, P)$, we get that there is a disjunct ψ of $\theta_S(\phi_1 \between \cdots \between \phi_n, P)$ with

$$\psi = \bigwedge_{[F] \in fg(rf(Reach(P)))_{/\equiv}} [F] = dec(P_1 \mid \ldots \mid P_n)([F]).$$

The initial marking M_0 of $\mathcal{N}[\![P]\!]$ is defined by $M_0 = dec(rf(P))$ and by Lemma 2.4.3 $dec(rf(P)) = dec(P_1 \mid \ldots \mid P_n)$, hence $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \psi$. Because $\theta_S(\phi_1 \between \cdots \between \phi_n, P)$ is in disjunctive normal form, and ψ is one of the disjuncts, also

$$\mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta_S(\phi_1 \between \cdots \between \phi_n, P).$$

For the other direction, assume $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta_S(\phi_1 \between \cdots \between \phi_n, P)$. By the structure of θ_S there is a disjunct

$$\psi = \bigwedge_{i=1}^{k} [F_i] = c_i$$

of $\theta_S(\phi_1 \between \cdots \between \phi_n, P)$ such that $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \psi$. That is, for $1 \leq i \leq k$, we have $M_0([F_i]) = c_i$. By the definition of the initial marking M_0 and Proposition 2.4.1, we get that $retrieve(M_0) = [P]$. The definition of the translation function implies $[P] \in [\![\phi_1 \between \cdots \between \phi_n]\!]_P$, and by Lemma 5.1.4 $P \models_S \phi_1 \between \cdots \between \phi_n$.
Base Case $P \models_S \phi_1 \between \cdots \between \phi_n$, where $\top \in sub(\phi_i)$ for one i with $1 \leq i \leq n$: Then by the semantics of PSTL and Lemma 5.1.3

$$\begin{aligned} P &\equiv P_1 \,|\, P_1' \,|\, \ldots P_n \,|\, P_n' \\ &\equiv P_1 \,|\, \ldots P_n \,|\, P', \end{aligned}$$

with $P' \equiv P_1' \,|\, \ldots \,|\, P_n'$. Furthermore $P_i \models_S \phi_i$ and $P_i \in \mathcal{P}_{\mathcal{F}}$ for $1 \leq i \leq n$. By Definition 5.1.5 $[P_i] \in [\![\phi_i]\!]_P$, hence

$$[P_1 \,|\, \ldots \,|\, P_n] \in [\![\phi_1 \between \cdots \between \phi_n]\!]_P.$$

That is, there is a disjunct ψ with

$$\psi = \bigwedge_{[F] \in fg(rf(Reach(P)))_{/\equiv}} [F] \geq dec(P_1 \,|\, \ldots \,|\, P_n)([F]).$$

Now we consider $M_0 = dec(rf(P))$. Since

$$\begin{aligned} P &\equiv P_1 \,|\, \ldots \,|\, P_n \,|\, P' \\ &\equiv P_1 \,|\, \ldots \,|\, P_n \,|\, rf(P') \\ &\equiv rf(P) \end{aligned}$$

we get by Lemma 2.4.3

$$dec(rf(P)) \;=\; dec(P_1 \,|\, \ldots \,|\, P_n) + dec(rf(P')).$$

Hence for all fragments $[F] \in fg(rf(Reach(P)))_{/\equiv}$ we get

$$M_0([F]) \geq dec(P_1 \,|\, \ldots \,|\, P_n)([F]),$$

which is by the semantics of LTL equivalent to $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \psi$, i.e.,

$$\mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta_S(\phi_1 \between \cdots \between \phi_n, P).$$

For the reverse direction, we assume $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta_S(\phi_1 \between \cdots \between \phi_n, P)$. Then there is at least one disjunct

$$\psi = \bigwedge_{i=1}^{k} p_i \geq c_i$$

with $M_0(p_i) \geq c_i$, where each $p_i = [F_i]$. Now we define the set $I_{\neq 0} \subseteq \{1, \ldots, k\}$ by

$$i \in I_{\neq 0} \quad \text{iff} \quad c_i > 0.$$

By the definition of the translation function,

$$[P'] = [\Pi_{i \in I_{\neq 0}} \Pi^{c_i} F_i] \in [\![\phi_1 \between \cdots \between \phi_n]\!]_P$$

and by Lemma 5.1.4 $P' \models_S \phi_1 \between \cdots \between \phi_n$. Since $M_0([F_i]) \geq c_i$ for all $1 \leq i \leq k$, $P \in \mathit{retrieve}(M_0) = [P' \mid P'']$ and by

$$\phi_1 \between \cdots \between \phi_n \quad \widehat{\equiv} \quad \phi_1 \between \cdots \between \phi_n \between \top,$$

we conclude $P \models_S \phi_1 \between \cdots \between \phi_n$.

Because the induction step is rather easy, we will only show the case for the conjunction. The case of $\vee$ is similar.

Case $\Phi_{\mathcal{S}} = \Phi_{\mathcal{S}1} \wedge \Phi_{\mathcal{S}2}$:

Let $P \models_S \Phi_{\mathcal{S}1} \wedge \Phi_{\mathcal{S}2}$. This is by definition of the consequence relation equivalent to

$$\begin{array}{rll}
 & & P \models_S \Phi_{\mathcal{S}1} \text{ and } P \models_S \Phi_{\mathcal{S}2} \\
\{\text{Induction Hypothesis}\} & \text{iff} & \mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta_S(\Phi_{\mathcal{S}1}, P) \text{ and } \mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta_S(\Phi_{\mathcal{S}2}, P) \\
\{\text{Semantics of LTL}\} & \text{iff} & \mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta_S(\Phi_{\mathcal{S}1}, P) \wedge \theta_S(\Phi_{\mathcal{S}2}, P) \\
\{\text{Definition 5.1.7}\} & \text{iff} & \mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta_S(\Phi_{\mathcal{S}1} \wedge \Phi_{\mathcal{S}2}, P).
\end{array}$$

□

Example 5.1.1. Consider the following process definitions

$$\begin{array}{rcl}
\mathit{BAG}(\mathit{in}, \mathit{out}) & := & \mathit{in}(x).(\overline{\mathit{out}}\langle x \rangle \mid \mathit{BAG}\lfloor \mathit{in}, \mathit{out} \rfloor) \\
\mathit{FILL}(\mathit{in}) & := & \nu \mathit{val}.\overline{\mathit{in}}\langle \mathit{val} \rangle.\mathit{FILL}\lfloor \mathit{in} \rfloor \\
P & \equiv & \mathit{BAG}\lfloor \mathit{in}, \mathit{out} \rfloor \mid \mathit{FILL}\lfloor \mathit{in} \rfloor \mid \nu \mathit{val}.\overline{\mathit{out}}\langle \mathit{val} \rangle
\end{array}$$

and the restriction formula

$$\Phi := BAG\lfloor in, out\rfloor \between FILL\lfloor in\rfloor.$$

Observe that $P \not\models_S \Phi$. We will now construct the structural translation of Φ to see that $\mathcal{N}[\![P]\!] \not\models_S \theta_S(\Phi, P)$. The reachable fragments of P are

$$fg(Reach(P))_{/\equiv} = \{[BAG\lfloor in, out\rfloor], [FILL\lfloor in\rfloor], [\nu val.\overline{out}\langle val\rangle]\},$$

hence the fragmentary semantics of Φ is given by

$$\begin{aligned} [\![BAG\lfloor in, out\rfloor]\!]_P &= \{[BAG\lfloor in, out\rfloor]\} \\ [\![FILL\lfloor in\rfloor]\!]_P &= \{[FILL\lfloor in\rfloor]\} \\ [\![BAG\lfloor in, out\rfloor \between FILL\lfloor in\rfloor]\!]_P &= \{[BAG\lfloor in, out\rfloor \mid FILL\lfloor in\rfloor]\}. \end{aligned}$$

Since $[\![\Phi]\!]_P$ only consists of one process congruence class, the translated formula is a conjunction of atoms.

$$\theta_S(\Phi, P) \quad = \quad [BAG\lfloor in, out\rfloor] = 1 \wedge [FILL\lfloor in\rfloor] = 1 \wedge [\nu val.\overline{out}\langle val\rangle] = 0$$

Since the initial marking of $\mathcal{N}[\![P]\!]$ yields $M_0([\nu val.\overline{out}\langle val\rangle]) = 1$, we get that $\mathcal{N}[\![P]\!] \not\models_{\text{LTL}} \theta_S(\Phi, P)$.

An interesting and helpful property of the translation is, that its satisfaction by the structural semantics of a process P' only depends on the fragments of P'. That is, if we consider a process P and P' such that P can react to P' the truth value of the translated formula with respect to P' under the structural semantics of P' is the same as the truth value of the formula translated with respect to P. This rather surprising result is due to the relation of the reachable fragments of P and P'. Since all reachable fragments of P' are also reachable from P, the fragmentary semantics with respect to P' are a subset of the fragmentary semantics with respect to P (refer to Lemma 5.1.5). Hence the formulas created due to processes in the fragmentary semantics w.r.t. P' are subformulas of the formulas created w.r.t. P.

