Bachelor thesis
Computing Science

Radboud University

Axiomatising Trace Semantics for an Algebra with Parallel Composition

Author:
Cas Haaijman
s4372662

First supervisor/assessor:
dr. J. C. (Jurriaan) Rot
jrot@cs.ru.nl

Second supervisor:
drs. J. (Jana) Wagemaker
J.Wagemaker@cs.ru.nl

Second assessor:
prof. dr. F. W. (Frits) Vaandrager
F.Vaandrager@cs.ru.nl

May 25, 2021

Abstract

Process algebra is a research area that uses algebras and calculi to study concurrent systems. Two processes are called trace equivalent if every sequence of actions that can be taken in one can also be taken in the other. Recent work by Aceto et al. showed the existence of an axiomatisation for trace equivalence for the algebra $\mathrm{BCCSP}_{\|}$. It was part of a large collection of axiomatisations for many different notions of equivalence.

In this thesis we prove that the axiomatisation for $\mathrm{BCCSP}_{\|}$ is sound and ground-complete using a mostly stand alone, ground-up approach. For soundness we construct a relation between the grammar and the language this grammar generates. For completeness we prove that the approach the work by Aceto et al. suggests holds. Finally we show completeness modulo completed trace semantics using a similar method, but use a normal form for the terms.

Contents

Chapter 1

Introduction

The world of computer science has used formal methods to enhance its understanding of both software and hardware for many years. Before process algebra, formal reasoning about software was mainly done using out of three styles of semantics: Operational semantics, Denotational semantics or Axiomatic semantics [3]. It turned out, however, that these styles were not ideal for describing the behaviour of programs that contained multiple parts running in parallel. Out of the need to describe these types of programs, the field of process algebras was born.

In process algebra, the term *process* refers to the behaviour of a system [3]. Often this system is a computer system, such as a network protocol [4], a software program [2], or a hardware specification [2]. It has, however, also been used to refer to other systems. For example, the behaviour of biological systems has been described using process algebra [9]. For example, one study in the field of systems biology looked at the movement in response to chemicals (chemotaxis) of E. coli using process algebra. In this example, the process in question was the behaviour of the bacteria.

An *algebra* is a mathematical object used to construct these processes. The algebra calculus of communicating systems (CCS) was one of the first algebras to be developed and remains one of the most influential. For example, the tool called Conurrency Workbench was originally developed for CCS. This tool allows software developers to analyse and manipulate a concurrent model of their code [6]. The algebra CCS uses symbols for a few different concepts. Some of these concepts are inactive processes, actions the process can perform and non-deterministic choice. It has a slightly larger set of operations (*grammar*) than the algebra we will be using in this thesis, $\mathrm{BCCSP}_{\|}$.

Most of these algebras describe a graph for each process in the algebra. Just like in finite automata are there nodes and edges for these graphs, and sometimes even loops or accepting nodes. These graphs are usually generated from algebraic terms that belong to one of the algebras. This is similar to

how regular expressions generate automata.

Alongside of the development of these algebras, many notions of equivalence were being developed. Some examples of these semantics are (completed) trace semantics, failure equivalence and bisimulation. These different notions of equivalence have different properties. For example, Trace semantics is unable to distinguish whether or not a process has ended in a deadlock or completed in some cases. Bisimulation is often seen as the most accurate semantics, and became the standard way to compare terms. It is also one of the more computationally hard notions of equivalence.

In order to make computation of equivalence easier, sets of equations which are called axiom systems might be used. When an axiom system is sound and complete for a specific algebra and notion of equivalence it is called an axiomatisation of that algebra and equivalence. Such an axiomatisation is sound and complete if and only if the set of equations say that two processes are equivalent, then that means that they actually are equivalent. Sometimes these axiomatisations are finite, but often this is not possible [1], [5].

The algebra we describe in this thesis is $\text{BCCSP}_{\|}$. It extends the algebra called BCCSP with the parallel interleaving operator $\|$. BCCSP itself is smaller than CCS, as it consist of only the more basic operators of CCS and another algebra called CSP, which was developed around the same time as CCS. Perhaps because of this simpler nature of BCCSP, it has been explored quite thoroughly in the context of the many notions of equivalence in for example [8]. Recently, a paper by Aceto et al. has done a similarly extensive investigation into the language $\text{BCCSP}_{\|}$. This investigation was very influential to this thesis and is its main inspiration.

1.1 Our Contribution

In this thesis we show that an axiomatisation for $\text{BCCSP}_{\|}$ modulo trace semantics is sound and complete. Unlike the previous proof, we do this using a mostly stand alone, ground-up approach. Our method for proving soundness specifically is an approach we did not see before. Our approach for completeness is similar but more in detail. We also go more in detail for completeness modulo completed trace semantics of another axiomatisation for $\text{BCCSP}_{\|}$.

1.2 Outline

First we set up the basic notions we need for our algebra in the Preliminaries section. The goal of this is to hopefully make the thesis accessible to people who are unfamiliar to the field. We explain, for example, how the algebra $\text{BCCSP}_{\|}$ is defined and what trace semantics are. Then we give a set of axioms which we show to be sound and ground-complete for the algebra

$\mathrm{BCCSP}_{\|}$ modulo trace semantics. We start of by relating the symbols used to construct terms in $\mathrm{BCCSP}_{\|}$ to the languages those terms generate. This helps us to prove the soundness part of the proof. Then we show we can reduce each term in $\mathrm{BCCSP}_{\|}$ to a BCCSP term. We will use this property to prove the ground-completeness part of the proof. Finally we will look at a different axiomatisation for a different notion of equivalence, completed trace semantics. We will once again proof ground-completeness by reduction to BCCSP. This time, however, we introduce a normal form of our terms in the process.

Chapter 2

Preliminaries

2.1 The Language BCCSP$_{\|}$

We start by defining the process algebra BCCSP$_{\|}$, which is the classical algebra BCCSP enriched with the interleaving parallel composition operator. For this algebra we first define a syntax. The set $\mathbb{P}(\mathcal{V})$ is the set of all terms generated by the following grammar:

$$t ::= \mathbf{0} \mid v \mid a.t \mid t + t \mid t \| t$$

Here v ranges over a countably infinite set of variables $\mathcal{V}$ and $a.$ ranges over a set of unary operators where a represents an element of a finite set of actions $\mathcal{A}$. We adopt standard convention that prefixing binds strongest and $+$ binds weakest. A BCCSP$_{\|}$ term is *closed* if it does not contain any variables. A set of closed terms generated by this grammar ($\mathbb{P}(\emptyset)$) will be abbreviated to $\mathbb{P}$. We refer to a closed term as a *process*.

Now that we have a syntax, we define a semantics to describe how processes behave in our algebra. The $a.$ symbols indicates that some action a can be taken. The '$+$' symbol represents a non-deterministic choice. Finally, the '$\|$' symbol means that the left and right terms are running in parallel. This all leads to the the semantics shown in table 2.1. For our algebra we will equip processes with these semantics. We can now use these rules to create a labeled state transition system ($\mathbb{P}$, $\mathcal{A}$, $\rightarrow$). The transitions $\rightarrow \subseteq \mathbb{P} \times \mathcal{A} \times \mathbb{P}$

$$(s_1)\ \frac{}{a.x \xrightarrow{a} x} \qquad (s_2)\ \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \qquad (s_3)\ \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$$

$$(s_4)\ \frac{x \xrightarrow{a} x'}{x \| y \xrightarrow{a} x' \| y} \qquad (s_5)\ \frac{y \xrightarrow{a} y'}{x \| y \xrightarrow{a} x \| y'}$$

Table 2.1: Semantics of BCCSP$_{\|}$ terms where x, x', y and y' are processes.

are restricted by the terms that can be derived using the rules as defined above. That is to say, $(p, a, p') \in \rightarrow$ if and only if its inclusion can be proven using these rules. From now on we will use $p \xrightarrow{a} p'$ to indicate $(p, a, p') \in \rightarrow$. As an example, the proof tree shown in fig. 2.1 proves that the transition $a.\mathbf{0} + b.\mathbf{0} \| c.\mathbf{0} \xrightarrow{a} \mathbf{0} \| c.\mathbf{0}$ holds.

$$\dfrac{\dfrac{\dfrac{}{a.\mathbf{0} \xrightarrow{a} \mathbf{0}}\,(s_1)}{a.\mathbf{0} + b.\mathbf{0} \xrightarrow{a} \mathbf{0}}\,(s_2)}{a.\mathbf{0} + b.\mathbf{0} \| c.\mathbf{0} \xrightarrow{a} \mathbf{0} \| c.\mathbf{0}}\,(s_4)$$

Figure 2.1: Proof of transition $a.\mathbf{0} + b.\mathbf{0} \| c.\mathbf{0} \xrightarrow{a} \mathbf{0} \| c.\mathbf{0}$

We can now use this proof and combine it with similar proofs to construct the transition system that belongs to the process $a.\mathbf{0} + b.\mathbf{0} \| c.\mathbf{0}$. The transition system that follows is shown in figure 2.2. Each transition has the rules used to prove its existence written next to it. The example from figure 2.1 is highlighted by a thicker line in the transition diagram.

2.2 Traces

A sequence of actions $w = a_1 a_2 \cdots a_k$ $(k \geq 0)$ is a *trace* of p_0 if there exists a sequence of transitions $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} p_k$ where all p_i with $i \in \{0, 1, \ldots, k\}$ are processes. We use $p_0 \xrightarrow{w} p_k$ as an abbreviation. The set of all possible finite sequences of actions in $\mathcal{A}$ will be denoted as $\mathcal{A}^*$ and any subset of $\mathcal{A}^*$ will be called a *language*. For the sequence with zero actions $(k = 0)$ we will use ϵ. This sequence is also known as the *empty trace*. Clearly, the empty trace is a trace of all processes as it is always possible to use no rule at all and end up where you started. The function that maps a process to its set of all traces will be denoted as $\mathtt{T}$ and has type $\mathbb{P} \rightarrow \mathcal{P}(\mathcal{A}^*)$ such that $\mathtt{T}(x)$ is the set of all traces from x, if $x \in \mathbb{P}$. The example process $p_{ex} = a.\mathbf{0} + b.\mathbf{0} \| c.\mathbf{0} \xrightarrow{a} \mathbf{0} \| c.\mathbf{0}$ from figure 2.2 has $\mathtt{T}(p_{ex}) = \{\epsilon, a, b, c, ac, bc, ca, cb\}$ as the set of all traces.

Two processes p_1 and p_2 are *trace equivalent* if their sets of traces are the same: $\mathtt{T}(p_1) = \mathtt{T}(p_2)$, denoted by $p_1 \sim_T p_2$. For example, process $q_{ex} = c.\mathbf{0} \| b.\mathbf{0} + a.\mathbf{0}$ also has $\mathtt{T}(q_{ex}) = \{\epsilon, a, b, c, ac, bc, ca, cb\}$. This is the same set as $\mathtt{T}(p_{ex})$, so $p_{ex} \sim_T q_{ex}$.

