Properties of the Timed Operational and Denotational Semantics of Orc

Ian Wehrman, David Kitchin, William R. Cook, Jayadev Misra
Department of Computer Sciences
The University of Texas at Austin
email: {iwehrman,dkitchin,wcook,misra}@cs.utexas.edu

December 14, 2007
Orc is a language for structured concurrent programming. Orc provides three powerful combinators that define the structure of a concurrent computation. These combinators support sequential and concurrent execution, and concurrent execution with blocking and termination.

Orc is particularly well-suited for task orchestration, a form of concurrent programming with applications in workflow, business process management, and web service orchestration. Orc provides constructs to orchestrate the concurrent invocation of services while managing time-outs, priorities, and failures of services or communication.

Previous work on the semantics of Orc has focused on its asynchronous behavior. In this report, we define a relative-time operational semantics of Orc that allows reasoning about delays, which are introduced explicitly by time-based constructs or implicitly by network delays. We develop a number of identities among Orc expressions and define an equality relation that is a congruence. We also develop a denotational semantics for Orc, in which the meaning of an Orc expression is a set of traces. A number of properties about the semantics are shown here, including equivalence of the operational and denotational semantics.
# Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td></td>
<td>ii</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Timed Operational Semantics</td>
<td>2</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Time-shifted Expressions</td>
<td>2</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Transition Rules</td>
<td>3</td>
</tr>
<tr>
<td>1.2.3</td>
<td>Executions and Traces</td>
<td>4</td>
</tr>
<tr>
<td>1.2.4</td>
<td>Substitution Events</td>
<td>5</td>
</tr>
<tr>
<td>1.2.5</td>
<td>Summary of Notation</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>Basic Operators on Sequences</td>
<td>5</td>
</tr>
<tr>
<td>1.3.1</td>
<td>Definitions of Operators</td>
<td>6</td>
</tr>
<tr>
<td>1.3.2</td>
<td>Coercion</td>
<td>7</td>
</tr>
<tr>
<td>1.3.3</td>
<td>Some Simple Facts</td>
<td>7</td>
</tr>
<tr>
<td>1.4</td>
<td>A Specific Set of Sequences</td>
<td>9</td>
</tr>
<tr>
<td>1.5</td>
<td>Basic Theorems on Executions</td>
<td>9</td>
</tr>
<tr>
<td>1.5.1</td>
<td>Evolution</td>
<td>10</td>
</tr>
<tr>
<td>1.5.2</td>
<td>Substitution Independence</td>
<td>15</td>
</tr>
<tr>
<td>1.6</td>
<td>Identities</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>Combinators applied to Executions</td>
<td>32</td>
</tr>
<tr>
<td>2.1</td>
<td>Base Expressions</td>
<td>32</td>
</tr>
<tr>
<td>2.2</td>
<td>Meanings of Execution Combinators</td>
<td>35</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Meaning of Symmetric Composition</td>
<td>35</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Meaning of Sequential Composition</td>
<td>36</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Meaning of Asymmetric Composition</td>
<td>36</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Monotonicity</td>
<td>37</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Distributivity</td>
<td>39</td>
</tr>
<tr>
<td>2.3</td>
<td>Characterization of Symmetric Composition</td>
<td>42</td>
</tr>
<tr>
<td>2.3.1</td>
<td>$\langle f \parallel g \rangle \subseteq \langle f \parallel g \rangle$</td>
<td>42</td>
</tr>
<tr>
<td>2.3.2</td>
<td>$\langle f \parallel g \rangle \subseteq \langle f \parallel g \rangle$</td>
<td>43</td>
</tr>
<tr>
<td>2.4</td>
<td>Characterization of Sequential Composition</td>
<td>44</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Preliminary Results</td>
<td>44</td>
</tr>
<tr>
<td>2.4.2</td>
<td>$\langle f &gt;x&gt; g \rangle \subseteq \langle f &gt;x&gt; g \rangle$</td>
<td>46</td>
</tr>
</tbody>
</table>
## CONTENTS

2.4.3 \([f] > x > [g] \subseteq [f > x > g]\) ........................................ 48

2.5 Characterization of Asymmetric Composition ................................. 50

2.5.1 Preliminary Results ......................................................... 50

2.5.2 \([f < x < g] \subseteq [f] < x < [g]\) ....................................... 56

2.5.3 \([f] < x < [g] \subseteq [f < x < g]\) ....................................... 62

3 Breadth and Trace Preservation .............................................. 66

3.1 Additional Operators on Sequences ........................................ 67

3.1.1 Breadth ................................................................. 67

3.1.2 Visible sequences and Traces ............................................ 71

3.1.3 Base Expressions are Broad ............................................ 73

3.2 Symmetric Composition ...................................................... 76

3.2.1 Preliminary Results ....................................................... 76

3.2.2 Symmetric Composition Preserves Breadth .......................... 77

3.2.3 Symmetric Composition Preserves Traces ............................ 80

3.3 Sequential Composition .................................................... 84

3.3.1 Preliminary Results ....................................................... 84

3.3.2 Sequential Composition Preserves Breadth .......................... 85

3.3.3 Sequential Composition Preserves Traces ............................ 89

3.4 Asymmetric Composition .................................................. 91

3.4.1 Preliminary Results on Constrained Partial Merge ............... 91

3.4.2 Preliminary Results on Constrained Full Merge .................. 100

3.4.3 Asymmetric Composition Preserves Breadth ....................... 103

3.4.4 Asymmetric Composition Preserves Traces .......................... 110

4 Traces are Denotations ....................................................... 113

4.1 The Denotation of an Orc Expression .................................... 114
Chapter 1

Introduction

1.1 Introduction

This monograph establishes a number of semantic properties of Orc expressions. An operational semantics of Orc is given elsewhere and abbreviated in Section 1.2. Executions and traces of Orc expressions are defined (see Section 1.2.3) based on this operational semantics.

In Section 2.2, page 35, we define combinators, corresponding to each Orc combinator, that can be applied to sets of executions (and traces). Then we can write \( U \mid V \), \( U >_x V \) and \( U <_x V \), where \( U \) and \( V \) are sets of executions. Denoting the set of executions of \( f \) by \([ f ]\), we establish in Chapter 2 that 
\[ [ f * g ] = [ f ] * [ g ] \]
where * is any orc combinator, \( \mid, >_x \) or \( <_x \).

The results of Chapter 2 are used in Chapter 3 to establish that the traces of \( f * g \) can be determined from the traces of \( f \) and \( g \). Denoting the traces of \( f \) by \( \{ f \} \), we write \( f \cong g \) to mean \( \{ f \} = \{ g \} \). We show that relation \( \cong \) is an equality relation; given \( f \cong g \), \( f \) and \( g \) can replace each other in all contexts. We establish even finer results by defining a partial order over expressions; \( f \subseteq g \) means that \( \{ f \} \subseteq \{ g \} \). Given \( f \subseteq g \), we prove that \( f * h \subseteq g * h \) and \( h * f \subseteq h * g \); i.e., replacing \( f \) by \( g \) in any context results in at least the same set of traces.