Lemma 5.1.6. *Let $P, P' \in \mathcal{P}$ such that $P \to^* P'$ and $\Phi_\mathcal{S}$ be a structural formula. Then*

$$\mathcal{N}[\![P']\!] \models_{\text{LTL}} \theta_S(\Phi_\mathcal{S}, P') \textit{ iff } \mathcal{N}[\![P']\!] \models_{\text{LTL}} \theta_S(\Phi_\mathcal{S}, P)$$

Proof *of Lemma 5.1.6*
By induction on the structure on $\Phi_\mathcal{S}$. The only interesting case is the base case.

Base Case $\Phi_{\mathcal{S}} = \varrho$:
For this proof, it is irrelevant, if ϱ contains $\top$, so we will consider both cases together. Let $\mathcal{N}[\![P']\!] \models_{\text{LTL}} \theta_S(\varrho, P')$, i.e

$$\mathcal{N}[\![P']\!] \models_{\text{LTL}} \bigvee_{[Q]\in[\![\varrho]\!]_{P'}} \bigwedge_{[F]\in fg(rf(Reach(P')))_{/\equiv}} [F] \sim dec(Q)([F]),$$

where $\sim \in \{=, \geq\}$. This implies the existence of a process congruence class $[Q] \in [\![\varrho]\!]_{P'}$ and a subformula ψ' of $\theta_S(\varrho, P')$ satisfying

$$\psi' = \bigwedge_{[F]\in fg(rf(Reach(P')))_{/\equiv}} [F] \sim dec(rf(Q))([F])$$

and $\mathcal{N}[\![P']\!] \models_{\text{LTL}} \psi'$. By Lemma 5.1.5 $[Q] \in [\![\varrho]\!]_P$, i.e. there is a subformula ψ of $\theta_S(\varrho, P)$ such that

$$\psi = \bigwedge_{[F]\in fg(rf(Reach(P)))_{/\equiv}} [F] \sim dec(rf(Q))([F]).$$

Since by definition of the fragmentary semantics $rf(Q)$ contains only fragments of P', $dec(rf(Q))([F]) = 0$ for all fragments $[F] \in fg(rf(Reach(P)))_{/\equiv} \backslash fg(rf(Reach(P')))_{/\equiv}$, hence $\mathcal{N}[\![P']\!] \models_{\text{LTL}} \psi$, i.e., $\mathcal{N}[\![P']\!] \models_{\text{LTL}} \theta_S(\varrho, P)$.

Now consider the reverse direction, i.e., for

$$\mathcal{N}[\![P']\!] \models_{\text{LTL}} \bigvee_{[Q]\in[\![\varrho]\!]_{P}} \bigwedge_{[F]\in fg(rf(Reach(P)))_{/\equiv}} [F] \sim dec(rf(Q))([F]).$$

There has to be a disjunct ψ and a process congruence class $[Q] \in [\![\varrho]\!]_P$ with

$$\psi = \bigwedge_{[F]\in fg(rf(Reach(P)))_{/\equiv}} [F] \sim dec(rf(Q))([F])$$

such that $\mathcal{N}[\![P']\!] \models_{\text{LTL}} \psi$. This is by the semantics of LTL equivalent to

$$M_0([F]) \sim dec(rf(Q))([F]) \text{ for all places } [F] \text{ of } \mathcal{N}[\![P']\!],$$

i.e. all fragments $[F] \in fg(rf(Reach(P')))_{/\equiv}$. Since $M_0 = dec(P')$,

$$M_0([F]) = 0 \text{ for all } [F] \in fg(rf(Reach(P)))_{/\equiv} \setminus fg(rf(Reach(P')))_{/\equiv}.$$

Hence $dec(rf(Q))[F] = 0$ for all fragments $[F]$ of P which are not fragments of P', i.e., $rf(Q)$ consists only of fragments of P' which implies that $[rf(Q)] \in [\![\varrho]\!]_{P'}$. Therefore there exists a disjunct ψ' of $\theta_S(\varrho, P')$

$$\psi' = \bigwedge_{[F]\in fg(rf(Reach(P')))_{/\equiv}} [F] \sim dec(rf(Q))([F])$$

with $\mathcal{N}[\![P']\!] \models_{\mathrm{LTL}} \psi'$. This implies $\mathcal{N}[\![P']\!] \models_{\mathrm{LTL}} \theta_S(\varrho, P')$.
Case $\Phi_{\mathcal{S}} = \Phi_{\mathcal{S}1} \wedge \Phi_{\mathcal{S}2}$:

Let $\mathcal{N}[\![P']\!] \models_{\mathrm{LTL}} \theta_S(\Phi_{\mathcal{S}1} \wedge \Phi_{\mathcal{S}2}, P')$.

$$\begin{array}{rll}
\{\text{Definition 5.1.7}\} & \text{iff} & \mathcal{N}[\![P']\!] \models_{\mathrm{LTL}} \theta_S(\Phi_{\mathcal{S}1}, P') \wedge \theta_S(\Phi_{\mathcal{S}2}, P') \\
\{\text{Semantics of LTL}\} & \text{iff} & \mathcal{N}[\![P']\!] \models_{\mathrm{LTL}} \theta_S(\Phi_{\mathcal{S}1}, P') \text{ and } \mathcal{N}[\![P']\!] \models_{\mathrm{LTL}} \theta_S(\Phi_{\mathcal{S}2}, P') \\
\{\text{Induction Hypothesis}\} & \text{iff} & \mathcal{N}[\![P']\!] \models_{\mathrm{LTL}} \theta_S(\Phi_{\mathcal{S}1}, P) \text{ and } \mathcal{N}[\![P']\!] \models_{\mathrm{LTL}} \theta_S(\Phi_{\mathcal{S}2}, P) \\
\{\text{Semantics of LTL}\} & \text{iff} & \mathcal{N}[\![P']\!] \models_{\mathrm{LTL}} \theta_S(\Phi_{\mathcal{S}1}, P) \wedge \theta_S(\Phi_{\mathcal{S}2}, P) \\
\{\text{Definition 5.1.7}\} & \text{iff} & \mathcal{N}[\![P']\!] \models_{\mathrm{LTL}} \theta_S(\Phi_{\mathcal{S}1} \wedge \Phi_{\mathcal{S}2}, P)
\end{array}$$

Case $\Phi_{\mathcal{S}} = \Phi_{\mathcal{S}1} \vee \Phi_{\mathcal{S}2}$:

Similar. □

5.2 Translating formulas with temporal modalities

In this section we will define a translation θ from a subset of PSTL to LTL. We will denote formulas containing temporal connectives by $\Phi_{\mathcal{T}}$. The translatable formulas are restricted in the possible atoms, because we can only translate structural formulas $\Phi_{\mathcal{S}}$ as given by Definition 5.1.6. Hence, the temporal formulas $\Phi_{\mathcal{T}}$ defined in Section 3.2 can be translated as long as the atoms are structural formulas $\Phi_{\mathcal{S}}$.

Definition 5.2.1. Let $\Phi_{\mathcal{T}}$ be a formula of PSTL and P be a π-Calculus process. Then the translation $\theta\colon \mathcal{PSTL} \times \mathcal{P} \to \mathcal{LTL}$ is defined by

$$\begin{array}{rcl}
\theta(\Phi_{\mathcal{S}}, P) & = & \theta_S(\Phi_{\mathcal{S}}, P) \\
\theta(\neg\Phi_{\mathcal{T}}, P) & = & \neg\theta(\Phi_{\mathcal{T}}, P) \\
\theta(\Phi_{\mathcal{T}1} \wedge \Phi_{\mathcal{T}2}, P) & = & \theta(\Phi_{\mathcal{T}1}, P) \wedge \theta(\Phi_{\mathcal{T}2}, P) \\
\theta(\bigcirc\Phi_{\mathcal{T}}, P) & = & \bigcirc\theta(\Phi_{\mathcal{T}}, P) \\
\theta(\Diamond\Phi_{\mathcal{T}}, P) & = & \Diamond\theta(\Phi_{\mathcal{T}}, P)
\end{array}$$

Another translation we need is a mapping from process sequences to occurrence sequences, to relate the semantics of PSTL and LTL. Therefore we exploit the isomorphic structure of the transition systems of a process P and its structural semantics $\mathcal{N}[\![P]\!]$. As Meyer has shown [Mey08], the function $f\colon \mathit{Reach}(P)_{/\equiv} \to \mathit{Reach}(\mathcal{N}[\![P]\!])$, with $f([Q]) = \mathit{dec}(\mathit{rf}(Q))$ is an isomorphism (see also Proposition 2.4.1, Section 2.4) . We lift this isomorphism to work on sequences.