2.3 Operations on sequences and languages

We define the operators $\cdot$, $_*$, $\|$ and $\|\|$ on sequences of actions and languages.

Definition 1.

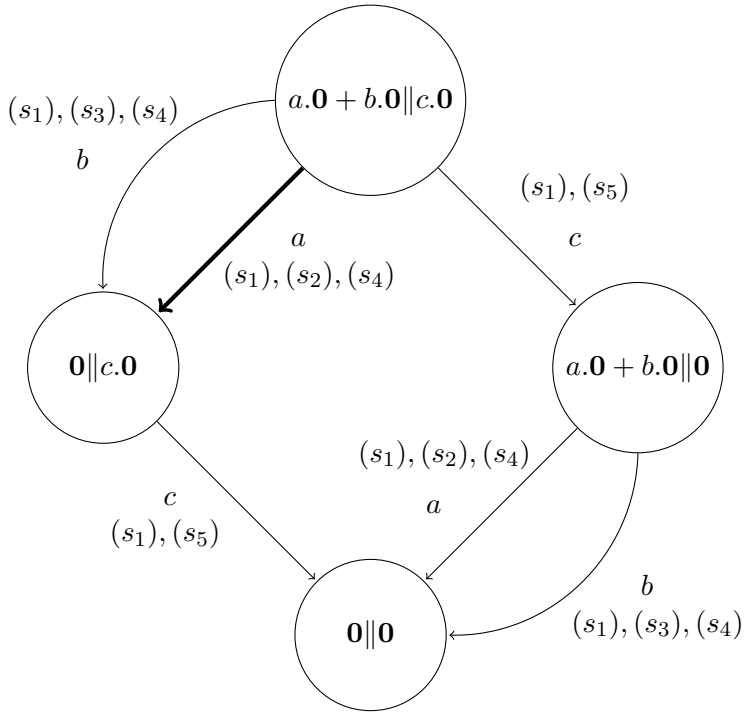

Figure 2.2: State transition diagram constructed from the operational semantics of the process $a.\mathbf{0} + b.\mathbf{0} \| c.\mathbf{0}$. The s-numbers in brackets indicate the rules used for each transition. The thicker line is highlighted because the proof for this transition is shown in fig. 2.1

$(\cdot)$: *The operator* $\cdot : \mathcal{A} \times \mathcal{P}(\mathcal{A}^*) \to \mathcal{P}(\mathcal{A}^*)$ *(pronounced as 'before') binds stronger than* $\cup$ *and is defined as:*

$$a \cdot L = \{aw \mid w \in L\} \tag{2.1}$$

where $a \in \mathcal{A}$ *and* $L \subseteq \mathcal{A}^*$.

$(_*)$: *Via the operator* $\cdot$ *we define the operator* $_* : \mathcal{A} \times \mathcal{P}(\mathcal{A}^*) \to \mathcal{P}(\mathcal{A}^*)$ *as:*

$$a_*L = a \cdot L \cup \{\epsilon\} \tag{2.2}$$

where $a \in \mathcal{A}$ *and* $L \subseteq \mathcal{A}^*$ *again.*

$(\|)$: *The interleaving operator* $\| : \mathcal{A}^* \times \mathcal{A}^* \to \mathcal{P}(\mathcal{A}^*)$*, is recursively defined as:*

$$\begin{aligned} v \,\|\, \epsilon &:= \{v\} \\ \epsilon \,\|\, w &:= \{w\} \\ av \,\|\, bw &:= a \cdot (v \,\|\, bw) \cup b \cdot (av \,\|\, w) \end{aligned} \tag{2.3}$$

where $v, w \in \mathcal{A}^*$ *and* $a, b \in \mathcal{A}$

$(\|)$: *Via this operator we define an interleaving operator* $\| : \mathcal{P}(\mathcal{A}^*) \times \mathcal{P}(\mathcal{A}^*) \to \mathcal{P}(\mathcal{A}^*)$*:*

$$L \| M := \bigcup_{v \in L, w \in M} v \,\|\, w \tag{2.4}$$

where $L, M \in \mathcal{P}(\mathcal{A}^*)$

2.4 Homomorphism

One of the main efforts of this thesis is trying to show that $\mathtt{T}$ is a homomorphism between processes and their sets of traces. A *homomorphism* between two algebraic structures is a map that preserves the operations in these structures. For this thesis those two algebraic structures are the grammar of $\text{BCCSP}_\|$ terms and languages. Specifically, we show in lemma 9 that $\mathtt{T}$ is a homomorphism by showing that the following relations hold:

$$\begin{aligned} \mathtt{T}(\mathbf{0}) &= \{\epsilon\} \\ \mathtt{T}(a.x) &= a_* f(x) \\ \mathtt{T}(x + y) &= x \cup y \\ \mathtt{T}(x \| y) &= x \| y \end{aligned}$$

2.5 Substitution

A substitution σ is a function with type $\mathbb{P}(\mathcal{V}) \to \mathbb{P}(\mathcal{V}')$ that satisfies the following equations:

$$\begin{aligned}\sigma(\mathbf{0}) &= \mathbf{0}\\ \sigma(v) &= z\\ \sigma(a.x) &= a.\sigma(x)\\ \sigma(x+y) &= \sigma(x)+\sigma(y)\\ \sigma(x\|y) &= \sigma(x)\|\sigma(y)\end{aligned}$$

with x and y any term in $\mathbb{P}(\mathcal{V})$, v any variable in $\mathcal{V}$. A substitution is therefore defined by its mapping from variables to terms. A *closed substitution* maps every variable to a closed term, so the type of a closed substitution becomes $\mathbb{P}(\mathcal{V}) \to \mathbb{P}$.

2.6 Equations and Their Logic

An equation is a pair (t, u) denoted by $t \approx u$ where we say that t is equationally similar to u. We want the semantics of equations to have the properties reflexivity, symmetry, transitivity, substitution and closure under $\text{BCCSP}_{\|}$ contexts. To achieve that, we equip equations with the rules of equational logic shown in table 2.2. An equation $t \approx u$ is now *derivable* from a set of equations $\mathcal{E}$ when $t \approx u$ can be proven using the rules of equational logic and $\mathcal{E}$. We will use the notation $\mathcal{E} \vdash t \approx u$ to say that $t \approx u$ is derivable from $\mathcal{E}$.

$$(e_1)\ \ t \approx t \qquad (e_2)\ \ \frac{t \approx u}{u \approx t} \qquad (e_3)\ \ \frac{t \approx u \quad u \approx v}{t \approx v} \qquad (e_4)\ \ \frac{t \approx u}{\sigma(t) \approx \sigma(u)}$$

$$(e_5)\ \ \frac{t \approx u}{a.t \approx a.u} \qquad (e_6)\ \ \frac{t \approx u \quad t' \approx u'}{t + t' \approx u + u'} \qquad (e_7)\ \ \frac{t \approx u \quad t' \approx u'}{t\|t' \approx u\|u'}$$

Table 2.2: Rules of equational logic

2.7 Soundness and Ground-Completeness

The equation $t \approx u$ with t and u $\text{BCCSP}_{\|}$ terms is said to be *sound* modulo $\mathtt{T}$ if $\sigma(t) \ \sim_T \ \sigma(u)$ for all closed substitutions σ. An axiom system $\mathcal{E}$ is a set of equations. This axiom system $\mathcal{E}$ is sound modulo $\mathtt{T}$ if and only if all of its equations are sound modulo $\mathtt{T}$.

For ground-completeness, we say that $\mathcal{E}$ is ground-complete modulo $\mathtt{T}$ if $p \ \sim_T \ q$ implies $\mathcal{E} \vdash p \approx q$ for all closed terms p, q.

When an axiom system is sound and ground-complete modulo some behavioural equivalence we call it an axiomatisation of that behavioural equivalence.

2.8 The Language BCCSP

The process algebra $\text{BCCSP}_{\|}$ is an extension of the language BCCSP from [8]. BCCSP has the same grammar and semantics as $\text{BCCSP}_{\|}$, except it does not have the interleaving parallel operator $\|$. Specifically, the grammar for BCCSP terms is:

$$t ::= \mathbf{0} \mid v \mid a.t \mid t + t$$

where again a ranges over a finite set of actions $\mathcal{A}$ and v ranges over a countably infinite set of variables $\mathcal{V}$. The semantics of BCCSP are defined by the rules (s_1), (s_2) and (s_3). We will use the $\mathbb{B}$ symbol to represent the set of all closed BCCSP terms. Since the derivation rules of BCCSP are included entirely in $\text{BCCSP}_{\|}$, any process in $\text{BCCSP}_{\|}$ without a $\|$ operator is also a BCCSP term. An important property of BCCSP is that the following axiom system $\mathcal{E}_{\mathbb{B}}$ is ground-complete modulo trace equivalence for BCCSP [8]:

$$\mathcal{E}_{\mathbb{B}} = \{(A0), (A1), (A2), (A3), (P0), (P1), (TD)\} \tag{2.5}$$

Here the elements $(A0), \ldots, (TD)$ are defined in table 3.1.

Theorem 1. *The set of axioms $\mathcal{E}_{\mathbb{B}}$ is ground-complete modulo trace equivalence:*

$$\forall p, q \in \mathbb{B}, p \sim_T q \implies \mathcal{E}_{\mathbb{B}} \vdash p \approx q$$

Chapter 3

The Proofs

In this chapter we proof the soundness and ground-completeness modulo trace semantics of the axiom system shown in table 3.1 for $\text{BCCSP}_{\|}$. We begin by setting up some preliminary lemmas. Then we use those for a proof that `T` is a homomorphism. From this the soundness modulo trace semantics quickly follows. For ground-completeness modulo trace semantics we reduce to BCCSP terms. Completeness then follows from this reduction together with the soundness of the axiom system. Finally we introduce a different semantic equivalence called completed trace semantics and proof ground-completeness modulo completed trace semantics using a similar method as for regular trace semantics.

3.1 Soundness

In this thesis we prove that the set of axioms $\mathcal{E}_T$ from [1], shown here in table 3.1, is sound and ground-complete modulo trace equivalence. For the soundness of the axiom system we first prove some properties of our newly defined operators. Then we move on with a proof that there exists a homomorphism between these operators and the grammar of processes. We do this by giving a lemma for each of the operators and the $\mathbf{0}$ symbol from our grammar and show that `T` maps each of these to an operator for languages as mentioned before in section 2.4. Next we show that the equations from

$(A0)$	$x + \mathbf{0} \approx x$	$(A1)$	$x + y \approx y + x$
$(A2)$	$(x + y) + z \approx x + (y + z)$	$(A3)$	$x + x \approx x$
$(P0)$	$x \| \mathbf{0} \approx x$	$(P1)$	$x \| y \approx y \| x$
(TD)	$a.x + a.y \approx a.(x + y)$	(TP)	$(x + y) \| z \approx x \| z + y \| z$
$(EL1)$	$a.x \| b.y \approx a.(x \| b.y) + b.(a.x \| y)$		

Table 3.1: the set of axioms $\mathcal{E}_T$

$\mathcal{E}_T$ translated to languages hold for all languages of which ϵ is an element of that language. Finally in this section we prove the actual soundness.