Finally, in Chapter 4 we present a meaning function \( \mu \) for arbitrary Orc expressions, including those with recursively defined expressions. The meaning \( \mu(f) \) of expression \( f \) is a set of traces. The final theorem establishes that \( \{ f \} = \mu(f) \); i.e., that the operational and denotational semantics are equivalent.

The main theorems we establish are:

**Theorem 1** \( \{ f * g \} = \{ f \} * \{ g \} \)

See Theorem 26, page 113.

**Theorem 2** Suppose \( U \subseteq U' \) and \( V \subseteq V' \). Then, \( U * V \subseteq U' * V' \).

See Theorem 9, page 37.

**Theorem 3** For any \( U \) and \( V \),
1. (Left Distributivity) \((\bigcup_i P_i \ast V) = (\bigcup_i R_i) \ast V\), for a family of sets \(P_i\).

2. (Right Distributivity) \((\bigcup_i U \ast Q_i) = U \ast (\bigcup_i Q_i)\), for a sequence of sets \(Q_i\), where \(Q_0 \subseteq Q_1 \subseteq \cdots\).

See Theorem 10, page 40.

**Theorem 4** \(\langle f \rangle = \mu(f)\) See Theorem 27, page 119.

### 1.2 Timed Operational Semantics

#### 1.2.1 Time-shifted Expressions

A time-shifted expression, written \(f^t\), is the expression that results from \(f\) after \(t\) units have elapsed without occurrence of an event. When it is not possible for \(t\) time units to elapse without \(f\) engaging in an event we write \(f^t = \bot\), where \(\bot\) is an unreachable expression described later. The time-shifted expression \(f^t\), for \(t \geq 0\), is defined in Figure 1.1 based on the structure of \(f\).

\[
\begin{align*}
(f \mid g)^t & \triangleq f^t \mid g^t \\
(f >x> g)^t & \triangleq f^t >x> g^t \\
(f <x< g)^t & \triangleq f^t <x< g^t \\
M(x)^t & \triangleq M(x) \\
M(m)^t & \triangleq \begin{cases} M(m) & \text{if } t = 0 \\ \bot & \text{otherwise.} \end{cases} \\
?k^t & \triangleq \{ (s,m) \mid (t+s,m) \in k \} \cup \{ x \in k \mid x = \omega \}
\end{align*}
\]

**Figure 1.1: Definition of Time-shifted Expressions**

The first three cases, for each of the combinators, are easy to justify informally. Expression \(M(x)^t\), where \(x\) is a variable, is simply \(M(x)\) because the site cannot be invoked until the parameter has a value. Expression \(M(m)^t\), where \(m\) is a value, must be invoked at time 0; therefore, \(M(m)^0 = M(m)\), whereas \(M(m)^t = \bot\) for \(t > 0\). The time-shifted handle \(?k^t\) may publish \(m\) at time \(s\) iff \(?k\) may publish \(m\) at \(t+s\); and \(?k^t\) includes \(\omega\) iff \(?k\) does.

The definitions for \(M(x)^t\) and \(M(m)^t\) in Figure 1.1 also encompass local sites \(\text{if(true)}^t\), \(\text{Signal}^t\), \(\text{let(m)}^t\), etc. Of particular importance is \(\text{Rtimer}\). Consider the handle \(?k\) that results from a call to \(\text{Rtimer}(3)\). It is easily seen that \(?k^2 = ?j\), where \(?j\) is a handle resulting from a call to \(\text{Rtimer}(1)\), i.e., \(\text{Rtimer}(3)\) behaves like \(\text{Rtimer}(1)\) after 2 times units have elapsed.
1.2.2 Transition Rules

We present a timed operational semantics of Orc based on a labeled transition system. The labels of the transition system are time-event pairs \((t, a)\). The relation \(f \xrightarrow{t,a} f'\), defined in Figure 1.2, states that expression \(f\) may transition exactly \(t\) time units after its evaluation starts with event \(a\) to expression \(f'\).

Events are either publication events, written \(!m\), or internal events, written \(\tau\). Publication events correspond to the communication of value \(m\) to the environment during a transition. Internal events correspond to state changes not intended to be observable by the environment. We refer to both publication and internal events as base events.

The times in the transition relation are relative to the start of evaluation of the expression. Furthermore, \(f \xrightarrow{t,a} f'\) specifies that no other events have occurred in the \(t\) units that have passed since the beginning of the evaluation of \(f\). Times are natural numbers (though we can use any totally-ordered set with a least element, such as the non-negative reals).

Notation  Henceforth, expressions are denoted by \(f, g, h\); variables by \(x, y, z\); events by \(a, b\); and times by \(t, s\). Sets of objects are denoted by the uppercase versions of their corresponding letters. We write \([m/x].f\) for the expression obtained from \(f\) by replacing every free occurrence of \(x\) by value \(m\). Parameters, which are either variables or values, are denoted by \(p\).
1.2.3 Executions and Traces

In this section, we formalize the notions of executions and traces for expressions. An execution of $f$ is a sequence of timed events in which $f$ may engage. A trace is an execution with the $\tau$ events removed. We write $f \triangleq u \Rightarrow g$, where $u$ is a sequence of timed events of the form $(t,a)$, to denote that $f$ may engage in event $a$ exactly $t$ units after the start of its evaluation, and transition to $g$ immediately after engaging in all the events in $u$.

Execution relation $\Rightarrow$ is derived from the reflexive and transitive closure of the transition relation $\rightarrow$ of Figure 1.2. However, we need to shift the times in forming the transitive closure. Given $f \xrightarrow{(s,a)} f'$ and $f' \xrightarrow{(t,b)} f''$, we can not claim that $f \xrightarrow{(s,a),(t,b)} f''$, because $b$ occurs $s + t$ units after the evaluation of $f$ starts. We define $u_t$ as the sequence that results from increasing each time component of $u$ by $t$. The definition of $u_t$ is also lifted to sets pointwise: $U_t \triangleq \{ u_t \mid u \in U \}$.

Define relation $\Rightarrow$ as the reflexive-transitive closure of relation $\rightarrow$ except that the time components accumulate.

$$ f \Rightarrow f \, (\text{Ex-Refl}) \quad \frac{f \xrightarrow{(t,a)} f'', f'' \xrightarrow{u} f'}{f \xrightarrow{(t,a)u} f'} \, (\text{Ex-Trans}) $$

Call $u$ an execution of $f$ if $f \Rightarrow f'$ for some $f' \neq \bot$. Note that the empty sequence $\epsilon$ is an execution of any expression by rule (Ex-Refl).

The definition of executions requires $f' \neq \bot$ so that all intermediate expressions in an execution (such as $f''$) are reachable—if any intermediate expression is unreachable, the final expression, $f'$, would be unreachable because $\bot$ has no transitions.