Definition 5.2.2. Let $P \in \mathcal{P}$ and $\pi = \pi(0)\pi(1)\pi(2)\ldots$ be a process sequence with $\pi(0) = P$. Furthermore, let f be the isomorphism of Proposition 2.4.1. Then $g\colon \mathcal{PC} \to \mathcal{OC}$ is defined by $g(\pi) = f(\pi(0))f(\pi(1))f(\pi(2))\ldots$.

Since f is an isomorphism, we directly get that $g(\pi)$ is an occurrence sequence, i.e. $g(\pi)(0) [\rangle\, g(\pi)(1) [\rangle\, g(\pi)(2)\ldots$, and since $\pi(0) = P$, we also get that $g(\pi)(0)$ is the initial marking M_0 of $\mathcal{N}[\![P]\!]$.

In order to prove the soundness of translation θ, we first prove the following lemma. It shows that if we consider two processes P and P' with $P \to^* P'$, the satisfaction of the translated formula $\theta(\Phi_{\mathcal{T}}, P)$ under the structural semantics of P' only depends on fragments of P'. I.e., the translation of the formula with respect to P is equivalent to the translation of $\Phi_{\mathcal{T}}$ with respect to P'.

Lemma 5.2.1. *Let P and P' be two processes with $P \to^* P'$. Furthermore, let $\Phi_{\mathcal{T}}$ be a formula of PSTL, σ be an occurrence sequence of $\mathcal{N}[\![P']\!]$ and $x \in \mathbb{N}$. Then the following equivalence holds:*

$$\sigma, x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}}, P') \textit{ iff } \sigma, x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}}, P)$$

Proof *of Lemma 5.2.1*
By induction on the structure of formulas.
Base Case $\Phi_{\mathcal{T}} = \Phi_{\mathcal{S}}$:
The statement follows immediately by Lemma 5.1.6, since the statement holds for all occurrence sequences of $\mathcal{N}[\![P']\!]$.

The induction step is a straightforward application of Definition 5.2.1 and the semantics of LTL. We will show the cases of $\wedge$ and $\bigcirc$.
Case $\Phi_{\mathcal{T}} = \Phi_{\mathcal{T}1} \wedge \Phi_{\mathcal{T}2}$:

$$\begin{array}{rll}
 & & \sigma, x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1} \wedge \Phi_{\mathcal{T}2}, P') \\
\{\text{Definition 5.2.1}\} & \text{iff} & \sigma, x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, P') \wedge \theta(\Phi_{\mathcal{T}2}, P') \\
\{\text{Semantics of LTL}\} & \text{iff} & \sigma, x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, P') \text{ and } \sigma, x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}2}, P') \\
\{\text{Induction Hypothesis}\} & \text{iff} & \sigma, x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}2}, P) \text{ and } \sigma, x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}2}, P) \\
\{\text{Semantics of LTL}\} & \text{iff} & \sigma, x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, P) \wedge \theta(\Phi_{\mathcal{T}2}, P) \\
\{\text{Definition 5.2.1}\} & \text{iff} & \sigma, x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1} \wedge \Phi_{\mathcal{T}2}, P)
\end{array}$$

Case $\Phi_{\mathcal{T}} = \bigcirc\Phi_{\mathcal{T}1}$:

$$\begin{array}{rll}
 & & \sigma, x \models_{\text{LTL}} \theta(\bigcirc\Phi_{\mathcal{T}1}, P') \\
\{\text{Definition 5.2.1}\} & \text{iff} & \sigma, x \models_{\text{LTL}} \bigcirc\theta(\Phi_{\mathcal{T}1}, P') \\
\{\text{Semantic of LTL}\} & \text{iff} & \sigma, x+1 \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, P') \\
\{\text{Induction Hypothesis}\} & \text{iff} & \sigma, x+1 \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, P) \\
\{\text{Semantics of LTL}\} & \text{iff} & \sigma, x \models_{\text{LTL}} \bigcirc\theta(\Phi_{\mathcal{T}1}, P) \\
\{\text{Definition 5.2.1}\} & \text{iff} & \sigma, x \models_{\text{LTL}} \theta(\bigcirc\Phi_{\mathcal{T}1}, P)
\end{array}$$

□

Corollary 5.2.1. *Let P and P' be two processes with $P \rightarrow^* P'$. Furthermore, let $\Phi_{\mathcal{T}}$ be a formula of PSTL. Then the following equivalence holds:*

$$\mathcal{N}[\![P']\!] \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}}, P') \textit{ iff } \mathcal{N}[\![P']\!] \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}}, P).$$

Proof *of Corollary 5.2.1*
Immediate by Lemma 5.2.1. □

That our translation is well-defined with respect to satisfaction of formulas is shown by the following proposition. That is, if we have a translatable temporal formula $\Phi_{\mathcal{T}}$, a process sequence π and an integer number x such that π and x satisfy $\Phi_{\mathcal{T}}$, then the translation of $\Phi_{\mathcal{T}}$ is satisfied by the image of π under the homomorphism g and x, and vice versa. We prove this statement by induction on the translatable formulas. The base case involves Theorem 5.1.1 of the last section, i.e., the well-definition of the structural translation, as well as Lemma 5.2.1, which can be applied due to the isomorphic behaviour of g. This proposition is the core of the proof for our main result, Theorem 5.2.1.

Proposition 5.2.1. *Let g be the isomorphism of Definition 5.2.2. Then for every process sequence π, $x \in \mathbb{N}$ and temporal formula $\Phi_{\mathcal{T}}$, the following equivalence holds:*

$$\pi, x \models \Phi_{\mathcal{T}} \textit{ iff } g(\pi), x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}}, \pi(0)).$$

Proof *of Proposition 5.2.1*
By induction on the structure of formulas.

Base Case $\Phi_{\mathcal{T}} = \Phi_{\mathcal{S}}$:

$$\begin{aligned}
& & & \pi, x \models \Phi_{\mathcal{S}} \\
&\{\text{Definition 3.2.3}\} & \text{iff} \quad & \pi(x) \models_S \Phi_{\mathcal{S}} \\
&\{\text{Theorem 5.1.1}\} & \text{iff} \quad & \mathcal{N}[\![\pi(x)]\!] \models_{\text{LTL}} \theta_S(\Phi_{\mathcal{S}}, \pi(x)) \\
&\{\text{Definition 5.2.1}\} & \text{iff} \quad & \mathcal{N}[\![\pi(x)]\!] \models_{\text{LTL}} \theta(\Phi_{\mathcal{S}}, \pi(x)) \\
&\{\pi(0) \rightarrow^* \pi(x) \text{ and Lemma 5.2.1}\} & \text{iff} \quad & \mathcal{N}[\![\pi(x)]\!] \models_{\text{LTL}} \theta(\Phi_{\mathcal{S}}, \pi(0)) \\
&\{\sigma \text{ occurrence seq. of } \mathcal{N}[\![\pi(x)]\!]\} & \text{iff} \quad & \sigma, 0 \models_{\text{LTL}} \theta(\Phi_{\mathcal{S}}, \pi(0)) \\
&\{g \text{ isomorph.} \Leftrightarrow g(\pi)(x) = \sigma(0)\} & \text{iff} \quad & g(\pi), x \models_{\text{LTL}} \theta(\Phi_{\mathcal{S}}, \pi(0))
\end{aligned}$$

Case $\Phi_{\mathcal{T}} = \neg\Phi_{\mathcal{T}1}$:

$$\begin{aligned}
& & & \pi, x \models \neg\Phi_{\mathcal{T}1} \\
&\{\text{Definition 3.2.3}\} & \text{iff} \quad & \pi, x \not\models \Phi_{\mathcal{T}1} \\
&\{\text{Induction Hypothesis}\} & \text{iff} \quad & g(\pi), x \not\models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, \pi(0)) \\
&\{\text{Definition 2.5.3}\} & \text{iff} \quad & g(\pi), x \models_{\text{LTL}} \neg\theta(\Phi_{\mathcal{T}1}, \pi(0)) \\
&\{\text{Definition 5.2.1}\} & \text{iff} \quad & g(\pi), x \models_{\text{LTL}} \theta(\neg\Phi_{\mathcal{T}1}, \pi(0))
\end{aligned}$$

Case $\Phi_{\mathcal{T}} = \Phi_{\mathcal{T}1} \wedge \Phi_{\mathcal{T}2}$:

$$\begin{aligned}
& & & \pi, x \models \Phi_{\mathcal{T}1} \wedge \Phi_{\mathcal{T}2} \\
&\{\text{Definition 3.2.3}\} & \text{iff} \quad & \pi, x \models \Phi_{\mathcal{T}1} \text{ and } \pi, x \models \Phi_{\mathcal{T}2} \\
&\{\text{Induction Hypothesis}\} & \text{iff} \quad & \quad\;\; g(\pi), x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, \pi(0)) \\
& & & \text{and } g(\pi), x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}2}, \pi(0)) \\
&\{\text{Definition 2.5.3}\} & \text{iff} \quad & g(\pi), x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, \pi(0)) \wedge \theta(\Phi_{\mathcal{T}2}, \pi(0)) \\
&\{\text{ Definition 5.2.1}\} & \text{iff} \quad & g(\pi), x \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1} \wedge \Phi_{\mathcal{T}2}, \pi(0))
\end{aligned}$$