3.1.1 Preliminary Lemmas

We start with some preliminary lemmas that give some simple properties of $\|$ that we need for some of the other lemmas. We also give show that `T` could be defined recursively by giving such an identity. The first lemma will be useful for the soundness of axiom (P1). It says that $\|$ is a symmetric relation.

Lemma 1. *For all sequences $v, w \in \mathcal{A}^*$, the following property holds:*

$$v \,\|\, w = w \,\|\, v$$

Proof. We inspect cases for v and w:

- $\mathbf{v = \epsilon \vee w = \epsilon}$:
  If $v = \epsilon$ then $v \,\|\, w = \{w\} = w \,\|\, v$. If $w = \epsilon$ then $v \,\|\, w = \{v\} = w \,\|\, v$.
- $\mathbf{v = av' \wedge w = bw'}$:
  If both $v = av'$ and $w = bw'$ with arbitrary $a, b \in \mathcal{A}$ and $v', w' \in \mathcal{A}^*$ then $v \,\|\, w = av' \,\|\, bw' = a \cdot (v' \,\|\, bw') \cup b \cdot (av' \,\|\, w') = b \cdot (av' \,\|\, w') \cup a \cdot (v' \,\|\, bw') = bw' \,\|\, av' = w \,\|\, v$.

□

The next lemma says that $a \cdot (v \,\|\, w)$ is a subset of both $av \,\|\, w$ and $v \,\|\, aw$ for arbitrary action a and sequences v and w. We use it in lemma 8.

Lemma 2. *For all actions $a \in \mathcal{A}$ and sequences $v, w \in \mathcal{A}^*$, the following property holds:*

$$a \cdot (v \,\|\, w) \subseteq av \,\|\, w \cap v \,\|\, aw$$

Proof. Take arbitrary $a \in \mathcal{A}, v, w \in \mathcal{A}^*$. We inspect cases for v and w:

- $\mathbf{v = \epsilon \wedge w = \epsilon}$:
  If both v and w are ϵ then it follows that $a \cdot (v \,\|\, w)$, $v \,\|\, aw$ and $av \,\|\, w$ are all equal to $\{a\}$ so the lemma holds.
- $\mathbf{v = bv' \wedge w = \epsilon}$:
  If only w is ϵ and $v = bv'$ with arbitrary $b \in \mathcal{A}$ and $v' \in \mathcal{A}^*$, then we see

$$\begin{aligned} a \cdot (v \,\|\, w) = a \cdot (v \,\|\, \epsilon) = a \cdot \{v\} &= \underline{\{av\}} \\ av \,\|\, w = av \,\|\, \epsilon &= \underline{\{av\}} \\ v \,\|\, aw = bv' \,\|\, aw &= b \cdot (v' \,\|\, aw) \cup a \cdot (bv' \,\|\, w) \\ &= b(v' \,\|\, aw) \cup a \cdot \{bv'\} = b \cdot (v' \,\|\, aw) \cup \underline{\{av\}} \end{aligned}$$

Since the underlined part of each set is equal to $a \cdot (v \,||\, w)$ it follows that both $a \cdot (v \,||\, w) \subseteq av \,||\, w$ and $a \cdot (v \,||\, w) \subseteq v \,||\, aw$.

- $\mathbf{v = \epsilon \wedge w = bw'}$:
  If instead v is ϵ and $w = bw'$ with arbitrary $b \in \mathcal{A}$ and $w' \in \mathcal{A}^*$, the proof is analogous to the previous case because of symmetry.

- $\mathbf{v = bv' \wedge w = cw'}$:
  If both $v = bv'$ and $w = cw'$ then the lemma follows from expanding the definitions of the three sets:

$$\begin{aligned} a \cdot (v \,||\, w) &= \underline{a \cdot (bv' \,||\, cw')} \\ av \,||\, w = abv' \,||\, cw' &= \underline{a \cdot (bv' \,||\, cw')} \cup c \cdot (abv' \,||\, w') \\ v \,||\, aw = bv' \,||\, acw' &= b \cdot (v' \,||\, acw') \cup \underline{a \cdot (bv' \,||\, cw')} \end{aligned}$$

  Since once again the underlined part of each set is equal to $a \cdot (v \,||\, w)$. It follows that both $a \cdot (v \,||\, w) \subseteq av \,||\, w$ and $a \cdot (v \,||\, w) \subseteq v \,||\, aw$.

□

The next lemma says that if a sequence au starts with an action a and is an element of $v \,||\, w$, that there exist two cases. Namely $v = av'$ starts with an a and then the rest of the word u is an element of $v' \,||\, w$ or the symmetric other case. We also use this lemma in lemma 8.

Lemma 3. *For all actions $a \in \mathcal{A}$ and sequences $u, v, w \in \mathcal{A}^*$, the following property holds:*

$$au \in v \,||\, w \implies$$
$$\Big(\exists v' \in \mathcal{A}^*, v = av' \wedge u \in v' \,||\, w\Big) \vee \Big(\exists w' \in \mathcal{A}^*, w = aw' \wedge u \in v \,||\, w'\Big)$$

Proof. Take arbitrary $u, v, w \in \mathcal{A}^*$ and $a \in \mathcal{A}$ such that $au \in v \,||\, w$. Note that v and w can not both be ϵ at the same time, as that would mean $au \in \{v \,||\, w\} = \{\epsilon\}$, which is false. We separate remaining cases for v and w:

- $\mathbf{v = \epsilon \wedge w = bw'}$:
  If $v = \epsilon$ and $w = bw'$ with arbitrary $b \in \mathcal{A}$ and $w' \in \mathcal{A}^*$, then $v \,||\, w = \{w\} = \{bw'\}$. Since $au \in v \,||\, w$ that means $au \in \{bw'\}$. Clearly this can only be the case if $a = b$ and $u = w'$. Therefore $\exists w' \in \mathcal{A}^*, w = aw'$ and $u \in \{u\} = \{w'\} = \epsilon \,||\, w' = v \,||\, w'$, so the conclusion of the lemma becomes true as well.

- $\mathbf{v = bv' \wedge w = \epsilon}$:
  The case where $v = bv'$ and $w = \epsilon$ with arbitrary $b \in \mathcal{A}$ and $v' \in \mathcal{A}^*$ is analogous to the previous case because of symmetry.

- $\mathbf{v = bv' \wedge w = cw'}$:
  Take $v = bv'$ and $w = cw'$ with arbitrary $b, c \in \mathcal{A}$ and $v', w' \in \mathcal{A}^*$. Now $v \,||\, w = bv' \,||\, cw' = b \cdot (v' \,||\, cw') \cup c \cdot (bv' \,||\, w')$. Since $au \in v \,||\, w$ there are two cases:
  - $\mathbf{au \in b \cdot (v' \,||\, cw')}$:
    If $au \in b \cdot (v' \,||\, cw')$ then it is clear that $a = b$ so $\exists v' \in \mathcal{A}^*, v = av'$. From the premise of this case it is also clear that $u \in v' \,||\, cw' = v' \,||\, w$. Therefore the conclusion of the lemma also follows.
  - $\mathbf{au \in c \cdot (bv' \,||\, w')}$:
    If $au \in c \cdot (bv' \,||\, w')$ then it is similarly clear that $a = c$ so $\exists w' \in \mathcal{A}^*, w = aw'$. From the premise of this case it is also similarly clear that $u \in bv' \,||\, w' = v \,||\, w'$. Again the conclusion of the lemma follows.

□

The next lemma can be seen as a recursive definition of $\mathtt{T}$ rather than as a lemma. Since recursive definitions usually demand a set to be the smallest set for which a property holds, we do that here as well. The lemma is useful to get a sense of the structure of $\mathtt{T}$. It is also used in the lemmas that work towards a proof of the existence of a homomorphism between processes and sets of traces, namely lemmas 6, 7 and 8.

In natural language, the lemma says that the set of traces of a process is the empty trace combined with the sets of traces from the subprocesses where each element of that set is preceded with the action that led to the subprocess from the original process.

Lemma 4. *$\mathtt{T}$ is the smallest function of type $\mathbb{P} \to \mathcal{P}(\mathcal{A}^*)$ with the property*

$$\forall x \in \mathbb{P}, \mathtt{T}(x) = \left(\bigcup_{x \xrightarrow{a} x'} a \cdot \mathtt{T}(x') \right) \cup \{\epsilon\}$$

Here 'smallest' means that for all processes x in the domain of $\mathtt{T}$, $\mathtt{T}(x) \subseteq f(x)$ for any function f of type $\mathbb{P} \to \mathcal{P}(\mathcal{A}^*)$ with the same property.

Proof. First we reformulate the property so that we can talk about the sequences that are elements of both languages. The property becomes

$$\forall w \in \mathcal{A}^*, \forall x \in \mathbb{P}, w \in \mathtt{T}(x) \iff w \in \left(\bigcup_{x \xrightarrow{a} x'} a \cdot \mathtt{T}(x') \right) \cup \{\epsilon\}$$

Now we show that this property holds for $\mathtt{T}$. We inspect two cases for w.

- $\mathbf{w = \epsilon}$:
  If $w = \epsilon$, then $w \in \mathtt{T}(x)$ because the empty trace is always possible from any process. from $w \in \{\epsilon\}$ it follows that $w \in \left(\bigcup_{x \xrightarrow{a} x'} a \cdot \mathtt{T}(x')\right) \cup \{\epsilon\}$. So both sides of the equivalence are true and therefore also the equivalence relation itself.

- $\mathbf{w = bw'}$:
  If w is some bw' we separate two cases of the equivalence relation.

  - $\mathbf{w \in \mathtt{T}(x) \implies w \in \left(\bigcup_{x \xrightarrow{a} x'} a \cdot \mathtt{T}(x')\right) \cup \{\epsilon\}}$:
    If $w \in \mathtt{T}(x)$ then we know some p exists such that $x \xrightarrow{w} p$. Furthermore, since $w = bw'$ we can break this up into two steps to conclude that there exists some p' such that

    $$x \xrightarrow{b} p' \xrightarrow{w'} p$$

    Now we can see that $w' \in \mathtt{T}(p')$ and $x \xrightarrow{b} p'$. From $w' \in \mathtt{T}(p')$, it follows that $w = bw' \in b \cdot \mathtt{T}(p')$. Together with $x \xrightarrow{b} p'$ it now follows that

    $$w \in b \cdot \mathtt{T}(p') \subseteq \left(\bigcup_{x \xrightarrow{a} x'} a \cdot \mathtt{T}(x') \right) \cup \{\epsilon\}$$

  - $\mathbf{w \in \left(\bigcup_{x \xrightarrow{a} x'} a \cdot \mathtt{T}(x')\right) \cup \{\epsilon\} \implies w \in \mathtt{T}(x)}$:
    If instead $w \in \left(\bigcup_{x \xrightarrow{a} x'} a \cdot \mathtt{T}(x')\right) \cup \{\epsilon\}$ we know that $w \notin \{\epsilon\}$ as $w \neq \epsilon$. Therefore there exists some p' and c such that $w \in c \cdot \mathtt{T}(p')$ and $x \xrightarrow{c} p'$. For $w = bw' \in c \cdot \mathtt{T}(p')$ to be possible, c has to be b, as all sequences in $c \cdot \mathtt{T}(p')$ start with c while bw' starts with b. It now follows from $bw' \in b \cdot \mathtt{T}(p')$ that $w' \in \mathtt{T}(p')$. From $w' \in \mathtt{T}(p')$ it follows that there exists some p such that $p' \xrightarrow{w'} p$. We already knew that $x \xrightarrow{b} p'$ so together we get

    $$x \xrightarrow{b} p' \xrightarrow{w'} p$$

    Therefore $w = bw' \in \mathtt{T}(x)$.