A trace $\pi$ is obtained from execution $u$ by removing each internal event $(t, \tau)$. The definition is also lifted pointwise to sets: $U \triangleq \{ \pi \mid u \in U \}$.

**Notation** The execution set and trace set of $f$ are written $\llbracket f \rrbracket$ and $\langle f \rangle$ respectively:

$$ \llbracket f \rrbracket \triangleq \{ u \mid f \Rightarrow f', \text{for some } f' \}, \text{ and } \langle f \rangle \triangleq \llbracket f \rrbracket. $$
1.2.4 Substitution Events

Substitution Rules

\[
[m/y].(?k) = \begin{cases} 
?k & \text{if } p = y \\
M(p) & \text{otherwise}
\end{cases}
\]

\[
[m/y].(M(p)) = \begin{cases} 
E(m) & \text{if } p = y \\
E(p) & \text{otherwise}
\end{cases}
\]

\[
[m/y].(f | g) = ([m/y].f) | ([m/y].g)
\]

\[
[m/y].(f > x > g) = \begin{cases} 
([m/y].f) > x > g & \text{if } x = y \\
([m/y].f) > x > ([m/y].g) & \text{otherwise}
\end{cases}
\]

\[
[m/y].(f < x < g) = \begin{cases} 
f < x < ([m/y].g) & \text{if } x = y \\
([m/y].f) < x < ([m/y].g) & \text{otherwise}
\end{cases}
\]

We have the following rule with substitution event:

\[ f \overset{t}{\rightarrow}^{[m/x]} [m/x].(f^t) \quad \text{(SUBST)} \]

Henceforth, we write \([m/x].f^t\) to mean \([m/x].(f^t)\), i.e., the time-shift operator binds more strongly than substitution.

1.2.5 Summary of Notation

A summary of notation used in the sequel is shown in Figure 1.3.

- \(f \overset{t,a}{\rightarrow} g\): \(f\) evaluates in one step to \(g\) with event \(a\) at time \(t\)
- \(f \overset{u}{\Rightarrow} g\): \(f\) evaluates in multiple steps to \(g\) with execution \(u\)
- \(f^t\): expression \(f\) shifted forward in time by \(t\) units
- \(u_t, U_t\): execution or trace \(u\) (or set \(U\)) delayed by \(t\) units
- \(\pi, U\): trace of an execution \(u\) (or set \(U\))
- \([f]\): the set of executions of \(f\)
- \(\{f\}\): \([f]\), the set of traces of \(f\)
- \([m/x].f\): replace all free occurrences of \(x\) by \(m\) in \(f\)
- \(f \approx g\): \(\{f\} = \{g\}\)

Figure 1.3: Summary of Notation

1.3 Basic Operators on Sequences

A sequence is a finite sequence of tuples of the form \((t, b)\), where \(b\) is an event and \(t\) is its associated time, \(t \geq 0\). The times of events in a sequence are monotone non-decreasing. An event is either base or substitution event. There is a special base event, \(\tau\). As is customary, the empty sequence is denoted by \(\epsilon\).
**CHAPTER 1. INTRODUCTION**

**Notation**  For event $a$, $a.time$ is its associated time; for sequence $x$, $x.time$ is the time of its last event if $x$ is non-empty and 0 if $x$ is empty. Note that

- $\epsilon.time = 0$,
- $p_{t}.time = p.time + t$, for $p \neq \epsilon$,
- $(ap).time = a.time$, if $p = \epsilon$, and $p.time$, if $p \neq \epsilon$.

We define a few basic operations on sequences (and sets of sequences) in this section.

### 1.3.1 Definitions of Operators

Henceforth, $u$, $v$, $p$ and $q$ denote sequences. And, uppercase equivalents of these symbols denote sets of sequences. Symbols $a$ and $b$ denote events.

The time-shift of $u$ by $t$, where $t \geq 0$, is written as $u_t$; it is the same sequence as $u$ except that the associated time of each event is increased by $t$. Formally, time-shift for an individual event is given by $(s,b)_t = (s+b)$. And, for a sequence,

- $\epsilon_t = \epsilon$
- $(au)_t = a_t u_t$

It is customary to write a non-empty sequence $v$ as $au_t$, where $t$ is the time associated with event $a$. Here, $u_t$ is the suffix of $v$ containing all events except $a$, and $u$ is obtained from this suffix by decreasing associated times by $t$.

**Observation 1** For sequence $u$, $(u_s)_t = u_{s+t}$.

The prefix-closure of $u$, written as $u^*$, is the set of prefixes of $u$. Formally,

- $\epsilon^* = \{\epsilon\}$,
- $(au)^* = \{\epsilon\} \cup au^*$

Note that $\{\epsilon\} \subseteq u^*$, for all $u$. Therefore, $(au)^* = \{\epsilon\} \cup au^*$ holds (vacuously) even when $a = \epsilon$. Set $U$ is prefix-closed if $u^* \subseteq U$, for every $u$ in $U$.

The trace of $u$, $\overline{u}$, is the subsequence obtained from $u$ by dropping all $\tau$ events from it. Formally,

- $\tau = \epsilon$,
- $\overline{\tau} = \epsilon$,
- $\overline{a} = a$ where $a \neq \tau$,
- $\overline{ab} = \overline{a} \overline{b}$

**Event Removal from front of a sequence** Define $u \setminus a$, where $a$ is any event, as follows:

$$u \setminus a = \begin{cases} \{v\} & \text{if } u = av_t \text{ where } t = a.time \\ \phi & \text{otherwise} \end{cases}$$
CHAPTER 1. INTRODUCTION

1.3.2 Coercion

We coerce time-shift, trace and removal operators, \( \alpha \), to sets of sequences:

\[
\alpha(U) = (\cup u : u \in U : \alpha(u))
\]

Thus, for example,

\[
U \setminus a = (\cup u : u \in U : u \setminus a) = \{v \mid av_t \in U\}, \text{ where } t = a.time
\]

We use the convention that if \( \alpha(u) \) is a sequence (as in \( ut \) or \( \pi \)), it is treated as the set \( \{\alpha(u)\} \). If \( U = \phi \), the empty set, \( \alpha(U) = \phi \).

An operator \( \alpha \) is coercive if it satisfies \( \alpha(U) = (\cup u : u \in U : \alpha(u)) \). Operators time-shift, trace and removal are, by definition, coercive. A coercive operator distributes over set union.

Prefix-closure is coercive over non-empty sets. For empty set \( \phi \), \( \phi^* = \{\epsilon\} \), not \( \phi \). Prefix-closure distributes over set union even when some of the sets are empty.

Observation 2 A coercive operator \( \alpha \) satisfies the monotonicity condition:

\( U \subseteq V \Rightarrow \alpha(U) \subseteq \alpha(V) \)

Observation 3 Composition of coercive operators is coercive.