Case $\Phi_{\mathcal{T}} = \bigcirc\Phi_{\mathcal{T}1}$:

$$\begin{aligned}
& & & \pi, x \models \bigcirc\Phi_{\mathcal{T}1} \\
&\{\text{Definition 3.2.3}\} & \text{iff} \quad & \pi, x+1 \models \Phi_{\mathcal{T}1} \\
&\{\text{Induction Hypothesis}\} & \text{iff} \quad & g(\pi), x+1 \models \theta(\Phi_{\mathcal{T}1}, \pi(0)) \\
&\{\text{Definition 2.5.3}\} & \text{iff} \quad & g(\pi), x \models \bigcirc\theta(\Phi_{\mathcal{T}1}, \pi(0)) \\
&\{\text{Definition 5.2.1}\} & \text{iff} \quad & g(\pi), x \models \theta(\bigcirc\Phi_{\mathcal{T}1}, \pi(0))
\end{aligned}$$

Case $\Phi_{\mathcal{T}} = \Diamond \Phi_{\mathcal{T}1}$**:**

$$\begin{array}{rll} & & \pi, x \models \Diamond \Phi_{\mathcal{T}1} \\ \{\text{Definition 3.2.3}\} & \text{iff} & \exists y \geq x \colon \pi, y \models \Phi_{\mathcal{T}1} \\ \{\text{Induction Hypothesis}\} & \text{iff} & \exists y \geq x \colon g(\pi), y \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, \pi(0)) \\ \{\text{Definition 2.5.3}\} & \text{iff} & g(\pi), x \models_{\text{LTL}} \Diamond \theta(\Phi_{\mathcal{T}1}, \pi(0)) \\ \{\text{Definition 5.2.1}\} & \text{iff} & g(\pi), x \models_{\text{LTL}} \theta(\Diamond \Phi_{\mathcal{T}1}, \pi(0)) \end{array}$$

□

The main result of this section is Theorem 5.2.1. It is a straightforward application of Proposition 5.2.1, where we set $x = 0$. That is, a process P satisfies a temporal formula $\Phi_{\mathcal{T}}$ if and only if the structural semantics of P satisfies the translated formula with respect to P.

Theorem 5.2.1. *Let P be a structurally congruent process and $\Phi_{\mathcal{T}}$ be a formula of PSTL as in Definition 3.2.1 where all atoms of $\Phi_{\mathcal{T}}$ are defined according to Definition 5.1.6. Then*

$$P \models \Phi_{\mathcal{T}} \quad \textit{iff} \quad \mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}}, P).$$

Proof *of Theorem 5.2.1*
Let $P \models \Phi_{\mathcal{T}}$, that is for every process sequence π with $\pi(0) = P$, we have

$$\pi, 0 \models \Phi_{\mathcal{T}}.$$

Hence by Proposition 5.2.1, $g(\pi), 0 \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}}, P)$, i.e., $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}}, P)$.

Conversely let $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}}, P)$, i.e., every occurrence sequence σ with $\sigma(0) = M_0$ satisfies $\theta(\Phi_{\mathcal{T}}, P)$, where M_0 is the initial marking of $\mathcal{N}[\![P]\!]$. Because g is an isomorphism, for every σ, $g^{-1}(\sigma)$ is a process sequence of P, i.e., there is a process sequence π with $g(\pi) = \sigma$. Hence $g(\pi), 0 \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}}, P)$. With Proposition 5.2.1, we conclude $\pi, 0 \models_{\text{LTL}} \Phi_{\mathcal{T}}$. Therefore $P \models \Phi_{\mathcal{T}}$. □

5.3 Example: Client/Server

We recall Example 2.3.1 and its tagged process system (refer to Definition 4.1.2 and Example 4.1.1). The system of a server S and two clients C communicating via the public channel *url* is given by the following process equations.

$$\begin{aligned} C(url) &:= \nu ip.\overline{url}\langle ip\rangle.ip(s).s(x).C\lfloor url\rfloor \\ S(url) &:= url(y).\nu ses.\overline{y}\langle ses\rangle.\overline{ses}\langle ses\rangle.S\lfloor url\rfloor \\ \mathrm{SYS} &:= S\lfloor url\rfloor \mid C\lfloor url\rfloor \mid C\lfloor url\rfloor \end{aligned}$$

The tagged process system of this definition is

$$\begin{aligned} C^{\mathrm{SYS}}(url) &:= \nu ip.\overline{url}\langle ip\rangle.ip(s).s(x).C^{C}\lfloor url\rfloor \\ C^{C}(url) &:= \nu ip.\overline{url}\langle ip\rangle.ip(s).s(x).C^{C}\lfloor url\rfloor \\ S^{\mathrm{SYS}}(url) &:= url(y).\nu ses.\overline{y}\langle ses\rangle.\overline{ses}\langle ses\rangle.S^{S}\lfloor url\rfloor \\ S^{S}(url) &:= url(y).\nu ses.\overline{y}\langle ses\rangle.\overline{ses}\langle ses\rangle.S^{S}\lfloor url\rfloor \\ \mathrm{SYS}^{\mathrm{SYS}} &:= C^{\mathrm{SYS}}\lfloor url\rfloor \mid C^{\mathrm{SYS}}\lfloor url\rfloor \mid S^{\mathrm{SYS}}\lfloor url\rfloor \end{aligned}$$

where we already omitted the unreachable process equations.

To construct the structural semantics of $P \equiv \mathrm{SYS}^{\mathrm{SYS}}$, we will abbreviate the fragments in $fg(rf(Reach(P)))$ by F_i where $0 \leq i \leq 8$. These fragments are given by the following equalities.

$$\begin{aligned} F_0 &= \mathrm{SYS}^{\mathrm{SYS}} \\ F_1 &= C^{\mathrm{SYS}}\lfloor url\rfloor \\ F_2 &= C^{C}\lfloor url\rfloor \\ F_3 &= S^{\mathrm{SYS}}\lfloor url\rfloor \\ F_4 &= S^{S}\lfloor url\rfloor \\ F_5 &= \nu ip.\overline{url}\langle ip\rangle.ip(s).s(x).C^{C}\lfloor url\rfloor \\ F_6 &= url(y).\nu ses.\overline{y}\langle ses\rangle.\overline{ses}\langle ses\rangle.S^{S}\lfloor url\rfloor \\ F_7 &= \nu ip.(ip(s).s(x).C^{C}\lfloor url\rfloor \mid \nu ses.\overline{ip}\langle ses\rangle.\overline{ses}\langle ses\rangle.S^{S}\lfloor url\rfloor) \\ F_8 &= \nu ses.(ses(x).C^{C}\lfloor url\rfloor \mid \overline{ses}\langle ses\rangle.S^{S}\lfloor url\rfloor) \end{aligned}$$

Hence the Petri net $\mathcal{N}[\![P]\!]$ consists of 9 places. It is shown in Figure 5.2. We will now define some properties in PSTL and present the translation into LTL. Consider the PSTL formulas $\Phi_{\mathcal{S}1}$, $\Phi_{\mathcal{S}2}$,$\Phi_{\mathcal{T}1}$ and $\Phi_{\mathcal{T}2}$ defined by

$$\begin{aligned} \Phi_{\mathcal{S}1} &= C \wedge free(url) \between S \wedge free(url) \between \top \\ \Phi_{\mathcal{S}2} &= \mathrm{res}\ i\ (C \wedge free(url,i) \between S \wedge free(url,i) \between \top) \\ \Phi_{\mathcal{T}1} &= \Diamond(\Phi_{\mathcal{S}1}) \\ \Phi_{\mathcal{T}2} &= \Box(\Phi_{\mathcal{S}1} \Rightarrow \Diamond(\Phi_{\mathcal{S}2})) \end{aligned}$$

We get that P satisfies neither $\Phi_{\mathcal{S}1}$ nor $\Phi_{\mathcal{S}2}$. With at most four reactions, P reaches a state

$$P' \equiv F_6 \,|\, F_5 \,|\, Q.$$

That is, all process sequences π of P include P'. This process satisfies $\Phi_{\mathcal{S}1}$, hence $P \models \Phi_{\mathcal{T}1}$. For Q we get that either $Q \equiv C^{\mathrm{SYS}}\lfloor url \rfloor$ or $Q \equiv F_6$. Now we consider the possible reactions of $F_5 \,|\, F_6$, i.e.,

$$\begin{array}{rl} & F_5 \,|\, F_6 \,|\, Q \\ \to & \nu ip.(ip(s).s(x).C^C\lfloor url \rfloor \,|\, \nu ses.\overline{ip}\langle ses\rangle.\overline{ses}\langle ses\rangle.S^S\lfloor url \rfloor) \,|\, Q \\ \equiv & F_7 \,|\, Q \end{array}$$

This process satisfies $\Phi_{\mathcal{S}2}$, since $\Phi_{\mathcal{S}2} \mathrel{\widehat{\equiv}} \Phi_{\mathcal{S}2} \between \top$,

$$\begin{array}{rcl} \nu ses.\overline{ip}\langle ses\rangle.\overline{ses}\langle ses\rangle.S^S\lfloor url \rfloor & \models_S & S \wedge \mathit{free}(url,i)\{i \leftarrow ip\} \\ ip(s).s(x).C^C\lfloor url \rfloor & \models_S & C \wedge \mathit{free}(url,i)\{i \leftarrow ip\} \\ Q & \models_S & \top \end{array}$$

and

$$ip \notin \mathit{fn}(F_7) \cup \mathit{fn}(S \wedge \mathit{free}(url,i)) \cup \mathit{fn}(C \wedge \mathit{free}(url,i)).$$

All reactions of F_7 reach a process which is again structurally congruent to F_7, hence these process sequences satisfy $\Phi_{\mathcal{T}2}$. The process Q does not intervene with these reactions, since it either expands to F_6, or it can not react at all.