Next we have to show T is the *smallest* function with this property. Take an arbitrary function f with the following property:

$$\forall w \in \mathcal{A}^*, \forall x \in \mathbb{P}, w \in f(x) \iff w \in \left(\bigcup_{x \xrightarrow{a} x'} a \cdot f(x') \right) \cup \{\epsilon\}$$

Now we would have to prove

$$\forall w \in \mathcal{A}^*, \forall x \in \mathbb{P}, w \in \mathtt{T}(x) \implies w \in f(x)$$

Take arbitrary x. We will use induction over words w to prove the property for all words.

Base case:
We assume $w = \epsilon \in \mathtt{T}(x)$. From $w \in \{\epsilon\}$ it follows that

$$w \quad \in \quad \{\epsilon\} \quad \subseteq \quad \left(\bigcup_{x \xrightarrow{a} x'} a \cdot f(x') \right) \cup \{\epsilon\}$$

And using the property of f therefore also $w \in f(x)$.

Inductive case:
We assume $w = bw' \in \mathtt{T}(x)$. The IH becomes:

$$\forall x \in \mathbb{P}, w' \in \mathtt{T}(x) \implies w' \in f(x)$$

Using the property of $\mathtt{T}$ and the fact that $w \neq \epsilon$ it follows that

$$w \in \bigcup_{x \xrightarrow{a} x'} a \cdot \mathtt{T}(x')$$

Therefore there exists c and x' such that $w \in c \cdot \mathtt{T}(x')$ and $x \xrightarrow{c} x'$. Since $w = bw' \in c \cdot \mathtt{T}(x')$ it follows that $b = c$ and $w' \in \mathtt{T}(x')$ again. Now using the IH it follows that $w' \in f(x')$. Since $w = bw'$ and $x \xrightarrow{b} x'$ we can combine this to see that

$$w = bw' \quad \in \quad b \cdot f(x') \quad \subseteq \quad \left(\bigcup_{x \xrightarrow{a} x'} a \cdot f(x') \right) \cup \{\epsilon\}$$

Using the property of f it now follows that $w \in f(x)$

□

3.1.2 Proof that T is a Homomorphism

For the soundness of $\mathcal{E}_T$ we show that $\mathtt{T}$ is a homomorphism. The following four lemmas all deal with one of the symbols of the grammar of processes.

The first of these lemmas shows that the process $\mathbf{0}$ corresponds with the language $\{\epsilon\}$.

Lemma 5. *The following property holds:*

$$\mathcal{T}(\mathbf{0}) = \{\epsilon\}$$

Proof. Since $\mathbf{0}$ has no possible transitions from it, its only trace is the empty trace. □

The second of these lemmas shows that $a.$ on processes corresponds with a_* on languages for every action a.

Lemma 6. *For all actions $a \in \mathcal{A}$ and processes $x \in \mathbb{P}$ the following property holds:*

$$\mathcal{T}(a.x) = a_* \mathcal{T}(x)$$

Proof. Take arbitrary a and x. The process $a.x$ only has one transition according to the operational semantics:

$$\frac{}{a.x \xrightarrow{a} x}\ (s_1)$$

Therefore using lemma 4 it follows that the following holds.

$$\mathtt{T}(a.x) \quad = \quad \left(\bigcup_{a.x \xrightarrow{a} x} a \cdot f(x) \right) \cup \{\epsilon\} \quad = \quad a \cdot \mathtt{T}(x) \cup \{\epsilon\} \quad = \quad a_* \mathtt{T}(x)$$

□

The third of these lemmas shows that $+$ on processes corresponds with $\cup$ on languages.

Lemma 7. *For all processes $x, y \in \mathbb{P}$ the following property holds:*

$$\mathcal{T}(x + y) = \mathcal{T}(x) \cup \mathcal{T}(y)$$

Proof. Take arbitrary $x, y \in \mathbb{P}$. Only two rules of the semantics apply directly to the process $x + y$:

$$\frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'}\ (s_2) \qquad \frac{y \xrightarrow{b} y'}{x + y \xrightarrow{b} y'}\ (s_3)$$

Take arbitrary $w \in \mathcal{A}^*$. If $w \in \bigcup_{x+y \xrightarrow{c} p} c \cdot \mathtt{T}(p)$ then there exists c and p such that $x + y \xrightarrow{c} p$ and $w \in c \cdot \mathtt{T}(p)$. Hence, $x \xrightarrow{c} p$ from (s_2) or $y \xrightarrow{c} p$ from (s_3), which results in $w \in \left(\bigcup_{x \xrightarrow{c} p} c \cdot \mathtt{T}(p) \right) \cup \left(\bigcup_{y \xrightarrow{c} p} c \cdot \mathtt{T}(p) \right)$

If instead $w \in \left(\bigcup_{x \xrightarrow{c} p} c \cdot \mathtt{T}(p) \right) \cup \left(\bigcup_{y \xrightarrow{c} p} c \cdot \mathtt{T}(p) \right)$ then similarly it follows that there exists c and p such that $x \xrightarrow{c} p$ or $y \xrightarrow{c} p$ which in both cases leads to $x + y \xrightarrow{c} p$. Therefore now $w \in \bigcup_{x+y \xrightarrow{c} p} c \cdot \mathtt{T}(p)$ Together it follows that

$$w \in \bigcup_{x+y \xrightarrow{c} p} c \cdot \mathtt{T}(p) \quad \iff \quad w \in \left(\bigcup_{x \xrightarrow{c} p} c \cdot \mathtt{T}(p) \right) \cup \left(\bigcup_{y \xrightarrow{c} p} c \cdot \mathtt{T}(p) \right) \tag{3.1}$$

Using lemma 4 it now follows that

$$\begin{aligned}
\mathtt{T}(x+y) &= \left(\bigcup_{x+y\xrightarrow{c}p} c\cdot\mathtt{T}(p)\right)\cup\{\epsilon\} && \text{(lemma 4)}\\
&= \left(\bigcup_{x\xrightarrow{c}p} c\cdot\mathtt{T}(p)\right)\cup\left(\bigcup_{y\xrightarrow{c}p} c\cdot\mathtt{T}(p)\right)\cup\{\epsilon\} && (3.1)\\
&= \left[\left(\bigcup_{x\xrightarrow{c}p} c\cdot\mathtt{T}(p)\right)\cup\{\epsilon\}\right]\cup\left[\left(\bigcup_{y\xrightarrow{c}p} c\cdot\mathtt{T}(p)\right)\cup\{\epsilon\}\right]\\
&= \mathtt{T}(x)\cup\mathtt{T}(y) && \text{(lemma 4)}
\end{aligned}$$

□

The fourth and last of these lemmas shows that $\|$ on processes corresponds with $\|$ on sets of traces.

Lemma 8. *For all $x, y \in \mathbb{P}$, the following property holds:*

$$\mathcal{T}(x\|y) = \mathcal{T}(x)\|\mathcal{T}(y)$$

Proof. Take arbitrary processes x and y. We use induction on sequences w to prove the rewritten property $w \in \mathtt{T}(x\|y) \iff w \in \mathtt{T}(x)\|\mathtt{T}(y)$ for all sequences $w \in \mathcal{A}^*$.

Base case:
For $w = \epsilon$ notice that $w \in \mathtt{T}(x\|y), w \in \mathtt{T}(x)$ and $w \in \mathtt{T}(y)$ because the empty trace is always a possible trace from any process. Now it becomes clear that $w \in \mathtt{T}(x)\|\mathtt{T}(y)$:

$$w = \epsilon \quad\in\quad \epsilon \between \epsilon \quad\subseteq\quad \bigcup_{u\in\mathtt{T}(x), v\in\mathtt{T}(y)} u \between v = \mathtt{T}(x)\|\mathtt{T}(y)$$

Inductive case:
For the inductive case $w = aw'$ we have the following IH:

$$\forall x, y \in \mathbb{P}, w' \in \mathtt{T}(x\|y) \overset{\text{IH}}{\iff} w' \in \mathtt{T}(x)\|\mathtt{T}(y)$$

We separate the equivalence relation in two cases

- $\mathbf{w \in \mathtt{T}(x\|y) \implies w \in \mathtt{T}(x)\|\mathtt{T}(y)}$:
  If $w \in \mathtt{T}(x\|y)$, then there exists p such that $x\|y \xrightarrow{aw'} p$ so also p' such that
  $$x\|y \xrightarrow{a} p' \xrightarrow{w'} p$$
  From $x\|y \xrightarrow{a} p'$ and the operational semantics we have two possible cases:

* $\mathbf{p' = x' \| y \wedge x \xrightarrow{a} x'}$:
For the case that follows from (s_4), $p' = x' \| y$ and $x \xrightarrow{a} x'$ hold. Hence we can write

$$x \| y \xrightarrow{a} x' \| y \xrightarrow{w'} p$$

Therefore we also know that $x' \| y \xrightarrow{w'} p$, so $w' \in \mathtt{T}(x' \| y)$. Using the IH we now see that $w' \in \mathtt{T}(x') \| \mathtt{T}(y)$. This means that there exists $t \in \mathtt{T}(x'), u \in \mathtt{T}(y)$ such that $w' \in t \,\|\!|\, u$ from (2.4). Since $w = aw'$ it follows that $w \in a \cdot (t \,\|\!|\, u)$. Using lemma 2 it follows that $w \in at \,\|\!|\, u$. Note that, since $t \in \mathtt{T}(x')$, this implies there exists a q such that $x' \xrightarrow{t} q$. Combining this with the fact from our assumption that $x \xrightarrow{a} x'$, it follows that:

$$x \xrightarrow{a} x' \xrightarrow{t} q$$

And therefore $at \in \mathtt{T}(x)$. Now it becomes clear that $w \in \mathtt{T}(x) \| \mathtt{T}(y)$:

$$w \quad \in \quad at \,\|\!|\, u \quad \subseteq \quad \bigcup_{r \in \mathtt{T}(x), s \in \mathtt{T}(y)} r \,\|\!|\, s = \mathtt{T}(x) \| \mathtt{T}(y)$$

* $\mathbf{p' = x \| y' \wedge y \xrightarrow{a} y'}$:
For the case that follows from (s_5), $p' = x \| y'$ and $y \xrightarrow{a} y'$ hold. Because of the symmetry of the equations and the lemma, the proof of this case is analogous to the previous case.