Observation 4 For coercive \( \alpha \) and \( \gamma \), where \( u \) ranges over all elements of \( U \),

\[
\begin{align*}
\alpha(u) \subseteq \gamma(u) & \Rightarrow \alpha(U) \subseteq \gamma(U), \\
\alpha(u) = \gamma(u) & \Rightarrow \alpha(U) = \gamma(U)
\end{align*}
\]

Idempotence Operator \( \alpha \) is idempotent if \( \alpha(\alpha(u)) = \alpha(u) \).

1.3.3 Some Simple Facts

Lemma 1 \( (ut)^* = (u)^*t \),

\[
\begin{align*}
(ut)^* &= (\pi)^*t, \\
u^* &= \pi^*.
\end{align*}
\]

Proof: Each of these may be proved by induction on the length of \( u \). We give a detailed proof for the last case, \( u^* = \pi^* \). If \( u = \epsilon \), the result follows easily. Next, let \( u = av \).

\[
\begin{align*}
\frac{u^*}{(av)^*} &= \{u = av\} \\
&= \{\text{definition of }^*\} \\
&= \{\epsilon\} \cup av^* \\
&= \{\text{distribute trace over union and concatenation}\}
\end{align*}
\]
A number of results over sequences can be coerced to sets of sequences using these observations. For example, we can derive \((U_t)^* = (U^*)_t\), as follows. From Lemma 1, page 7, \((u_t)^* = (u^*)_t\). The operator on each side of the identity is coercive, since it is a composition of two coercive operators (see Observation 3, page 7). Applying Observation 4, page 7, the result follows.

Henceforth, we will state results mostly over sequences, and derive the corresponding results over sets using coercion.

**Observation 5** Operators prefix-closure and trace are idempotent, i.e.,

\[(u^*)^* = u^*,\]
\[\overline{\pi} = \pi\]

Note that time-shift is not idempotent. Also note that for coercive and idempotent \(\alpha\), \(\alpha(\alpha(U)) = \alpha(U)\), by applying Observation 4, page 7 to the definition of idempotence.

**Observation 6** Let \(f \xrightarrow{a} f'\) where \(a\) is a substitution at time \(t\). Then, \([f'] = [f] \setminus a\), and \(a[f']_t \subseteq [f]\).

Proof: Given \(f \xrightarrow{a} f'\), for any \(u \in [f']\), i.e., \(f' \xrightarrow{u} a u_t \in [f]\). Therefore, \(a[f']_t \subseteq [f]\). We show \([f'] = [f] \setminus a\) by mutual inclusion.

- \([f'] \subseteq [f] \setminus a:\)
  \[
  u \in [f'] \subseteq [f] \setminus a \\
  \subseteq \{a[f']_t \subseteq [f]\} \\
  \Rightarrow \{[f] \setminus a = \{v \mid av_t \in [f]\}\}
  \]

- \([f] \setminus a \subseteq [f']:\)
  \[
  u \in [f] \setminus a \\
  \Rightarrow \{[f] \setminus a = \{v \mid av_t \in [f]\}\} \\
  \Rightarrow \{\text{meaning of execution}\}\]
\[ f \overset{a}{\rightarrow} f'' \Rightarrow u, \text{ for some } f'' \]
\[ \Rightarrow \{ \text{since } a \text{ is a substitution and } f \overset{a}{\rightarrow} f', \text{ we get } f' = f'' \} \]
\[ f \overset{a}{\rightarrow} f' \overset{u}{\Rightarrow} \Rightarrow \{ \text{obviously} \} \]
\[ u \in [f'] \]

1.4 A Specific Set of Sequences

We define set \( A(t) \), for any \( t, t \geq 0 \), to contain sequences of substitutions, as follows. For \( t \geq 0 \),
\[
A(t) = \{ u_r | u_r \text{ is a finite sequence of substitutions at time } r, 0 \leq r \leq t \}.
\]
Similarly, we define set \( D(t) \), for any \( t \geq 0 \) to contain sequences of substitutions as follows:
\[
D(t) = \{ p | p \text{ is a finite sequence of substitutions with nondecreasing time } \leq t \}
\]
For sets of times \( T \), \( A(T) \) and \( D(T) \) are defined coercively. Observe that in any sequence of \( A(t) \), all events occur at the same time. Also, \( A(t) \) and \( D(t) \) contain the empty sequence.

For sets of sequences \( U \) and \( V \), their concatenation and partial concatenation, written \( UV \) and \( U \cdot V \) respectively, are defined by
\[
UV = \{ uv | u \in U, v \in V \}, \text{ and } U \cdot V = U \cup UV.
\]
Partial concatenation is right-associative: \( U \cdot V \cdot W = U \cdot (V \cdot W) \).

Observation 7
1. \( A(t)^* = A(t) \) and \( D(t)^* = D(t) \).
2. \( \overline{A(t)} = A(t) \) and \( \overline{D(t)} = D(t) \).
3. \( A(s + t) = A(s) \cup A(t)_s \) and \( D(s + t) = D(s) \cdot D(t)_s \).
4. For \( s \leq t \), \( A(s) \subseteq A(t) \) and \( D(s) \subseteq D(t) \).
5. \( A(s) \subseteq D(t) \)
6. \( A(0) \setminus [m/x] = A(0) \) and \( D(0) \setminus [m/x] = D(0) \)

The sets \( A(t) \) and \( D(t) \) are first used in Sections 3.1.1 and 2.1, respectively.

1.5 Basic Theorems on Executions

We derive two basic theorems on executions in this section.
1.5.1 Evolution

Theorem 5 (Evolution) \( f^s \xrightarrow{t,a} h \equiv f^{s+t,a} \xrightarrow{t,a} h \)

Proof: First, we dispose of the case where \( h = \bot \). In that case, both sides of the equivalence are true (because \( f^{t,a} \xrightarrow{t,a} \bot \), for all \( t, a \) and \( f \)). Henceforth, assume that \( h \neq \bot \).

If \( s = 0 \), the result follows by appealing to the proposition \( f^0 = f \). Henceforth, let \( s > 0 \). If \( f \) is \( 0, M(m) \), if or let, then \( f^s \) is \( \bot \), which has transition only to \( \bot \). Since \( h \neq \bot \), \( f \) is not one of the given expressions. If \( f \) is \( R\text{timer}(u) \), then \( R\text{timer}(u)^s \xrightarrow{t,a} h \) where \( h \neq \bot \) arises only when \( u \geq s + t \). Then, the result is easy to see.

We give proofs by structural induction for expressions of the form \( (f \mid g) \), \((f >x> g)\) and \((f <x< g)\) in place of \( f \).