Observe that $\Phi_{\mathcal{S}1}$ and $\Phi_{\mathcal{S}2}$ are well-formed formulas with respect to Definition 5.1.6. In particular, they are restriction formulas (Definition 5.1.4). Hence, we can translate these formulas into LTL. We decompose $\Phi_{\mathcal{S}1}$ into its fragmentary formulas

$$\begin{array}{rcl} \phi_1 & = & C \wedge \mathit{free}(url) \\ \phi_2 & = & S \wedge \mathit{free}(url) \end{array}$$

The fragmentary semantics of these formulas with respect to P are

$$\begin{array}{rcl} [\![\phi_1]\!]_P & = & \{[F_2], [F_5]\} \\ [\![\phi_2]\!]_P & = & \{[F_4], [F_6]\} \end{array}$$

hence the fragmentary semantics of $\Phi_{\mathcal{S}1}$ are

$$[\![\Phi_{\mathcal{S}1}]\!]_P = \{[F_2 \,|\, F_4], [F_2 \,|\, F_6], [F_5 \,|\, F_4], [F_5 \,|\, F_6]\}.$$

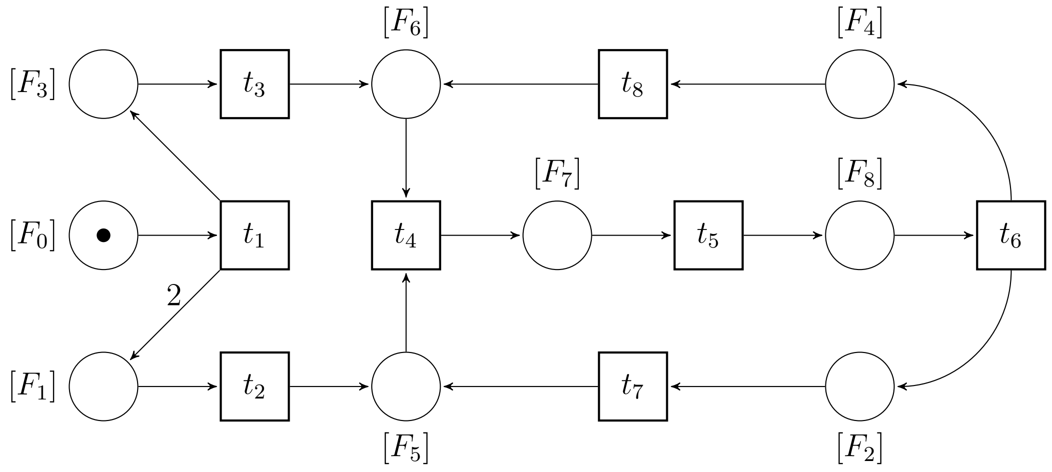

Figure 5.2: The structural semantics of $P \equiv \text{SYS}^{\text{SYS}}$

The translated formula $\theta(\Phi_{\mathcal{S}1}, P)$ is

$$\begin{aligned}
\theta(\Phi_{\mathcal{S}1}, P) = \quad & ([F_0] \geq 0 \wedge [F_1] \geq 0 \wedge [F_2] \geq 1 \wedge [F_3] \geq 0 \wedge [F_4] \geq 1 \\
& \wedge [F_5] \geq 0 \wedge [F_6] \geq 0 \wedge [F_7] \geq 0 \wedge [F_8] \geq 0) \\
\vee \quad & ([F_0] \geq 0 \wedge [F_1] \geq 0 \wedge [F_2] \geq 1 \wedge [F_3] \geq 0 \wedge [F_4] \geq 0 \\
& \wedge [F_5] \geq 0 \wedge [F_6] \geq 1 \wedge [F_7] \geq 0 \wedge [F_8] \geq 0) \\
\vee \quad & ([F_0] \geq 0 \wedge [F_1] \geq 0 \wedge [F_2] \geq 0 \wedge [F_3] \geq 0 \wedge [F_4] \geq 1 \\
& \wedge [F_5] \geq 1 \wedge [F_6] \geq 0 \wedge [F_7] \geq 0 \wedge [F_8] \geq 0) \\
\vee \quad & ([F_0] \geq 0 \wedge [F_1] \geq 0 \wedge [F_2] \geq 0 \wedge [F_3] \geq 0 \wedge [F_4] \geq 0 \\
& \wedge [F_5] \geq 1 \wedge [F_6] \geq 1 \wedge [F_7] \geq 0 \wedge [F_8] \geq 0)
\end{aligned}$$

For the sake of readability, we will omit all atoms of the form $[F_i] \geq 0$, i.e., we simplify the translated formula to

$$\begin{aligned}
\theta(\Phi_{\mathcal{S}1}, P) \quad = \quad & ([F_2] \geq 1 \wedge [F_4] \geq 1) \\
& \vee([F_5] \geq 1 \wedge [F_4] \geq 1) \\
& \vee([F_2] \geq 1 \wedge [F_6] \geq 1) \\
& \vee([F_5] \geq 1 \wedge [F_6] \geq 1)
\end{aligned}$$

The formula $\Phi_{\mathcal{S}2}$ already is a fragmentary formula. The fragmentary semantics of $\Phi_{\mathcal{S}2}$ are

$$[\![\Phi_{\mathcal{S}2}]\!]_P \quad = \quad \{[F_7], [F_8]\}.$$

We have already shown that $F_7 \models_S \Phi_{\mathcal{S}2}$. That $F_8 \models_S \Phi_{\mathcal{S}2}$ is due to

$$\begin{array}{rcl} ses(x).C^{C}\lfloor url \rfloor & \models_S & C \wedge \mathit{free}(url, i)\{i \leftarrow ses\} \\ \overline{ses}\langle ses\rangle.S^{S}\lfloor url \rfloor & \models_S & S \wedge \mathit{free}(url, i)\{i \leftarrow ses\} \end{array}$$

and the fact that ses is fresh with respect to F_8 and the formulas $C \wedge \mathit{free}(url, i)$ and $S \wedge \mathit{free}(url, i)$. Therefore, the translation of $\Phi_{\mathcal{S}2}$ with respect to P is

$$\theta(\Phi_{\mathcal{S}1}, P) \quad = \quad [F_7] \geq 1 \vee [F_8] \geq 1.$$

We see, that $\mathcal{N}[\![P]\!] \not\models_{\text{LTL}} \theta(\Phi_{\mathcal{S}1}, P)$ and $\mathcal{N}[\![P]\!] \not\models_{\text{LTL}} \theta(\Phi_{\mathcal{S}2}, P)$. Now we consider the occurrence sequences of the structural semantics of P to verify that indeed $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, P)$ and $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}2}, P)$. As the translations of the formulas are easy after translating the structural formulas, we will not explicitly write them down. After transition t_1 of the Petri net of Figure 5.2 fires, there are three possible subsequent firing transitions, because the place $[F_1]$ carries 2 tokens. If t_3 and t_2 fire at least once, we reach a marking M with $M([F_5]) \geq 1$ and $M([F_6]) \geq 1$. (If t_2 fires twice, the only enabled transition of $\mathcal{N}[\![P]\!]$ is t_3). Hence this marking satisfies $\theta(\Phi_{\mathcal{S}1},)$, i.e., $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \Diamond\theta(\Phi_{\mathcal{S}1}, P)$, which is $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, P)$.