– $\mathbf{w \in \mathtt{T}(x) \| \mathtt{T}(y) \implies w \in \mathtt{T}(x \| y)}$:
If $w \in \mathtt{T}(x) \| \mathtt{T}(y)$ then there exists some $t \in \mathtt{T}(x), u \in \mathtt{T}(y)$ such that $w \in t \,\|\!|\, u$. Now, since $w = aw'$, from lemma 3 it follows that there are two cases.

* $\mathbf{\exists t', w' \in t' \,\|\!|\, u \wedge t = at'}$:
For the first case $\exists t', w' \in t' \,\|\!|\, u \wedge t = at'$ holds. Take this t'. Since $t = at' \in \mathtt{T}(x)$ it follows that x' and q exists such that:

$$x \xrightarrow{a} x' \xrightarrow{t'} q$$

So $t' \in \mathtt{T}(x')$. So now it becomes clear that $w' \in \mathtt{T}(x') \| \mathtt{T}(y)$:

$$w' \quad \in \quad t' \,\|\!|\, u \quad \subseteq \quad \bigcup_{u \in \mathtt{T}(x'), v \in \mathtt{T}(y)} u \,\|\!|\, v = \mathtt{T}(x') \| \mathtt{T}(y)$$

Now we can use the IH to conclude that $w' \in \mathtt{T}(x' \| y)$. We already know that $x \xrightarrow{a} x'$ so using rule (s_4) we see that

$x\|y \xrightarrow{a} x'\|y$. Together with lemma 4 we see:

$$w = aw' \in a \cdot \mathtt{T}(x'\|y) \subseteq \left(\bigcup_{x\|y \xrightarrow{b} p} b \cdot \mathtt{T}(p) \right) \cup \{\epsilon\} = \mathtt{T}(x\|y)$$

$*$ $\exists \mathbf{u'}, \mathbf{aw'} \in \mathbf{a} \cdot (\mathbf{t} \coprod \mathbf{u'}) \wedge \mathbf{u} = \mathbf{au'}$:
The case where $\exists u', aw' \in a \cdot (t \coprod u') \wedge u = au'$ holds is analogous to the previous case because of symmetry.

□

Next we combine the last few lemmas to prove that T is indeed a homomorphism.

Lemma 9. *T is a homomorphism such that it preserves the algebraic structure between processes and languages according to the following relations:*

$$f(\mathbf{0}) = \{\epsilon\} \tag{3.2}$$
$$f(a.x) = a_* f(x) \tag{3.3}$$
$$f(x + y) = x \cup y \tag{3.4}$$
$$f(x\|y) = x\|y \tag{3.5}$$

Proof. The lemmas 5 - 8 prove the relations 3.2 - 3.5 respectively. □

3.1.3 Soundness

In the previous subsection we showed that T is a homomorphism from processes to languages. Now we create a new axiom system E_g derived from $\mathcal{E}_T$, where all processes are replaced with languages and the operations on those terms are replaced with the corresponding operations according to the homomorphism T. The axiom system that follows from this transformation is displayed in table 3.2. The following lemma then proves that the equations that follow from this replacing of terms hold for all languages with $\{\epsilon\}$ as an element.

$(A0_g)$	$L \cup \{\epsilon\} = L$	$(A1_g)$	$L \cup M = M \cup L$
$(A2_g)$	$(L \cup M) \cup N = L \cup (M \cup N)$	$(A3_g)$	$L \cup L = L$
$(P0_g)$	$L\|\{\epsilon\} = L$	$(P1_g)$	$L\|M = M\|L$
(TD_g)	$(a_*L) \cup (a_*M) = a_*(L \cup M)$	(TP_g)	$(L \cup M)\|N = L\|N \cup M\|N$
$(EL1_g)$	$(a_*L)\|(b_*M) = a_*\Big(L\|(b_*M)\Big) \cup b_*\Big((a_*L)\|M\Big)$		

Table 3.2: the set of equations E_g that hold for any language of which $\{\epsilon\}$ is an element

Lemma 10. *The set of equations E_g in table 3.2 is valid if $L, M, N \in \mathcal{P}(\mathcal{A}^*)$ are arbitrary languages and ϵ is an element of each language.*

Proof. $(A0_g)$ follows from our assumption that $\epsilon \in L$. $(A1_g)$, $(A2_g)$ and $(A3_g)$ follow from the associativity, commitivity and idempotency of $\cup$ respectively. The rest we separate:

$(P0_g)$:

$$\begin{aligned} L \| \{\epsilon\} &= \bigcup_{v \in L, w \in \{\epsilon\}} v \,\|\, w && (2.4) \\ &= \bigcup_{v \in L} v \,\|\, \epsilon \\ &= \bigcup_{v \in L} \{v\} && (2.3) \\ &= L \end{aligned}$$

$(P1_g)$:

$$\begin{aligned} L \| M &= \bigcup_{v \in L, w \in M} v \,\|\, w && (2.4) \\ &= \bigcup_{w \in M, v \in L} w \,\|\, v && \text{(lemma 1)} \\ &= M \| L \end{aligned}$$

(TD_g):

$$\begin{aligned} (a_* L) \cup (a_* M) &= (a \cdot L \cup \{\epsilon\}) \cup (a \cdot M \cup \{\epsilon\}) && (2.2) \\ &= (a \cdot L \cup a \cdot M) \cup \{\epsilon\} \\ &= a \cdot (L \cup M) \cup \{\epsilon\} && (2.1) \\ &= a_*(L \cup M) && (2.2) \end{aligned}$$

(TP_g):

$$\begin{aligned} (L \cup M) \| N &= \bigcup_{v \in L \cup M, w \in N} v \,\|\, w && (2.4) \\ &= \left(\bigcup_{v \in L, w \in N} v \,\|\, w \right) \cup \left(\bigcup_{v \in M, w \in N} v \,\|\, w \right) \\ &= L \| N \cup M \| N && (2.4) \end{aligned}$$

$(EL1_g)$:

$$
\begin{aligned}
&(a_*L)\|(b_*M)\\
&= (a\cdot L\cup\{\epsilon\})\|(b\cdot M\cup\{\epsilon\}) && (2.2)\\
&= \bigcup_{v\in(a\cdot L\cup\{\epsilon\}),w\in(b\cdot M\cup\{\epsilon\})} v\,\|\,w && (2.4)\\
&= \left(\bigcup_{v'\in L,w'\in M} av'\,\|\,bw'\right)\cup\left(\bigcup_{v'\in L} av'\,\|\,\epsilon\right)\cup\left(\bigcup_{w'\in M}\epsilon\,\|\,bw'\right)\cup\epsilon\,\|\,\epsilon\\
&= \left(\bigcup_{v'\in L,w'\in M} a\cdot(v'\,\|\,bw')\cup b\cdot(av'\,\|\,w')\right)\cup\left(\bigcup_{v'\in L}\{av'\}\right)\\
&\quad\cup\left(\bigcup_{w'\in M}\{bw'\}\right)\cup\{\epsilon\} && (2.3)\\
&= \left(\bigcup_{v'\in L,w'\in M} a\cdot(v'\,\|\,bw')\right)\cup\left(\bigcup_{v'\in L,w'\in M} b\cdot(av'\,\|\,w')\right)\\
&\quad\cup\left(\bigcup_{v'\in L} a\cdot\{v'\}\right)\cup\left(\bigcup_{w'\in M} b\cdot\{w'\}\right)\cup\{\epsilon\} && (2.1)\\
&= a\cdot\left(\bigcup_{v'\in L,w'\in M}(v'\,\|\,bw')\right)\cup b\cdot\left(\bigcup_{v'\in L,w'\in M}(av'\,\|\,w')\right)\\
&\quad\cup a\cdot\left(\bigcup_{v'\in L}(v'\,\|\,\epsilon)\right)\cup b\cdot\left(\bigcup_{w'\in M}(\epsilon\,\|\,w')\right)\cup\{\epsilon\} && (2.1)\\
&= a\cdot\left[\left(\bigcup_{v'\in L,w'\in M}(v'\,\|\,bw')\right)\cup\left(\bigcup_{v'\in L}(v'\,\|\,\epsilon)\right)\right]\\
&\quad\cup b\cdot\left[\left(\bigcup_{v'\in L,w'\in M}(av'\,\|\,w')\right)\cup\left(\bigcup_{w'\in M}(\epsilon\,\|\,w')\right)\right]\cup\{\epsilon\}\\
&= a\cdot\left(\bigcup_{v'\in L,w\in b\cdot M\cup\{\epsilon\}}(v'\,\|\,w)\right)\cup b\cdot\left(\bigcup_{v\in a\cdot L\cup\{\epsilon\},w'\in M}(v\,\|\,w')\right)\cup\{\epsilon\}\\
&= a\cdot\Big(L\|(b\cdot M\cup\{\epsilon\})\Big)\cup b\cdot\Big((a\cdot L\cup\{\epsilon\})\|M\Big)\cup\{\epsilon\} && (2.4)\\
&= \Big[a\cdot\Big(L\|(b\cdot M\cup\{\epsilon\})\Big)\cup\{\epsilon\}\Big]\cup\Big[b\cdot\Big((a\cdot L\cup\{\epsilon\})\|M\Big)\cup\{\epsilon\}\Big]\\
&= a_*\Big(L\|(b_*M)\Big)\cup b_*\Big((a_*L)\|M\Big) && (2.2)
\end{aligned}
$$

□

Now we finally prove the soundness of our axiom system using the homomorphism and the fact that the axioms translated using the homomorphism hold.