\( (f \mid g) \): Suppose \( (f \mid g)^s \xrightarrow{t,a} h \). From definition, that is \( f^s \mid g^s \xrightarrow{s+t,a} h \). Assume, without loss in generality, that this is deduced by applying \((\text{SYM1})\), i.e.,

\[
\begin{align*}
  f^s &\xrightarrow{t,a} f', \text{ and } h = f' \mid (g^s)^t \\
  \Rightarrow \text{(induction)} &\Rightarrow f^{s+t,a} \xrightarrow{t,a} f' \\
  \Rightarrow \text{(Apply (SYM1))} &\Rightarrow f \mid g \xrightarrow{s+t,a} f' \mid (g^s)^t \\
  \Rightarrow \{g^{s+t} = (g^s)^t\} &\Rightarrow f \mid g \xrightarrow{s+t,a} f' \mid (g^s)^t \\
  \Rightarrow \{h = f' \mid (g^s)^t\} &\Rightarrow f \mid g \xrightarrow{s+t,a} h
\end{align*}
\]

In the other direction, suppose that \( f \mid g \xrightarrow{s+t,a} h \). Assume, without loss in generality, that this is deduced by applying \((\text{SYM1})\), i.e.,

\[
\begin{align*}
  f \xrightarrow{s+t,a} f', \text{ and } h = f' \mid g^{s+t} \\
  \Rightarrow \text{(induction)} &\Rightarrow f^s \xrightarrow{t,a} f' \\
  \Rightarrow \text{(Apply (SYM1))} &\Rightarrow f^s \mid g^s \xrightarrow{t,a} f' \mid (g^s)^t \\
  \Rightarrow \{g^{s+t} = (g^s)^t\} &\Rightarrow f^s \mid g^s \xrightarrow{t,a} f' \mid (g^s)^t
\end{align*}
\]
\[
fs \to_{t,a} f' \to_{t_a} g^{s+t}
\]
\[
\Rightarrow \{ h = f' \to_{t_a} g^{s+t}\}
\]
\[
fs \to_{t_a} h
\]
\[
\Rightarrow \{\text{definition}\}
\]
\[
(f \mid g)s \to_{t,a} h
\]
\[
(f > x> g): \text{Suppose } (f > x> g)s \to_{t_a} h. \text{ From definition, that is } f^s > x> g \to_{t_a} h.
\]
There are two cases, depending on whether or not \(a\) is a publication event.

First assume \(a \neq !v\), and by rule (SEQ1N):

\[
fs \to_{t,a} f', \text{ and } h = f' > x> g.
\]

Now,

\[
fs \to_{t,a} f'
\]
\[
\Rightarrow \{\text{induction}\}
\]
\[
f^{s+t,a} \to_{t_a} f'
\]
\[
\Rightarrow \{\text{Apply (SEQ1N)}\}
\]
\[
f > x> g \to_{s+t,a} f' > x> g
\]
\[
\Rightarrow \{ h = f' > x> g\}
\]
\[
f > x> g \to_{s+t,a} h
\]

Next assume \(a = !v\), and by rule (SEQ1V):

\[
fs \to_{t,m} f', \text{ and } h = (f' > x> g) \mid [m/x].g.
\]

Now,

\[
fs \to_{t,m} f'
\]
\[
\Rightarrow \{\text{induction}\}
\]
\[
f^{s+t,v} \to_{t,v} f'
\]
\[
\Rightarrow \{\text{Apply (SEQ1V)}\}
\]
\[
f > x> g \to_{s+t,v} (f' > x> g) \mid [m/x].g
\]
\[
\Rightarrow \{ h = (f' > x> g) \mid [m/x].g\}
\]
\[
f > x> g \to_{s+t,v} h
\]

In the other direction, suppose that \(f > x> g \to s+t,a\) \(h\). Again there are two cases corresponding to the presence of a publication. First assume \(a \neq !v\), and by rule (SEQ1N):

\[
f \to_{s+t,a} f', \text{ and } h = f' > x> g
\]

Now,
$f \xrightarrow{s+t,a} f'$

$\Rightarrow \{\text{induction}\}
\begin{align*}
  f_s t.a \xrightarrow{} f' \\
\end{align*}$

$\Rightarrow \{\text{Apply (SEQ1N)}\}
\begin{align*}
  f^s >x> g t.a \xrightarrow{} f' >x> g \\
  \Rightarrow \{h = f' >x> g\}
  f^s >x> g t.a \xrightarrow{} h \\
  \Rightarrow \{\text{definition}\}
  (f >x> g)^s t.a \xrightarrow{} h
\end{align*}$

Next assume $a = !v$, and by rule (SEQ1V):

$f \xrightarrow{s+t,lv} f'$, and $h = (f' >x> g) | [m/x].g$

Now,

$\begin{align*}
  f \xrightarrow{s+t,lv} f' \\
  \Rightarrow \{\text{induction}\}
  f_s t,lm \xrightarrow{} f' \\
\end{align*}$

$\Rightarrow \{\text{Apply (SEQ1V)}\}
\begin{align*}
  f^s >x> g t,lm \xrightarrow{} (f' >x> g) | [m/x].g \\
  \Rightarrow \{h = (f' >x> g) | [m/x].g\}
  f^s >x> g t,lm \xrightarrow{} h \\
  \Rightarrow \{\text{definition}\}
  (f >x> g)^s t,lm \xrightarrow{} h
\end{align*}$

$(f <x< g):$ Suppose $(f <x< g)^s t.a \xrightarrow{} h$. From definition, that is $f^s <x< g^s t.a \xrightarrow{} h$. First assume the transition is due to a transition of $f^s$ by (ASYM1N), i.e.,

$f^s t.a \xrightarrow{} f'$, and $h = f' <x< (g^s)^t$.

Now,

$\begin{align*}
  f^s t.a \xrightarrow{} f' \\
  \Rightarrow \{\text{induction}\}
  f s+t,a \xrightarrow{} f' \\
\end{align*}$

$\Rightarrow \{\text{Apply (ASYM1N)}\}
\begin{align*}
  f <x< g s+t,a \xrightarrow{} f' <x< g^{s+t} \\
  \Rightarrow \{g^{s+t} = (g^s)^t\}
  f <x< g s+t,a \xrightarrow{} f' <x< (g^s)^t \\
  \Rightarrow \{h = f' <x< (g^s)^t\}
  f <x< g s+t,a \xrightarrow{} h
\end{align*}$
Next assume the transition is due to a transition of $g^s$. There are two cases, depending on whether or not $a$ is a publication event. First assume $a \neq !v$, and by rule (ASYM2):

$$g^s \xrightarrow{t,a} g' \text{ and } h = (f^s)^t < x < g'.$$

Now,

$$g^s \xrightarrow{t,a} g' \Rightarrow \{\text{induction}\}$$

$$g^{s+t,a} \xrightarrow{} g'$$

$$\Rightarrow \{\text{Apply (ASYM2)}\}$$

$$f < x < g^{s+t,a} \xrightarrow{} f^{s+t} < x < g'$$

$$\Rightarrow \{f^{s+t} = (f^s)^t\}$$

$$f < x < g^{s+t,a} \xrightarrow{} (f^s)^t < x < g'$$

$$\Rightarrow \{h = (f^s)^t < x < g'\}$$

$$f < x < g^{s+t,a} \xrightarrow{} h$$

Finally assume $a = !v$, and by rule (ASYM1V):