This marking enables t_4, and after this transition fires, we get the marking M' with $M'([F_7]) \geq 1$, i.e., $\theta(\Phi_{\mathcal{S}2}, P)$ is true. All possible firing transitions either lead to markings, where $\theta(\Phi_{\mathcal{S}1}, P)$ is false, hence trivially satisfying the implication, or to a marking M'' where again $M''([F_5]) \geq 1$ and $M''([F_6]) \geq 1$. So, we can conclude that $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \Box(\theta(\Phi_{\mathcal{S}1}, P) \Rightarrow \Diamond(\theta(\Phi_{\mathcal{S}2}, P)))$, that is $\mathcal{N}[\![P]\!] \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}2}, P)$.

5.4 Example: Bag

In this section, we will examine a data structure called *bag*. A bag can be filled with arbitrary values and returns them in no specific order. We have already used process equations defining a bag in Example 5.1.1. Now we will not only consider a process filling the bag, but also a *consumer* process, which removes values from the bag. We will directly give the tagged process system, since the definition of the bag and the filling process are the same as in Example 5.1.1 and the definition of the consuming process is straightforward. The following tagged process system describes the system of a bag, a filling and a consuming process.

$$\begin{array}{rcl}
\mathrm{SYS}^{\mathrm{SYS}} & := & FILL^{\mathrm{SYS}}\lfloor in\rfloor \mid BAG^{\mathrm{SYS}}\lfloor in, out\rfloor \mid CONS^{\mathrm{SYS}}\lfloor out\rfloor \\
FILL^{\mathrm{SYS}}(in) & := & \nu val.\overline{in}\langle val\rangle.FILL^{FILL}\lfloor in\rfloor \\
FILL^{FILL}(in) & := & \nu val.\overline{in}\langle val\rangle.FILL^{FILL}\lfloor in\rfloor \\
BAG^{\mathrm{SYS}}(in, out) & := & in(x).(\overline{out}\langle x\rangle.D^{BAG} \mid BAG^{BAG}\lfloor in, out\rfloor) \\
BAG^{BAG}(in, out) & := & in(x).(\overline{out}\langle x\rangle.D^{BAG} \mid BAG^{BAG}\lfloor in, out\rfloor) \\
CONS^{\mathrm{SYS}}(out) & := & in(y).CONS^{CONS}\lfloor out\rfloor \\
CONS^{CONS}(out) & := & in(y).CONS^{CONS}\lfloor out\rfloor
\end{array}$$

Furthermore, let $P \equiv \mathrm{SYS}^{\mathrm{SYS}}$. The set $Reach(P)_{/\equiv}$ is infinite, since an arbitrary number of communications between the bag and the filling process can occur, where the consumer never uses data. On the other hand, the set of reachable fragments is finite up to structural congruence, because all data values $\nu val.\overline{out}\langle cal\rangle.D^{BAG}$ are fragments and structurally congruent. Again, we abbreviate the fragments.

$$\begin{array}{rcl}
F_0 & = & \mathrm{SYS}^{\mathrm{SYS}} \\
F_1 & = & FILL^{\mathrm{SYS}}\lfloor in\rfloor \\
F_2 & = & BAG^{\mathrm{SYS}}\lfloor in\rfloor \\
F_3 & = & FILL^{FILL}\lfloor in\rfloor \\
F_4 & = & BAG^{BAG}\lfloor in\rfloor \\
F_5 & = & \nu val.\overline{in}\langle val\rangle.FILL^{FILL}\lfloor in\rfloor \\
F_6 & = & in(x).(\overline{out}\langle x\rangle.D^{BAG} \mid BAG^{BAG}\lfloor in, out\rfloor) \\
F_7 & = & \nu val.\overline{out}\langle val\rangle.D^{BAG} \\
F_8 & = & CONS^{\mathrm{SYS}}\lfloor out\rfloor \\
F_9 & = & out(x).CONS^{CONS}\lfloor out\rfloor \\
F_{10} & = & CONS^{CONS}\lfloor out\rfloor \\
F_{11} & = & D^{BAG}
\end{array}$$

Now we consider the formulas

$$\begin{array}{rcl}
\Phi_{\mathcal{S}1} & = & BAG \wedge \mathit{free}(in) \between FILL \wedge \mathit{free}(in) \between \top \\
\Phi_{\mathcal{S}2} & = & \diamondsuit_{rf}(\mathrm{res}\ v\ (BAG \wedge \mathit{free}(v, out))) \\
\Phi_{\mathcal{T}} & = & \Box(\Phi_{\mathcal{S}1} \Rightarrow \Diamond\Phi_{\mathcal{S}2})
\end{array}$$

The formula $\Phi_{\mathcal{T}}$ states, that whenever we have a bag and a filling process able to communicate over the channel *in*, we finally get a processes, which is able to send a saved value over the channel *out*.

The process P satisfies neither $\Phi_{\mathcal{S}1}$ nor $\Phi_{\mathcal{S}2}$, but we will show that P satisfies $\Phi_{\mathcal{T}}$. After at most four reactions of P, we get a process which is congruent to

$$\begin{aligned} & \nu val.\overline{in}\langle val\rangle.FILL^{FILL}\lfloor in\rfloor \mid in(x).(\overline{out}\langle x\rangle.D^{BAG} \mid BAG^{BAG}\lfloor in, out\rfloor) \mid P' \\ \equiv\ & F_5 \mid F_6 \mid P' \end{aligned}$$

where $P' \equiv CONS^{\mathrm{SYS}}\lfloor out\rfloor$ or $P' \equiv out(x).CONS^{CONS}\lfloor out\rfloor$. Because

$$\begin{aligned} \nu val.\overline{in}\langle val\rangle.FILL^{FILL}\lfloor in\rfloor &\models_S FILL \wedge \mathit{free}(in) \\ in(x).(\overline{out}\langle x\rangle.D^{BAG} \mid BAG^{BAG}\lfloor in, out\rfloor) &\models_S BAG \wedge \mathit{free}(in) \\ P' &\models_S \top \end{aligned}$$

and due to the commutativity of $\between$, we get that

$$F_5 \mid F_6 \mid P' \models_S \Phi_{\mathcal{S}1}.$$

Hence there is at least one reachable state of P satisfying $\Phi_{\mathcal{S}1}$. Now we direct our attention to $\Phi_{\mathcal{S}2}$. As a first step of the examination of this formula, we expand the abbreviation $\diamondminus_{rf}$.

$$\Phi_{\mathcal{S}2} = \text{res } out \ \text{ res } v\ (BAG \wedge \mathit{free}(v, out) \between \top) \vee \text{res } v\ (BAG \wedge \mathit{free}(v, out) \between \top)$$

Observe, that no fragment satisfies the first part of the disjunction, since there is at most one restriction on every fragment. The only fragment satisfying this formula is $F_7 = \nu val.\overline{out}\langle val\rangle.D^{BAG}$. The only other fragment which is a sequential process of BAG with out as a free name is F_6. But $F_6 \not\models_S \Phi_{\mathcal{S}2}$, even though F_6 contains the bound name x. Consider the process

$$F_6' = \nu x.in(x).(\overline{out}\langle x\rangle.D^{BAG} \mid BAG^{BAG}\lfloor in, out\rfloor)$$

which is structurally congruent to F_6. Now, we try to find a name $m \notin \mathit{fn}(F_6') \cup \mathit{fn}(BAG \wedge \mathit{free}(v, out))$ such that $F_6 \models_S BAG \wedge \mathit{free}(m, out)$. We can not choose $m = in$, since $in \in \mathit{free}(F_6')$. The only possibility for m would be $m = x$, but

$$in(x).(\overline{out}\langle x\rangle.D^{BAG} \mid BAG^{BAG}\lfloor in, out\rfloor) \not\models_S BAG \wedge \mathit{free}(x, out).$$

For the satisfaction of $\Phi_{\mathcal{T}}$, we have to consider every reachable process Q satisfying $\Phi_{\mathcal{S}1}$ and show that every possible process sequence starting in Q satisfies $\Diamond\Phi_{\mathcal{S}2}$. If $F_7 \in \mathit{fg}(\mathit{rf}(Q))$, this is obviously true. Let $F_7 \notin \mathit{fg}(\mathit{rf}(Q))$, which corresponds intuitively to the situation, that the consuming process has removed every value from the bag. Since $Q \models_S \Phi_{\mathcal{S}1}$, there are Q' and Q'' such that $Q \equiv Q' \mid Q'' \mid R$ and

$$\begin{aligned} Q' &\models_S FILL \wedge \mathit{free}(in) \\ Q'' &\models_S BAG \wedge \mathit{free}(in). \end{aligned}$$

If $Q' = F_3$, it can react to F_5. Similarly, $Q'' = F_4$ can react to F_6. Therefore, we eventually get the situation described above, that is $F_5 \mid F_6$, which only can react to a process structurally congruent to F_7. Hence every process Q satisfying $\Phi_{\mathcal{S}1}$ where $F_7 \notin \mathit{fg}(\mathit{rf}(Q))$ must react to a process containing F_7. That is, every process sequence starting in Q satisfies $\Diamond\Phi_{\mathcal{S}2}$. Altogether we have shown that $P \models \Phi_{\mathcal{T}}$.