Theorem 2. *The set of axioms $\mathcal{E}_T$ is sound modulo trace equivalence: For all closed substitutions σ, BCCSP$_\parallel$ terms t and u and axioms in $\mathcal{E}_T$,*

$$\mathcal{E}_T \vdash t \approx u \quad \Longrightarrow \quad \mathcal{T}(\sigma(t)) = \mathcal{T}(\sigma(u))$$

Proof. From lemma 9 it follows that $\mathtt{T}$ is a homomorphism between BCCSP$_\parallel$ terms and their sets of traces:

$$\begin{aligned} \mathtt{T}(\mathbf{0}) &= \{\epsilon\} \\ \mathtt{T}(a.x) &= a_*\mathtt{T}(x) \\ \mathtt{T}(x+y)) &= \mathtt{T}(x) \cup \mathtt{T}(y) \\ \mathtt{T}(x \| y) &= \mathtt{T}(x) \| \mathtt{T}(y) \end{aligned}$$

where x, y are processes. Take arbitrary BCCSP$_\parallel$ terms t, u. Since σ is a closed substitution, $\sigma(t)$ and $\sigma(u)$ are both processes. Since $\epsilon \in \mathtt{T}(x)$ for any x, we can use lemma 10 on sets of traces. The set of equations E_g (table 3.2) is exactly those that you get if you apply $\mathtt{T}$ to each side of each equation from our axiom system $\mathcal{E}_T$ (table 3.1). Therefore each equation of $\mathcal{E}_T$ can be proven by applying T and then using the corresponding equation from E_g which is proven to hold in lemma 10. For example, the equation corresponding to axiom (TD) does indeed hold:

$$\begin{aligned} \mathtt{T}(\sigma(a.x + a.y)) &= \mathtt{T}(a.\sigma(x) + a.\sigma(y))) && (\sigma) \\ &= \mathtt{T}(a.\sigma(x))) \cup \mathtt{T}(a.\sigma(y)) && \text{(lemma 9)} \\ &= (a_*\mathtt{T}(\sigma(x))) \cup (a_*\mathtt{T}(\sigma(y))) && \text{(lemma 9)} \\ &= a_*(\mathtt{T}(\sigma(x) \cup \mathtt{T}(\sigma(y))) && (TD_g) \\ &= a_*(\mathtt{T}(\sigma(x) + \sigma(y))) && \text{(lemma 9)} \\ &= \mathtt{T}(a.(\sigma(x) + \sigma(y))) && \text{(lemma 9)} \\ &= \mathtt{T}(\sigma(a.(x+y))) && (\sigma) \end{aligned}$$

We can use similar reasoning for every equation $t \approx u$ to see that $\mathtt{T}(\sigma(t)) = \mathtt{T}(\sigma(u))$ holds, so therefore the theorem holds. □

3.2 Ground-Completeness

Our proof for ground-completeness is much shorter that that for soundness. We first prove that all processes can be reduced to BCCSP terms with our axiom system $\mathcal{E}_T$. Then we use the ground-completeness of the axiom system $\mathcal{E}_\mathbb{B}$ modulo trace equivalency for BCCSP terms to show that $\mathcal{E}_T$ is indeed ground-complete modulo trace equivalency for BCCSP$_\parallel$ terms.

3.2.1 Reduction to BCCSP

We reduce $\text{BCCSP}_{\|}$ to BCCSP in lemma 13. For this lemma we first need the following two lemmas.

Technically, the ground-completeness of $\mathcal{E}_{\mathbb{B}}$ modulo trace semantics from [8] uses a different definition of the function $\mathtt{T}$. In their proof traces are only defined for BCCSP terms, so this version of the trace function maps only BCCSP terms to their sets of traces. We will call this functions $\mathtt{T}_{\mathbb{B}}$ for now. Clearly, when we apply $\mathtt{T}$ to any BCCSP term, the result will be the same as if we had applied $\mathtt{T}_{\mathbb{B}}$ to that term. In the spirit of the ground-up approach of this thesis, we use a lemma to make it clear that they are indeed the same.

Lemma 11. *For all BCCSP terms $p \in \mathbb{B}$, the following property holds:*

$$\mathtt{T}(p) = \mathtt{T}_{\mathbb{B}}(p)$$

Proof. (s_4) and (s_5) are the only rules that are part of the semantics of $\text{BCCSP}_{\|}$ terms, but not of the semantics of BCCSP terms. Neither rule is applicable on any term $p \in \mathbb{B}$, since BCCSP terms never contain the $\|$ operator. Therefore if we derive the set of all traces using the semantics of either algebra we will get the same result, so the functions are identical for p. □

Since $\mathtt{T}(p) = \mathtt{T}_{\mathbb{B}}(p)$ for BCCSP terms p we simply use $\mathtt{T}$ for both BCCSP terms and $\text{BCCSP}_{\|}$ terms from now on. The next lemma says that if you take two closed BCCSP terms p, q, then there exists a closed BCCSP term equationally similar to $p\|q$. This lemma basically unfolds all the interleaving terms using the axioms (TP) and $(EL1)$ to get a much more complicated term without any interleaving.

Lemma 12. *For all BCCSP terms $p, q \in \mathbb{B}$, there exists another BCCSP term $r \in \mathbb{B}$ such that the following property holds:*

$$\mathcal{E}_T \vdash p\|q \approx r$$

Proof. We use induction over processes p to prove the lemma for all $p \in \mathbb{B}$.

Base case (p):
Take $p \approx \mathbf{0}$. For $r \approx q$ it is now clear that $r \in \mathbb{B}$ and

$$p\|q \approx \mathbf{0}\|q \approx q\|\mathbf{0} \approx q \approx r$$

Inductive cases (p):

- $\mathbf{p \approx p_1 + p_2}$:
  Take $p \approx p_1 + p_2$ with arbitrary $p_1, p_2 \in \mathbb{B}$. The IHs then become:

$$\forall q \in \mathbb{B}, \exists r_1 \in \mathbb{B}, \mathcal{E}_T \vdash p_1\|q \overset{\text{IH}}{\approx} r_1$$

$$\forall q \in \mathbb{B}, \exists r_2 \in \mathbb{B}, \mathcal{E}_T \vdash p_2 \| q \overset{\text{IH}}{\approx} r_2$$

Take these $p_1 \| q \overset{\text{IH}}{\approx} r_1 \in \mathbb{B}$ and $p_2 \| q \overset{\text{IH}}{\approx} r_2 \in \mathbb{B}$. Now if we take $r_1 + r_2 \approx r \in \mathbb{B}$ we see:

$$p \| q \approx (p_1 + p_2) \| q \approx p_1 \| q + p_2 \| q \overset{\text{IH}}{\approx} r_1 + r_2 \approx r$$

– $\mathbf{p \approx a.p'}$
Take $p \approx a.p'$ with arbitrary $a \in \mathcal{A}$ and $p' \in \mathbb{B}$. The IH then becomes:

$$\forall q \in \mathbb{B}, \exists r_p \in \mathbb{B}, \mathcal{E}_T \vdash p' \| q \overset{\text{IH}}{\approx} r_p \tag{3.6}$$

We now need to prove the property

$$\forall q \in \mathbb{B}, \exists r \in \mathbb{B}, \mathcal{E}_T \vdash a.p' \| q \approx r$$

We do this by induction over process q:

Base case (q):
Take $q \approx \mathbf{0}$. For $r \approx a.p'$ it is clear that $r \in \mathbb{B}$ and

$$a.p' \| q \approx a.p' \| \mathbf{0} \approx a.p' \approx r$$

Inductive cases (q):

· $\mathbf{q \approx q_1 + q_2}$:
Take $q \approx q_1 + q_2$ with arbitrary $q_1, q_2 \in \mathbb{B}$. The IHs are:

$$\exists r_1 \in \mathbb{B}, \mathcal{E}_T \vdash a.p' \| q_1 \overset{\text{IH}}{\approx} r_1$$

$$\exists r_2 \in \mathbb{B}, \mathcal{E}_T \vdash a.p' \| q_2 \overset{\text{IH}}{\approx} r_2$$

Take these $a.p' \| q_1 \overset{\text{IH}}{\approx} r_1 \in \mathbb{B}$ and $a.p' \| q_2 \overset{\text{IH}}{\approx} r_2 \in \mathbb{B}$. Now if we take $r_1 + r_2 \approx r \in \mathbb{B}$ we see:

$$\begin{aligned} a.p' \| q &\approx a.p' \| (q_1 + q_2) \\ &\approx (q_1 + q_2) \| a.p' \\ &\approx q_1 \| a.p' + q_2 \| a.p' \\ &\approx a.p' \| q_1 + a.p' \| q_2 \\ &\overset{\text{IH}}{\approx} r_1 + r_2 \\ &\approx r \end{aligned}$$

· $\mathbf{q \approx b.q'}$:
Take $q \approx b.q'$ with arbitrary $b \in \mathcal{A}$ and arbitrary $q' \in \mathbb{B}$. The IH is:

$$\exists r_q \in \mathbb{B}, \mathcal{E}_T \vdash a.p' \| q' \overset{\text{IH}}{\approx} r_q \tag{3.7}$$

Take $p' \| q \approx p' \| b.q' \overset{\text{IH}}{\approx} r_p \in \mathbb{B}$ from the first IH (3.6) and $a.p' \| q' \overset{\text{IH}}{\approx} r_q \in \mathbb{B}$ from the second IH (3.7). Now for $a.r_p + b.r_q \approx r \in \mathbb{B}$ we see:

$$\begin{aligned} a.p' \| q &\approx a.p' \| b.q' \\ &\approx a.(p' \| b.q') + b.(a.p' \| q') \\ &\overset{\text{IH}}{\approx} a.r_p + b.r_q \\ &\approx r \end{aligned}$$

□

The next lemma says that $\text{BCCSP}_{\|}$ can indeed be reduced. Given the previous result this becomes somewhat trivial, as all operators except for $\|$ already are in BCCSP and the case of $\|$ follows directly from the previous lemma.

Lemma 13. *Every $\text{BCCSP}_{\|}$ term is equivalent to a BCCSP term:*

$$\forall p \in \mathbb{P}, \exists q \in \mathbb{B}, \mathcal{E}_T \vdash p \approx q$$

Proof. We use induction over processes p to prove the lemma for all $p \in \mathbb{P}$.

Base case:
Take $p \approx \mathbf{0}$. For $q \approx \mathbf{0} \in \mathbb{B}$ it is clear that $p \approx q$ so also $\mathcal{E}_T \vdash p \approx q$.

Inductive case:

- $\mathbf{p \approx a.p'}$
  For $p \approx a.p'$ with arbitrary action a and process p', the IH is

  $$\exists q' \in \mathbb{B}, \mathcal{E}_T \vdash p' \overset{\text{IH}}{\approx} q'$$

  Take $q \approx a.q' \in \mathbb{B}$. Now it follows that $p \approx a.p' \overset{\text{IH}}{\approx} a.q' \approx q$
- $\mathbf{p \approx p_1 + p_2}$
  For $p \approx p_1 + p_2$ with arbitrary processes p_1, p_2, the IHs are

  $$\exists q_1 \in \mathbb{B}, \mathcal{E}_T \vdash p_1 \overset{\text{IH}}{\approx} q_1$$

  $$\exists q_2 \in \mathbb{B}, \mathcal{E}_T \vdash p_2 \overset{\text{IH}}{\approx} q_2$$

  Take $q_1 + q_2 \approx q \in \mathbb{B}$ with $q_1 \overset{\text{IH}}{\approx} p_1$ and $q_2 \overset{\text{IH}}{\approx} p_2$. Now it follows that $p \approx p_1 + p_2 \overset{\text{IH}}{\approx} q_1 + q_2 \approx q$

– $\mathbf{p \approx p_1 \| p_2}$
Take $p \approx p_1 \| p_2$ with arbitrary processes p_1, p_2, the IHs are

$$\exists q_1 \in \mathbb{B}, \mathcal{E}_T \vdash p_1 \overset{\text{IH}}{\approx} q_1$$

$$\exists q_2 \in \mathbb{B}, \mathcal{E}_T \vdash p_2 \overset{\text{IH}}{\approx} q_2$$

Take these $p_1 \overset{\text{IH}}{\approx} q_1 \in \mathbb{B}$ and $p_2 \overset{\text{IH}}{\approx} q_2 \in \mathbb{B}$ from the IHs. From lemma 12 it follows that there exists $q \in \mathbb{B}$ such that $q_1 \| q_2 \approx q$. It follows that $p \approx p_1 \| p_2 \overset{\text{IH}}{\approx} q_1 \| q_2 \approx q$.