$$g^s \xrightarrow{t,v} g' \text{ and } h = [m/x].(f^s)^t.$$

Now,

$$g^s \xrightarrow{t,v} g' \Rightarrow \{\text{induction}\}$$

$$g^{s+t,v} \xrightarrow{} g'$$

$$\Rightarrow \{\text{Apply (ASYM1V)}\}$$

$$f < x < g^{s+t,v} \xrightarrow{} [m/x].f^{s+t}$$

$$\Rightarrow \{f^{s+t} = (f^s)^t\}$$

$$f < x < g^{s+t,v} \xrightarrow{} [m/x].(f^s)^t$$

$$\Rightarrow \{h = [m/x].(f^s)^t\}$$

$$f < x < g^{s+t,v} \xrightarrow{} h$$

In the other direction, suppose that $f < x < g^{s+t,a} \xrightarrow{} h$. First assume the transition is due to a transition of $f$ by (ASYM1N), i.e.,

$$f^{s+t,a} \xrightarrow{} f', \text{ and } h = f' < x < g^{s+t}$$

Now,
CHAPTER 1. INTRODUCTION

\[ f^{s+\alpha} \Rightarrow f' \]
\[ f^{s} \xrightarrow{t,\alpha} f' \]
\[ \{ \text{induction} \} \]
\[ f^{s} \xrightarrow{t,\alpha} f' \]
\[ \{ \text{Apply (ASYM1N)} \} \]
\[ f^{s} < x < g^{s} \xrightarrow{t,\alpha} f' < x < (g^{s})^{t} \]
\[ \{ (g^{s})^{t} = g^{s+t} \} \]
\[ f^{s} < x < g^{s} \xrightarrow{t,\alpha} f' < x < g^{s+t} \]
\[ \{ h = f' < x < g^{s+t} \} \]
\[ f^{s} < x < g^{s} \xrightarrow{t,\alpha} h \]
\[ \{ \text{definition} \} \]
\[ (f < x < g)^{s} \xrightarrow{t,\alpha} h \]

Next assume the transition is due to \( g \). There are two cases depending on whether or not \( a \) is a publication event. First assume \( a \neq !v \), and by rule (ASYM2):

\[ g^{s+t,\alpha} \xrightarrow{t} g', \text{ and } h = f^{s+t} < x < g' \]

Now,

\[ g^{s+t,\alpha} \xrightarrow{t} g' \]
\[ g^{s} \xrightarrow{t,\alpha} g' \]
\[ \{ \text{Apply (ASYM2)} \} \]
\[ f^{s} < x < g^{s} \xrightarrow{t,\alpha} (f')^{t} < x < g' \]
\[ \{ (f^{s})^{t} = f^{s+t} \} \]
\[ f^{s} < x < g^{s} \xrightarrow{t,\alpha} f^{s+t} < x < g' \]
\[ \{ h = f^{s+t} < x < g' \} \]
\[ f^{s} < x < g^{s} \xrightarrow{t,\alpha} h \]
\[ \{ \text{definition} \} \]
\[ (f < x < g)^{s} \xrightarrow{t,\alpha} h \]

Finally assume \( a = !v \), and by rule (ASYM1V):

\[ g^{s+t,!v} \xrightarrow{t} g', \text{ and } h = [m/x].f^{s+t}. \]

Now,

\[ g^{s+t,!v} \xrightarrow{t} g' \]
\[ g^{s} \xrightarrow{t,!,m} g' \]
\[ \{ \text{Apply (ASYM1V)} \} \]
\[ f^s < x < g^s \overset{t_{lm}}{\rightarrow} [m/x].(f^s)^t \]

\[ \Rightarrow \{ (f^s)^t = f^{s+t} \} \]

\[ f^s < x < g^s \overset{t_{lm}}{\rightarrow} [m/x].f^{s+t} \]

\[ \Rightarrow \{ h = [m/x].f^{s+t} \} \]

\[ f^s < x < g^s \overset{t_{lm}}{\rightarrow} h \]

\[ \Rightarrow \{ \text{definition} \} \]

\[ (f < x < g)^s \overset{t_{lm}}{\rightarrow} h \]

**Theorem 6 (Shift)** \( f^t \not\Rightarrow g \) iff \( \not\Rightarrow f \).

**Proof:**

If \( u = \epsilon, u_t = \epsilon \) and \( f = f^t = g \). Otherwise, \( u = (s, a)u'_s \), and

\[ f^t \overset{s,a}{\rightarrow} f' \overset{u'}{\Rightarrow} g \]

\[ \equiv \{ \text{Thm. 5 (Evolution) on page 10} \} \]

\[ f \overset{t+s,a}{\rightarrow} f' \overset{u'}{\Rightarrow} g \]

\[ \equiv \{ \text{definition of executions} \} \]

\[ f \overset{(t+s,a)u'_{s+}}{\Rightarrow} g \]

\[ \equiv \{ \text{Obs. 1 on page 6} \} \]

\[ f \overset{(t+s,a)(u')_t}{\Rightarrow} g \]

\[ \equiv \{ \text{definition of shifting} \} \]

\[ f \overset{(s,a)u'_s}{\Rightarrow} g \]

**Observation 8** \( u \in \lfloor f^t \rfloor \equiv u_t \in \lfloor f \rfloor \)

**Proof:** Follows from Theorem 6, page 15.

### 1.5.2 Substitution Independence

The goal of this section is to show that in an execution of an Orc expression, a pair of adjacent events, \((t, a)(t, b)\), can be swapped, given that \(a\) is not a substitution and \(b\) is a substitution. First, we prove a lemma.

**Lemma 2** Suppose \( f \overset{(0,a)}{\rightarrow} f' \), where \(a\) is not a substitution. Then, \([m/x].f \overset{(0,a)}{\rightarrow} [m/x].(f')\).

**Proof:** Proof is by induction on the structure of \( f \).

- **0:** The expression 0 only transitions as a result of rule (\textsc{Subst}).
- **?k:** By the operational semantics, the only transition of ?k is by rule (\textsc{Return}), where \(?k \overset{t_{lm}}{\rightarrow} 0\). The result follows because \([m/x].?k = ?k\) and \([m/x].0 = 0\).
• \( M(m) \): By the operational semantics, the only transition of \( M(m) \) is by rule (\text{Call}) where \( M(m) \xrightarrow{\text{Call}} M(m) \). The result follows because \( [m/x].M(m) = M(m) \) and \([m/x].?k = ?k\).

• \( M(x) \): The expression \( M(x) \) only transitions as a result of rule (\text{Subst}).