Now we translate $\Phi_{\mathcal{T}}$ to an equivalent formula of LTL. Therefore, we first translate the structural formulas $\Phi_{\mathcal{S}1}$ and $\Phi_{\mathcal{S}2}$. The fragmentary semantics of $\Phi_{\mathcal{S}1}$ and $\Phi_{\mathcal{S}2}$ are computed by

$$\begin{aligned} [\![BAG \wedge \mathit{free}(in)]\!]_P &= \{[F_4], [F_6]\} \\ [\![FILL \wedge \mathit{free}(in)]\!]_P &= \{[F_3], [F_5]\} \\ [\![\Phi_{\mathcal{S}1}]\!]_P &= \{[F_4 \mid F_3], [F_4 \mid F_5], [F_6 \mid F_3], [F_6 \mid F_5]\} \\ [\![\Phi_{\mathcal{S}2}]\!]_P &= \{[F_7]\}. \end{aligned}$$

If we omit all atoms of the form $p \geq 0$, the translation of $\Phi_{\mathcal{T}}$ is

$$\begin{aligned} \theta(\Phi_{\mathcal{T}}, P) = \Box(& (([F_4] \geq 1 \wedge [F_3] \geq 1) \\ & \vee ([F_4] \geq 1 \wedge [F_5] \geq 1) \\ & \vee ([F_6] \geq 1 \wedge [F_3] \geq 1) \\ & \vee ([F_6] \geq 1 \wedge [F_5] \geq 1)) \\ & \Rightarrow (\Diamond[F_7] \geq 1)). \end{aligned}$$

The structural semantics of P are depicted in Figure 5.3). For the satisfaction of the translation of $\Phi_{\mathcal{T}}$, we will name the transitions as defined in this figure. We show that every marking satisfying the premise of the implication also satisfies the conclusion, i.e., $\Diamond[F_7] \geq 1$. Let M be a marking with $M([F_5]) \geq 1$, $M([F_6]) \geq 1$ and $M([F_7]) = 0$. Then, t_4 is enabled. Firing t_4 results in a marking M' with $M'([F_7]) = 1$. Now, if a marking satisfies $[F_3] \geq 1$ and $[F_5] = 0$, the transition t_5 is enabled, whose firing marks $[F_5]$ with one token. A similar argument holds for markings satisfying $[F_4] \geq 1$ and transition t_6 which marks $[F_6]$ with one token.

Furthermore because of the structure of the net, only $[F_7]$ and $[F_{11}]$ can carry more than one token at a time. We already assumed that $[F_7]$ carries no tokens, and even if there are one or more tokens on $[F_{11}]$, $M([F_{11}) = c$ with $c \geq 1$, t_9 can only fire c times. Furthermore, the only circular dependencies in the net involve the transition t_4. Since every M satisfying the premise above leads to a marking where t_4 is enabled, this transition will eventually fire in all occurrence sequences and the resulting marking satisfies $[F_7] \geq 1$. Hence for all occurrence sequences of $\mathcal{N}[\![P]\!]$, we get $\sigma \models_{\text{LTL}} \theta(\Phi_{\mathcal{T}1}, P)$.

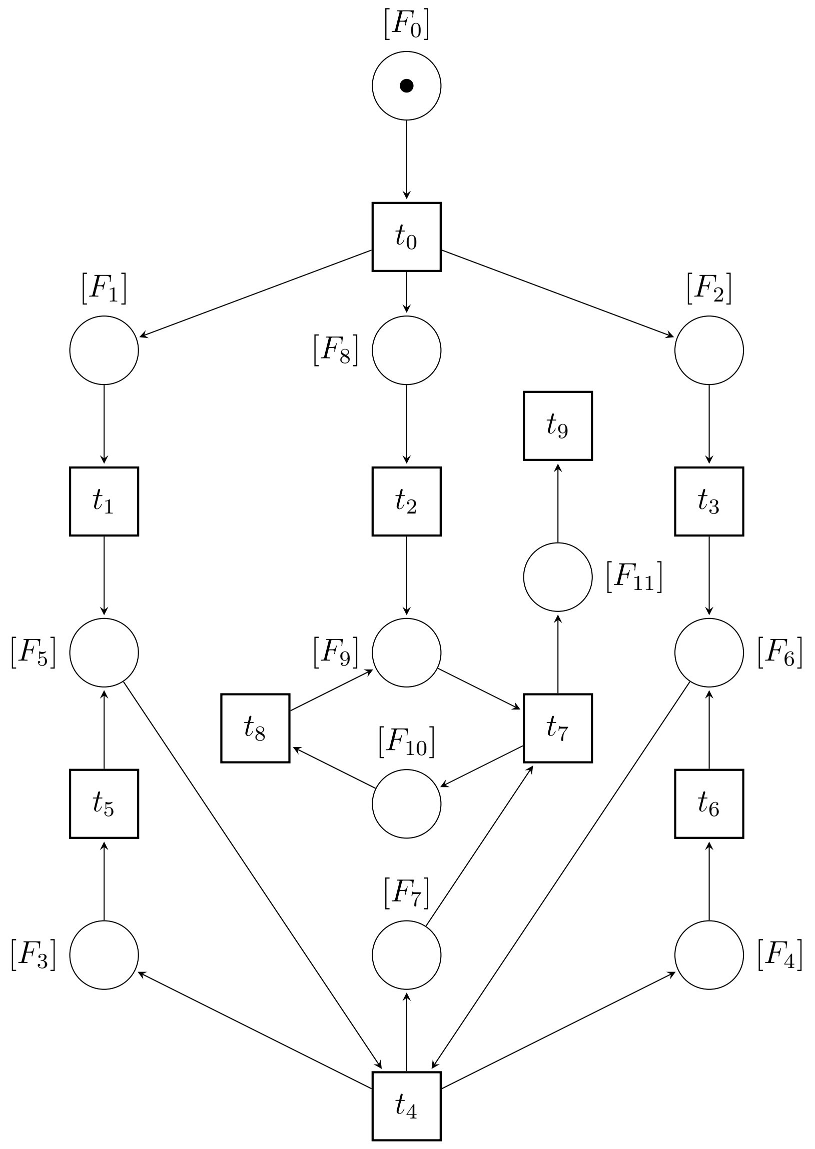

Figure 5.3: The structural semantics of P

6 Conclusion and related work

Conclusion

In this thesis, we have defined a structural and temporal logic for the π-Calculus called PSTL, which allows us to reason about the structural configuration and the temporal evolution of processes. We have split the logic into two distinctive subsets. First the structural formulas, which specify the structural aspect of a process and second the temporal formulas, which describe the possible reactions of the processes.

Besides the standard Boolean connectives, the syntax of structural formulas includes an operator resembling the parallel composition of processes and a quantifier on restricted names of processes. We have shown that the structural subset of our logic is well-defined with respect to the Gabbay-Pitts-Property [GP99]. Furthermore, our restriction quantifier is νx-proper. This property distinguishes well-defined quantification on hidden names from flawed and unsatisfactory quantification as shown by Cardelli and Gordon [CG01]. We have explicitly proved many equivalences of structural formulas, thereby increasing the intuition and understanding of structural reasoning with our logic.

Our temporal formulas are by design similar to LTL formulas, with the difference that the atoms of our logic are possibly complex structural formulas. We have thereby a rather strong restriction on the possible combinations of the operators of the structural syntax and the temporal modalities. This restriction is intended, to obtain clearly readable formulas and to simplify reasoning with our logic.

For the same purpose, we have introduced new atoms specifying derivatives of processes. That is, all processes directly obtained from a process equation can be identified by a single, simple atom instead of mentioning all these processes explicitly. The drawback of this introduction is a blow-up of the number of process equations defining a process system. This disadvantage is reduced by the fact that normally a significant number of the newly created equations can be ignored, since they are never expanded during the reactions of the process system.

Our last contribution is a translation of a subset of our logic to standard LTL on Petri nets constructed according to the structural semantics by Meyer [Mey08]. This translation strongly relies on definitions introduced by Meyer that is in particular a new normal form for processes called restricted form and the decomposition function, which splits processes along subprocesses not connected by restricted names. We have shown that our translation is well-defined in the sense, that a process satisfies a formula if and only if the structural semantics of the process satisfies the translated formula. With this result, our logic can be used as a specification language for model checking π-Calculus processes. The tool PETRUCHIO, developed by Strazny [Str07] is an implementation of Meyers structural semantics. It uses the Model Checking Kit (MCKIT) [SSE03] together with the model checker SPIN [Hol97] as a back end to check the resulting Petri nets with respect to a given LTL formula. At present, the work flow with PETRUCHIO normally is as follows:

The user specifies the π-Calculus process she wants to check and uses the internal compiler of PETRUCHIO for the translation to a Petri net. After this compilation, the user identifies the important places of the resulting Petri net, describes the property to check with LTL and then invokes the model checker. Since the Petri nets created with respect to the structural semantics consist normally of a large number of places and transitions, the identification of important places is very error-prone. Hence using our logic and subsequently applying the translation to LTL on Petri nets defined in this thesis would increase the usability of PETRUCHIO significantly. For a schematic picture of the resulting work flow, see Figure 6.1.