□

3.2.2 Ground-Completeness

Now we know that $\text{BCCSP}_\|$ can be reduced to BCCSP. We combine this with the properties that BCCSP is ground-complete and $\text{BCCSP}_\|$ is sound to prove the ground-completeness modulo trace equivalence of $\text{BCCSP}_\|$.

Theorem 3. *The set of axioms $\mathcal{E}_T$ is ground-complete modulo trace equivalence:*

$$\forall p, q \in \mathbb{P}, p \sim_T q \implies \mathcal{E}_T \vdash p \approx q$$

Proof. Take arbitrary p and q such that $\mathtt{T}(p) = \mathtt{T}(q)$. From lemma 13 it follows that there exists $p', q' \in \mathbb{B}$ such that $\mathcal{E}_T \vdash p \approx p'$ and $\mathcal{E}_T \vdash q \approx q'$. Using the soundness of $\mathcal{E}_T$ we now see that $\mathtt{T}(p) = \mathtt{T}(p')$ and $\mathtt{T}(q) = \mathtt{T}(q')$ (theorem 2). From the original assumption therefore also $\mathtt{T}(p') = \mathtt{T}(q')$ holds. Using the property that $\mathcal{E}_\mathbb{B}$ is ground-complete modulo trace equivalency (theorem 1), it follows that $\mathcal{E}_\mathbb{B} \vdash p' \approx q'$. Since $\mathcal{E}_\mathbb{B} \subseteq \mathcal{E}_T$, it follows that $\mathcal{E}_T \vdash p' \approx q'$ as we can simply ignore the extra axioms. Therefore $\mathcal{E}_T \vdash p \approx p' \approx q' \approx q$ □

3.3 Ground-Completeness of $\text{BCCSP}_\|$ modulo Completed Trace Semantics

Next we look at a different axiomatisation. Specifically we prove the ground-completeness modulo *completed trace semantics* of $\text{BCCSP}_\|$. A sequence w is a *completed trace* of process p_0 if there exists the sequence of transitions $p_0 \xrightarrow{w} p_k$, *but no transitions from p_k exist.* The function $\mathtt{CT} : \mathbb{P} \to \mathcal{P}(\mathcal{A}^*)$ maps a process to its set of completed traces. In completed trace semantics, two processes p, q are equivalent if their sets of completed traces are the same, denoted by $p \sim_{CT} q$. The following two example processes $p = a.b.\mathbf{0}$ and $q = a.b.\mathbf{0} + a.\mathbf{0}$ demonstrate why traces and completed traces differ and

why we can not use the same axiom system as before. Specifically, the sets of traces of both processes are the same:

$$\mathtt{T}(a.b.\mathbf{0}) = \{\epsilon, a, ab\} = \mathtt{T}(a.b.\mathbf{0} + a.\mathbf{0})$$

While their sets of completed traces are different:

$$\mathtt{CT}(a.b.\mathbf{0}) = \{ab\} \neq \{a, ab\} = \mathtt{CT}(a.b.\mathbf{0} + a.\mathbf{0})$$

Clearly the axiomatisation of $\text{BCCSP}_{\|}$ modulo completed trace equivalence must also be different to the axiomatisation of $\text{BCCSP}_{\|}$ modulo trace equivalence. The following table 3.3 displays the new axiomatisation E_{CT}. We show this axiomatisation to be ground-complete for $\text{BCCSP}_{\|}$ modulo completed trace semantics. It uses mostly the same axioms, but the axioms (TD) and (TP) from $\mathcal{E}_T$ are replaced with (CTD) and (CTP).

$(A0)$	$x + \mathbf{0} \approx x$	$(A1)$	$x + y \approx y + x$
$(A2)$	$(x + y) + z \approx x + (y + z)$	$(A3)$	$x + x \approx x$
$(P0)$	$x\|\mathbf{0} \approx x$	$(P1)$	$x\|y \approx y\|x$
(CTD)	$a.(b.x + z) + a.(c.y + w) \approx a.(b.x + c.y + z + w)$		
(CTP)	$(a.x + b.y + w)\|z \approx (a.x + w)\|z + (b.y + w)\|z$		
$(EL1)$	$a.x\|b.y \approx a.(x\|b.y) + b.(a.x\|y)$		

Table 3.3: the set of axioms $\mathcal{E}_{CT}$

For the proof of completeness we once again reduce $\text{BCCSP}_{\|}$ terms to BCCSP terms, as ground-completeness for $\text{BCCSP}_{\|}$ modulo completed trace semantics has also been proven in [8] for the following axiom system $\mathcal{E}_{CT,\mathbb{B}}$:

$$\mathcal{E}_{CT,\mathbb{B}} = \{(A0), (A1), (A2), (A3), (P0), (P1), (CTD)\} \tag{3.8}$$

Theorem 4. *The set of axioms $\mathcal{E}_{CT,\mathbb{B}}$ is ground-complete modulo trace equivalence:*

$$\forall p, q \in \mathbb{B}, p \sim_T q \implies \mathcal{E}_{CT,\mathbb{B}} \vdash p \approx q$$

3.3.1 The Normal Form of BCCSP Terms

This time our proof that $\text{BCCSP}_{\|}$ terms can be reduced to BCCSP terms uses a slightly different strategy. Instead of proving it directly with nested proofs by induction, we first prove that each BCCSP term can be rewritten to a useful normal form. The set of all BCCSP terms that are in normal form is generated by the following rules:

1. the term $\mathbf{0}$ is an element of the set

2. if for all $i \in I$, x_i is an element of the set and $a_i \in \mathcal{A}$, then the term

$$\sum_{i \in I} a_i.x_i$$

is also an element of the set.

We denote this set as $\mathbb{B}_{norm}$. Since this set is defined recursively we can use it for a proof by induction. First we prove that each BCCSP term can indeed be rewritten to a term in normal form

Lemma 14. *Every BCCSP term p is equivalent to a term in normal form:*

$$\forall p \in \mathbb{B}, \exists q \in \mathbb{B}_{norm}, \mathcal{E}_{CT,\mathbb{B}} \vdash p \approx q$$

Proof. We use structural induction on BCCSP terms p, to show that the property holds for all p.

Base Case:
Take $p \approx \mathbf{0}$. The normal form for p is p itself, as $\mathbf{0}$ is a term in normal form.

Inductive Cases:

- $\mathbf{p = a.x}$:
  Take $p = a.x$ where a is an arbitrary action and x an arbitrary BCCSP term in normal form. According to the induction hypothesis, x can be rewritten to a term $\hat{x} \in \mathbb{B}_{norm}$. In this case $p = a.x \approx a.\hat{x}$ is in normal form.
- $\mathbf{p = x + y}$:
  Take $p = x + y$ with x, y arbitrary BCCSP terms. Take $\hat{x}, \hat{y}$ which are normal forms of x and y respectively according to the IH. If both $\hat{x} = \mathbf{0}$ and $\hat{y} = \mathbf{0}$ then the normal form of p is $\mathbf{0}$:

  $$p = x + y \approx x_n + y_n \approx \mathbf{0} + \mathbf{0} \approx \mathbf{0}$$

  If either $\hat{x} = \mathbf{0}$ or $\hat{y} = \mathbf{0}$ but not both, then the normal form of p can be constructed like this:

  $$p = x + y \approx \mathbf{0} + y \approx y \approx \hat{y} \quad \text{or} \quad p = x + y \approx x + \mathbf{0} \approx x \approx \hat{x}$$

  Otherwise, $\hat{x}$ must be of the form $\sum_{i \in I} a_i.x_i$ and $\hat{y}$ must be of the form $\sum_{j \in J} a_j.x_j$. In this case, we can construct the normal form of p like this:

  $$\begin{aligned} p = x + y &\approx \hat{x} + \hat{y} \\ &\approx \sum_{i \in I} a_i.x_i + \sum_{j \in J} a_j.x_j \\ &\approx \sum_{i \in I \cup J} a_i.x_i \end{aligned}$$

□

3.3.2 Reduction to BCCSP

The next lemma makes a process by placing two BCCSP terms in normal form around a parallel composition operator. We show that we can 'break apart' the partial terms of the left-hand term: The process is equivalent to a sum of those partial terms all individually parallel to the original right-hand term.

Lemma 15. *For all BCCSP terms q that are in normal form and $p = \sum_{i \in I} a_i.x_i$ with arbitrary $a_i \in \mathcal{A}$, $x_i \in \mathbb{B}_{norm}$ and $|I| > 0$, the following property holds:*

$$\mathcal{E}_{CT} \vdash p \| q \approx \sum_{i \in I} (a_i.x_i \| q)$$

Proof. We use induction on the size of I, to prove the property for all sizes of I larger than 0.

Base case:
Take $|I| = 1$. This means that $p = a.x$ for some $a \in \mathcal{A}$ and $x \in \mathbb{B}_{norm}$. Immediately it follows that $p \| q \approx a.x \| q$ is of the form we desire.

Inductive case:
Take $|I| = n > 1$. Our induction hypothesis is that for all sets of indices with $|J| = n - 1$,

$$\mathcal{E}_{CT} \vdash \sum_{j \in J} a_j.x_j \| q \overset{\text{IH}}{\approx} \sum_{j \in J} (a_j.x_j \| q)$$

We call the first two elements of I 1 and 2. Take $K = I \setminus \{1, 2\}$. We can now rewrite p to the following:

$$p = a_1.x_1 + a_2.x_2 + \sum_{k \in K} a_k.x_k$$

with arbitrary $a_1, a_2 \in \mathcal{A}$ and arbitrary $x_1, x_2 \in \mathbb{B}_{norm}$. Note that this is possible because we have at least two terms $a_i.x_i$ and if there are exactly two terms we can use $(A0)$ to say that the K term is $\mathbf{0}$. Now

we see that

$$
\begin{aligned}
p\|q &\approx \left(\sum_{i\in I} a_i.x_i\right)\|q \\
&\approx \Big(a_1.x_1 + a_2.x_2 + \sum_{k\in K} a_k.x_k\Big)\|q \\
&\approx \left(\Big(a_1.x_1 + \sum_{k\in K} a_k.x_k\Big)\|q\right) + \left(\Big(a_2.x_2 + \sum_{k\in K} a_k.x_k\Big)\|q\right) && (CTP) \\
&\approx \left(\sum_{j\in(\{1\}\cup K)} (a_j.x_j)\ \|q\right) + \left(\sum_{j\in(\{2\}\cup K)} (a_j.x_j)\ \|q\right) \\
&\overset{\text{IH}}{\approx} \left(\sum_{j\in(\{1\}\cup K)} (a_j.x_j\|q)\right) + \left(\sum_{j\in(\{2\}\cup K)} (a_j.x_j\ \|q)\right) && (*) \\
&\approx \sum_{i\in(\{1,2\}\cup K)} (a_i.x_i\|q) && (A1),\ (A2),\ (A3)
\end{aligned}
$$

which is the form we desire.
$(*)$: $|\{1\} \cup K| = n - 1$ and $|\{2\} \cup K| = n - 1$, so we can use the induction hypotheses.