• \( f | g \): Without loss in generality, suppose that \( f \xrightarrow{(0,a)} f' \), so that \( f | g \xrightarrow{(0,a)} f' | g \). We show \([m/x].(f | g) \xrightarrow{(0,a)} [m/x].(f' | g)\).

\[
\begin{align*}
[&m/x].(f | g) \\
= & \{\text{from substitution rules}\} \\
[&m/x].f | [m/x].g \\
\xrightarrow{(0,a)} & \{\text{from } f \xrightarrow{(0,a)} f', \text{ inductively, } [m/x].f \xrightarrow{(0,a)} [m/x].f'\} \\
& \{\text{apply rule (SYM1) from operational semantics}\} \\
[&m/x].f' | ([m/x].g)^0 \\
= & \{\text{simplify the last term}\} \\
[&m/x].f' | [m/x].g \\
= & \{\text{from substitution rules}\} \\
[&m/x].(f' | g)
\end{align*}
\]

• \( f > x > g \): We have two proofs for the two rules (\text{SEQ1N}) and (\text{SEQ1V}).

Case 1) Suppose \( f \xrightarrow{(0,a)} f' \) and (\text{SEQ1N}) was applied in deducing \( f > x > g \xrightarrow{(0,a)} f' > x > g \).

First we consider the case where substitution is made to the bound variable \( x \). We show \([m/x].(f > x > g) \xrightarrow{(0,a)} [m/x].(f' > x > g)\).

\[
\begin{align*}
[&m/x].(f > x > g) \\
= & \{\text{from substitution rules}\} \\
([&m/x].f) > x > g \\
\xrightarrow{(0,a)} & \{\text{from } f \xrightarrow{(0,a)} f', \text{ inductively, } [m/x].f \xrightarrow{(0,a)} [m/x].f'\} \\
& \{\text{apply rule (SEQ1N) from operational semantics}\} \\
([&m/x].f') > x > g \\
= & \{\text{from substitution rules}\} \\
[&m/x].(f' > x > g)
\end{align*}
\]

Next, consider the case where substitution is made to variable \( y \), \( y \neq x \). We show \([m/y].(f > x > g) \xrightarrow{(0,a)} [m/y].(f' > x > g)\).

\[
\begin{align*}
[&m/y].(f > x > g) \\
= & \{\text{from substitution rules}\} \\
([&m/y].f) > x > ([&m/y].g) \\
\xrightarrow{(0,a)} & \{\text{from } f \xrightarrow{(0,a)} f', \text{ inductively, } [m/y].f \xrightarrow{(0,a)} [m/y].f'\} \\
& \{\text{apply rule (SEQ1N) from operational semantics}\}
\end{align*}
\]
Case 2) Suppose \( f \xrightarrow{(0,\text{ln})} f' \) and (SEQ1V) was applied in deducing \( f > x > g \xrightarrow{(0,\tau)} f' > x > g \mid [n/x].g \).

First we consider the case where substitution is made to the bound variable \( x \). We show \([m/x].(f > x > g) \xrightarrow{(0,\tau)} [m/x].(f' > x > g \mid [n/x].g)\).

\[
[m/x].(f > x > g) = \begin{cases} 
\{\text{from substitution rules}\} 
[m/y].(f' > x > g) 
\end{cases}
\]

\[
\xrightarrow{(0,\tau)} \begin{cases} 
\{\text{from } f \xrightarrow{(0,\text{ln})} f', \text{ inductively, } [m/x].f \xrightarrow{(0,\text{ln})} [m/x].f' \}; 
\text{ apply rule (SEQ1V) from operational semantics}\} 
([m/x].f') > x > g \mid [n/x].g 
\end{cases}
\]

\[
= \begin{cases} 
\{[n/x].g \text{ has no free occurrence of } x; \text{ so } [m/x].([n/x].g) = [n/x].g\} 
([m/x].f') > x > g \mid [m/x].([n/x].g) 
\end{cases}
\]

\[
= \begin{cases} 
\{\text{from substitution rules, } ([m/x].f') > x > g = [m/x].(f' > x > g)\} 
([m/x].(f' > x > g) \mid [m/x].([n/x].g) 
\end{cases}
\]

\[
= \begin{cases} 
\{\text{from substitution rules}\} 
[m/x].(f' > x > g \mid [n/x].g) 
\end{cases}
\]

Next, consider the case where substitution is made to variable \( y \), \( y \neq x \). We show \([m/y].(f > x > g) \xrightarrow{(0,\tau)} [m/y].(f' > x > g \mid [n/x].g)\).

\[
[m/y].(f > x > g) = \begin{cases} 
\{\text{from substitution rules}\} 
([m/y].f) > x > [m/y].g 
\end{cases}
\]

\[
\xrightarrow{(0,\tau)} \begin{cases} 
\{\text{from } f \xrightarrow{(0,\text{ln})} f', \text{ inductively, } [m/y].f \xrightarrow{(0,\text{ln})} [m/y].f' \}; 
\text{ apply rule (SEQ1V) from operational semantics}\} 
([m/y].f') > x > [m/y].g \mid [n/x].([m/y].g) 
\end{cases}
\]

\[
= \begin{cases} 
\{\text{from substitution rules, } ([m/y].f') > x > [m/y].g = [m/y].(f' > x > g)\} 
[m/y].(f' > x > g) \mid [n/x].([m/y].g) 
\end{cases}
\]

\[
= \begin{cases} 
\{x \neq y; \text{ so } [n/x].([m/y].g) = [m/y].([n/x].g)\} 
[m/y].(f' > x > g) \mid [m/y].([n/x].g) 
\end{cases}
\]

\[
= \begin{cases} 
\{\text{from substitution rules}\} 
[m/y].(f' > x > g \mid [n/x].g) 
\end{cases}
\]

\[
\bullet f < x < g: \text{ We have three proofs for the three rules (ASYM1), (ASYM2N), (ASYM2V).}
\]

Case 1) Suppose \( f \xrightarrow{(0,a)} f' \) and (ASYM1) was applied in deducing \( f < x < g \xrightarrow{(0,a)} f' < x < g \).

First, we consider the case where substitution is made to the bound variable \( x \). We show \([m/x].(f < x < g) \xrightarrow{(0,a)} [m/x].(f' < x < g)\).
Next, consider the case where substitution is made to variable $y$, $y \neq x$. We show $[m/y].(f < x < g) \rightarrow [m/y].(f' < x < g)$.

$[m/y].(f < x < g) = \{\text{from substitution rules}\}
\quad f < x < [m/y].g
\quad \rightarrow \{\text{using induction on } f \rightarrow [m/y].f, [m/y].g' \}
\quad [m/y].g' < x < [m/y].g
\quad = \{\text{from substitution rules}\}
\quad [m/y].(f' < x < g)$

Case 2) Suppose $g' \rightarrow g'$, $a$ is not a publication, and (ASYM2N) was applied in deducing $f < x < g \rightarrow f < x < g'$.

First, we consider the case where substitution is made to the bound variable $x$. We show $[m/x].(f < x < g) \rightarrow [m/x].(f < x < g')$.