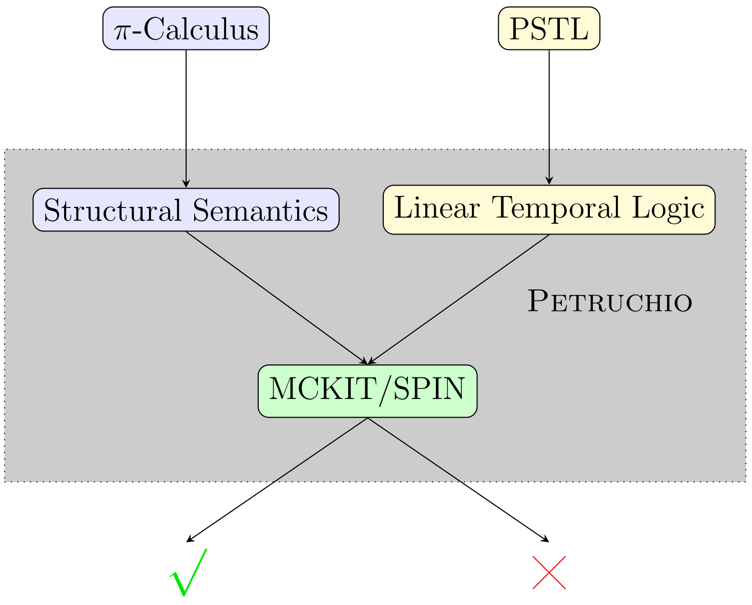

Figure 6.1: Model checking π-Calculus-processes against PSTL-formulas with PETRUCHIO

In this thesis, we have not considered the “until” modality at all since we used a definition of LTL following the early work of Pnueli. But our approach can be generalised in a straightforward way, to handle formulas containing this modality, due to the similarity of the semantics of PSTL and LTL.

Our result for translatable formulas is far from optimal. We restrict the use of many operators and do not allow for negation at all. However, we believe that translating formulas with negation is possible, by developing a treatment similar to the translation of formulas containing the atomic formula $\top$. Furthermore, we think that all formulas are equivalent to a formula in a normal form similar to the restricted form of processes [Mey08]. For developing this normal form, more structural equivalences than shown in the thesis are probably needed.

It is not possible to relate processes with their derivatives, which would be convenient, for example to identify data values saved in a data structure. Such identification is not reflected in the structural semantics and hence not expressible with our logic.

Finally, our approach has yet to be implemented in the tool Petruchio.

Related Work

Even though structural properties have been always been discussed in concurrency literature, most research concerned solely the behaviour of processes, for example Hennessy-Milner-Logic [HM85]. However, the last years have shown increasing interest in analysing the structural configurations of processes. This development has been furthered by Gabbay and Pitts, who used Fraenkel-Mostovski (FM) set theory to model hidden names [GP99]. Cardelli and Gordon [CG01] used their results to explore logical properties of name restriction in the Ambient Calculus [CGT98], an extension of the π-Calculus. Gabbay embedded the π-Calculus into the FM set theory [Gab03], while Pitts defined a first order logic concerned with variable binding [Pit03]. Independent from this development, Caires and Monteiro explored the same questions by studying logical specifications of mobility in processes [CM98]. All these developments concentrate on deductive systems for the defined logics, which are not considered in this thesis at all.

In 2003, Caires and Cardelli published their work on a spatial and temporal logic [CC01, CC02] for the π-Calculus, for which they provided a model-checking algorithm. Charatonik and Talbot have examined decidability for the ambient logic with name restriction with negative results [CT01].

Recently Mardare and Policriti have defined a Hilbert-style axiomatic system for a spatial logic [MP08] on a subset of the Calculus of Communicating Systems

[Mil80]. Even though, they proved decidability of model checking for this subset, their work lack consideration of name restriction.

Bibliography

[CC01] L. Caires and L. Cardelli. A spatial logic for concurrency (part i). In Takayasu Ito, editor, *Proceedings of the Fourth International Symposium on Theoretical Aspects of Computer Science (TACS 01)*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[CC02] L. Caires and L. Cardelli. A spatial logic for concurrency (part ii. In *In Proceedings of the 13th International Conference on Concurrency Theory (CONCUR*, pages 209–225. Springer-Verlag, 2002.

[CG00] L. Cardelli and A. D. Gordon. Anytime, anywhere: modal logics for mobile ambients. In *POPL '00: Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 365–377, New York, NY, USA, 2000. ACM.

[CG01] L. Cardelli and A. D. Gordon. Logical properties of name restriction. In *TLCA*, pages 46–60, 2001.

[CG06] L. Cardelli and A. D. Gordon. Ambient logic. *Mathematical Structures in Computer Science*, 2006. To appear.

[CGT98] W. Charatonik, A. D. Gordon, and J. Talbot. Mobile ambients. In *Theoretical Computer Science*, pages 140–155. Springer-Verlag, 1998.

[CM98] L. Caires and L. Monteiro. Verifiable and executable logic specifications of concurrent objects in $\mathcal{L}_\pi$. In Chris Hankin, editor, *Programming Languages and Systems, Proceedings of ESOP'98*, Lecture Notes in Computer Science, pages 42–56. Springer-Verlag, 1998.

[CT01] W. Charatonik and J. Talbot. The decidability of model checking mobile ambients. In *CSL '01: Proceedings of the 15th International Workshop on Computer Science Logic*, pages 339–354, London, UK, 2001. Springer-Verlag.

[CZ97] A. V. Chagrov and M. V. Zakharyaschev. *Modal logics*, volume 35 of *Oxford logic guides*. Clarendon Press, 1997.

[Dam89] M. Dam. *Relevance logic and Concurrent Composition.* PhD thesis, University of Edinburgh, 1989.

[Dam96] M. Dam. Model checking mobile processes. *Information and Computation*, 129:35–51, August 1996.

[EC80] E. A. Emerson and E. M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, pages 169–181, London, UK, 1980. Springer-Verlag.

[Gab03] M. J. Gabbay. The pi-calculus in FM. In Fairouz Kamareddine, editor, *Thirty-five years of Automath*, volume 28 of *Kluwer applied logic series*, pages 247–269. Kluwer, November 2003.

[GP99] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax involving binders. *Logic in Computer Science, Symposium on*, 0:214, 1999.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *JACM*, 32(1):137–161, 1985.

[Hoa78] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978.

[Hol97] G. Holzmann. The model checker spin. In *IEEE Trans. on Software Engineering*, volume 23, pages 279–295, May 1997.

[Lam94] Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16:872–923, 1994.

[Mey07] R. Meyer. A Petri Net Semantics for π-Calculus Verification. In *Dagstuhl "zehn plus eins"*, pages 76–77. Verlagshaus Mainz GmbH Aachen, 2007.

[Mey08] R. Meyer. A theory of structural stationarity in the π-calculus. Accepted for publication in Acta Informatica, 2008.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[Mil99] R. Milner. *Communicating and Mobile Systems: the Pi-Calculus.* Cambridge University Press, June 1999.

[MP08] R. Mardare and A. Policriti. A complete axiomatic system for a process-based spatial logic. In *MFCS*, pages 491–502, 2008.

[MPW91] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. volume 527 of *Lecture Notes in Computer Science*, pages 45–60. Springer-Verlag, 1991.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts i and ii. *Information and Computation*, 100(1):1–77, 1992.

[Pet62] C. A. Petri. Kommunikation mit Automaten. Technical report, 1962.

[Pit03] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Inf. Comput.*, 186(2):165–193, 2003.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium Foundations of Computer Science (FOCS 1977)*, pages 46–57, 1977.

[Rei85] W. Reisig. *Petri nets: an introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[SSE03] C. Schröter, S. Schwoon, and J. Esparza. The Model-Checking Kit. In W. van der Aalst and E. Best, editors, *Applications and Theory of Petri Nets 2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 463–472. Springer-Verlag, 2003.

[Str07] T. Strazny. Entwurf und Implementierung von Algorithmen zur Berechnung von Petrinetz-Semantiken für Pi-Kalkül-Prozesse. Master's thesis, University of Oldenburg, 2007.

[SW01] D. Sangiorgi and D. Walker. *PI-Calculus: A Theory of Mobile Processes*. Cambridge University Press, New York, NY, USA, 2001.

[Vaa93] F. W. Vaandrager. Expressiveness results for process algebras. pages 609–638. Springer-Verlag, 1993.

Symbol Index

Subject Index

Hiermit versichere ich, die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt zu haben. Die verwendete Literatur und sonstige Hilfsmittel sind vollständig angegeben.

Oldenburg, den 27. November 2008