□

Now we can use these lemmas to prove a lemma similar to lemma 12, only now we apply it to BCCSP terms that are in normal form.

Lemma 16. *For all terms $p, q \in \mathbb{B}_{norm}$, there exists another term $r \in \mathbb{B}_{norm}$ such that the following property holds:*

$$\mathcal{E}_{CT} \vdash p\|q \approx r$$

Proof. We use induction on p to prove the property for all terms p. We do this induction using the structure of BCCSP terms that are in normal form.

Base Case:
Take $p = \mathbf{0}$. We can now choose $r = q$, as

$$p\|q \approx \mathbf{0}\|q \approx q$$

Inductive Case: Take $p = \sum_{i\in I} a_i.x_i$ with $|I| > 0$. The induction hypotheses become for all $i \in I$ that

$$\forall q_i \in \mathbb{B}_{norm}, \exists r_i \in \mathbb{B}_{norm}, x_i\|q_i \approx r_i \tag{3.9}$$

Using lemma 15 we see that

$$p\|q \approx \Big(\sum_{i\in I} a_i.x_i\Big)\|q \approx \sum_{i\in I}(a_i.x_i\|q) \tag{3.10}$$

We now prove that for each $i \in I$, the following property holds:

$$\exists r_i' \in (\mathbb{B}_{norm} \setminus \{\mathbf{0}\}), r_i' \approx a_i.x_i\|q \tag{3.11}$$

We do this using induction on q to prove the property for all terms q. We once again use the structure of BCCSP terms that are in normal form for this induction.

Base Case:
Take $q = \mathbf{0}$. We can now construct r_i' like this:

$$\begin{aligned} a_i.x_i\|q &\approx a_i.x_i\|\mathbf{0} \\ &\approx a_i.x_i \\ &\approx r_i' \in (\mathbb{B}_{norm} \setminus \{\mathbf{0}\}) \end{aligned} \tag{$P0$}$$

Inductive Case:
Take $q = \sum_{j\in J} b_j.y_j$ with arbitrary $b_j \in \mathcal{A}$, $y_j \in \mathbb{B}_{norm}$ and $|J| > 0$. The induction hypotheses become for all y_j:

$$\exists r_{i,j} \in (\mathbb{B}_{norm} \setminus \{\mathbf{0}\}), r_{i,j} \approx a_i.x_i\|y_j \tag{3.12}$$

We now take these $r_{i,j,1} \approx a_i.x_i\|y_j$ for each y_j from (3.12) and also take $r_{i,j,2} \approx x_i\|b_j.y_j$ for each y_j from (3.9). Then we can construct r_i' like this:

$$\begin{aligned} a_i.x_i\|q &\approx a_i.x_i\|\Big(\sum_{j\in J} b_j.y_j\Big) \\ &\approx \Big(\sum_{j\in J} b_j.y_j\Big)\|a_i.x_i \\ &\approx \sum_{j\in J}(b_j.y_j\|a_i.x_i) && \text{(lemma 15)} \\ &\approx \sum_{j\in J} b_j.(y_j\|a_i.x_i) + a_i(b_j.y_j\|x_i) && \text{(EL1)} \\ &\approx \sum_{j\in J} b_j.(a_i.x_i\|y_j) + a_i(x_i\|b_j.y_j) && \text{(P1)} \\ &\overset{\text{IH}}{\approx} \sum_{j\in J} b_j.r_{i,j,1} + a_i.r_{i,j,2} \\ &= r_i' \in (\mathbb{B}_{norm} \setminus \{\mathbf{0}\}) \end{aligned}$$

So now we have shown for each $a_i.x_i \| q$ that there exists an r'_i that is in normal form and not $\mathbf{0}$. If we sum these r'_i, then it follows from associativity that the result will be in normal form as well. We can therefore combine this with (3.10) to construct r:

$$\begin{aligned} p \| q &\approx \sum_{i \in I} (a_i.x_i \| q) && (3.10) \\ &\approx \sum_{i \in I} r'_i && (3.11) \\ &= r \in \mathbb{B}_{norm} \end{aligned}$$

□

We have shown that we can construct a BCCSP term from combining two BCCSP terms with an interleaving parallel operator. However, all the BCCSP terms have to be in normal form. Luckily, we have already shown that every BCCSP term is equationally similar to one in normal form in lemma 14. Using these two we can now easily show that every $\text{BCCSP}_\|$ term can also be reduced to a BCCSP term in normal form. The proof of this is so similar to the one in lemma 13, that it is mostly left out.

Lemma 17. *For every $\text{BCCSP}_\|$ term p there exists BCCSP term q in normal form for which the following holds:*

$$\mathcal{E}_{CT} \vdash p \approx q$$

Proof. The cases where $p = \mathbf{0}$, $p = a.p'$ and $p = p_1 + p_2$ are analogous to the same cases of the proof for 13.

$\mathbf{p = p_1 \| p_2}$:
Take $p \approx p_1 \| p_2$ with arbitrary processes p_1, p_2. The induction hypotheses are

$$\exists q_1 \in \mathbb{B}, \mathcal{E}_{CT} \vdash p_1 \overset{\text{IH}}{\approx} q_1$$
$$\exists q_2 \in \mathbb{B}, \mathcal{E}_{CT} \vdash p_2 \overset{\text{IH}}{\approx} q_2$$

Take these $p_1 \overset{\text{IH}}{\approx} q_1 \in \mathbb{B}$ and $p_2 \overset{\text{IH}}{\approx} q_2 \in \mathbb{B}$ from the IHs. From lemma 14 it follows that there exists $q_1 \approx q'_1 \in \mathbb{B}_{norm}$ and $q_2 \approx q'_2 \in \mathbb{B}_{norm}$. From lemma 16 it follows that there exists $q \in \mathbb{B}$ such that $q'_1 \| q'_2 \approx q$. It follows that $p \approx p_1 \| p_2 \overset{\text{IH}}{\approx} q_1 \| q_2 \approx q'_1 \| q'_2 \approx q$.

□

Now we can use this reduction very similarly to the proof of theorem 3 to prove ground-completeness for $\text{BCCSP}_\|$ modulo closed trace equivalence. For this proof we assume that the axiomatisation is sound modulo completed trace semantics.

Theorem 5. *The set of axioms $\mathcal{E}_{CT}$ is ground-complete modulo trace equivalence:*

$$\forall p, q \in \mathbb{P}, p \sim_T q \implies \mathcal{E}_{CT} \vdash p \approx q$$

Proof. This proof of this theorem is analogous to the proof of theorem 3, but uses theorem 4 instead of theorem 1, lemma 17 instead of lemma 13, soundness modulo completed trace semantics instead of soundness modulo trace semantics and $\mathcal{E}_{CT}$ instead of $\mathcal{E}_T$ □

Chapter 4

Related Work

Until very recently, the main attempts at describing the axiomatisability of the parallel composition operator were done modulo bisimulation [1]. Then [1] took a thorough look at the language BCCSP. It showed for the whole spectrum of semantical equivalence whether or not the language $\text{BCCSP}_{\|}$ had a finite, ground complete axiomitisation. Our thesis borrows a lot of ideas from that paper. For example, the paper suggests to reduces $\text{BCCSP}_{\|}$ terms to BCCSP for the proof of ground-completeness for trace semantics. Our thesis does a more thorough exploration of the proof for soundness. Traditionally, soundness is mostly left out of a proof for an axiomatisation [8]. The approach [1] takes for proving correctness modulo both trace semantics as completed trace semantics is similar to the approach in this paper. It reduces to BCCSP terms as well, but the reducability follows from an earlier result that another axiom system was reducable and the fact that these other axioms were specific versions of that axiom system.

Besides this recent paper, axiomatisation of algebras with a parallel composition operator modulo *bisimulation* semantics have been explored before[7], but no systematic exploration modulo the rest of the spectrum of semantics had been done [1].

Chapter 5

Conclusions

First we explored the basics of process algebra using some examples of CCS. Then we more thoroughly explored a part of process algebra, by expanding on the axiomatisability of BCCSP modulo trace semantics. For soundness, we showed that the function that maps a BCCSP term to its set of traces is actually a homomorphism. For ground-completeness, we showed that $\text{BCCSP}_{\|}$ terms can indeed be reduced to BCCSP terms using the axioms we provided.

5.1 Future Work

It may be interesting to investigate whether the `CT` function is a homomorphism as well, or if another approach to proving soundness has to be taken for completed trace semantics.

The algebra $\text{BCCSP}_{\|}$ is very similar to CCS, as it has most terms of CCS, but leaves out the 'renaming' and 'restriction' operators. Perhaps the axiomatisations for trace and completed trace semantics could be extended to work for CCS as well, which might especially be interesting given the popularity of CCS. Some challenges in doing this might be that the semantics of both parallel composition and non-deterministic choice is slightly different or the fact that recursive processes are often allowed in CCS. Both of these differences might lead to very different axiomatisations if they exist at all.

Bibliography

[1] Luca Aceto, Valentina Castiglioni, Anna Ingólfsdóttir, Bas Luttik, and Mathias Ruggaard Pedersen. On the axiomatisability of parallel composition: A journey in the spectrum. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*, volume 171 of *LIPIcs*, pages 18:1–18:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[2] J.C.M. Baeten, editor. *Applications of Process Algebra (2nd ed)*. Cambridge University Press, United Kingdom, 2005.

[3] Jos C. M. Baeten. A brief history of process algebra. *Theor. Comput. Sci.*, 335(2-3):131–146, 2005.

[4] Ansgar Fehnker, Rob van Glabbeek, Peter Höfner, Annabelle McIver, Marius Portmann, and Wee Lum Tan. A Process Algebra for Wireless Mesh Networks. In Helmut Seidl, editor, *Programming Languages and Systems*, pages 295–315, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[5] Robin Milner. *Communication and concurrency*. PHI Series in computer science. Prentice Hall, 1989.

[6] F. Moller and P. Stevens. Edinburgh concurrency workbench user manual(version 7.1). `http://www.dcs.ed.ac.uk/home/cwb/`. Accessed: 2021-05-24.

[7] David Michael Ritchie Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *Theoretical Computer Science, 5th GI-Conference, Karlsruhe, Germany, March 23-25, 1981, Proceedings*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[8] Rob J. van Glabbeek. The linear time - branching time spectrum I. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra*, pages 3–99. North-Holland / Elsevier, 2001.

[9] Nicolas Wua. Applications of process algebra in systems biology. *Life Sciences Interface Doctoral Training Centre Student Projects, Brasenose College, University of Oxford, Oxford, UK*, 2006.