$[m/x].(f < x < g) = \{\text{from substitution rules}\}
\quad f < x < [m/x].g
\quad \rightarrow \{\text{using induction on } g \rightarrow g', [m/x].g \rightarrow [m/x].g' \}
\quad [m/x].g' < x < [m/x].g'
\quad = \{\text{from substitution rules}\}
\quad [m/x].(f < x < g')$

Next, we consider the case where substitution is made to variable $y$, $y \neq x$. We show $[m/y].(f < x < g) \rightarrow [m/y].(f < x < g')$.

$[m/y].(f < x < g) = \{\text{from substitution rules}\}
\quad f < x < [m/y].g
\quad \rightarrow \{\text{using induction on } g \rightarrow g', [m/y].g \rightarrow [m/y].g' \}
\quad [m/y].g' < x < [m/y].g'
\quad = \{\text{from substitution rules}\}
\quad [m/y].(f < x < g')$
CHAPTER 1. INTRODUCTION

\[ [m/y].f < x < [m/y].g' \]
\[ [m/y].(f < x < g') \]

Case 3) Suppose \( g \stackrel{(0,1)}{\rightarrow} g' \) and (ASYM2V) was applied in deducing \( f < x < g \stackrel{(0,\tau)}{\rightarrow} [n/x].f \).

First, we consider the case where substitution is made to the bound variable \( x \). We show \([m/x].(f < x < g) \stackrel{(0,\tau)}{\rightarrow} [m/x].[n/x].f\).

\[
[m/x].(f < x < g) = \{\text{from substitution rules}\}
\]
\[
[m/y].f < x < [m/y].g
\]

\[
\stackrel{(0,\tau)}{\rightarrow} \{\text{using induction on } g \stackrel{(0,1)}{\rightarrow} g', [m/x].g \stackrel{(0,1)}{\rightarrow} [m/x].g'; \text{ apply rule (ASYM2V) from operational semantics}\}
\]
\[ [n/x].(f^0) \]
\[ = \{\text{simplify}\} \]
\[ [n/x].f \]
\[ = \{[n/x].f \text{ has no free occurrence of } x\}
\]
\[ [m/x].([n/x].f) \]

Next, we consider the case where substitution is made to variable \( y \), \( y \neq x \).

We show \([m/y].(f < x < g) \stackrel{(0,\tau)}{\rightarrow} [m/y].[n/x].f\).

\[
[m/y].(f < x < g) = \{\text{from substitution rules}\}
\]
\[
[m/y].f < x < [m/y].g
\]

\[
\stackrel{(0,\tau)}{\rightarrow} \{\text{using induction on } g \stackrel{(0,1)}{\rightarrow} g', [m/y].g \stackrel{(0,1)}{\rightarrow} [m/y].g'; \text{ apply rule (ASYM2V) from operational semantics}\}
\]
\[ [n/x].([(m/y).f^0) \]
\[ = \{\text{simplify}\} \]
\[ [n/x].([m/y].f) \]
\[ = \{[n/x].f \text{ and } [m/y].f \text{ are substitutions to different variables}\}
\]
\[ [m/y].([n/x].f) \]

\[ \square \]

**Theorem 7** Let \( p(t,a)(t,b)q \) be an execution of expression \( g \), where \( a \) is not a substitution and \( b \) is a substitution \([m/x]\). Then, \( p(t,b)(t,a)q \) is also an execution of \( g \).

**Proof:** Let \( g \stackrel{p}{\Rightarrow} f \stackrel{(t,a)}{\Rightarrow} f' \stackrel{(0,b)}{\Rightarrow} f'' \), where \( a \) is not a substitution and \( b \) is. We show that \( g \stackrel{p}{\Rightarrow} f \stackrel{(t,b)}{\Rightarrow} (0,a) \stackrel{(0,a)}{\Rightarrow} f'' \). It is is sufficient to show for any expression \( f \) that \( f \stackrel{(t,a)}{\Rightarrow} f' \stackrel{(0,b)}{\Rightarrow} f'' \) implies \( f \stackrel{(t,b)}{\Rightarrow} (0,a) \stackrel{(0,a)}{\Rightarrow} f'' \).

\[
\Rightarrow \{\text{from the Evolution theorem, Theorem 5, page 10, } f \stackrel{(t,a)}{\Rightarrow} f' \text{ implies } f \stackrel{(0,a)}{\Rightarrow} f'\}
CHAPTER 1. INTRODUCTION

\[ f^{t} (0, a) \rightarrow f' (0, b) \rightarrow f'' \]

\[ \Rightarrow \{ \text{from Lemma 2, page 15, } f^{t} (0, a) \rightarrow f' \text{ implies } [m/x].f^{t} (0, a) \rightarrow [m/x].(f') \} \]

\[ [m/x].f^{t} (0, a) \rightarrow [m/x].(f') \text{ and } f'' (0, b) \rightarrow f'' \]

\[ \Rightarrow \{ b = [m/x]. \text{ So, from } f' (0, b) \rightarrow f'', f'' = [m/x].(f') \} \]

\[ [m/x].f^{t} (0, a) \rightarrow f'' \Rightarrow \{ \text{Given } b = [m/x], f^{t} (t, b) \rightarrow [m/x].f^{t} \} \]

\[ f^{t} (t, b) \rightarrow [m/x].f^{t} (0, a) \rightarrow f'' \]

We can prove this theorem under a weaker restriction: \( a \) and \( b \) are not substitutions to the same variable. In that case, two substitutions, being applied at the same time to different variables, may be executed in either order. We don’t, however, need this generality for our subsequent proofs.

**Substitution Independent Set** Set \( U \) is substitution independent if

\[ p(t, a)(t, b)q \in U \text{ implies } p(t, b)(t, a)q \in U, \]

whenever \( a \) is not a substitution and \( b \) is a substitution.

**Observation 9**

1. For any expression \( f \), \([ f ]\) is substitution independent.

2. Let \( U \) be substitution independent, and \( p(0, b)q \in U \), where no event in \( p \) is a substitution and \( b \) is a substitution. Then, \((0, b)pq \in U\).

3. Union of substitution independent sets is substitution independent.

Proof: Part (1) follows from Theorem 7, page 19. Part (2) follows by applying induction on the length of \( p \). For \( p = \epsilon \), the result is immediate. Given \( p = r(0, a) \), from the definition of substitution independent set, \( r(0, b)(0, a)q \in U \), and inductively, \((0, b)r(0, a)q = (0, b)pq \in U\). Part (3) follows from the definition of substitution independent set.

**Lemma 3** Let \( U \) be a substitution independent set and \( a \) be a substitution at time 0. Then, \( U \setminus a = U \setminus a \).

Proof: We show \( U \setminus a \subseteq U \setminus a \) and \( U \setminus a \subseteq U \setminus a \).

- \( U \setminus a \subseteq U \setminus a \): We show that for any \( p \), where \( p \in U \setminus a \), \( \overline{p} \in U \setminus a \).

  \[ p \in U \setminus a \]

  \[ \Rightarrow \{ \text{definition of } \setminus a; \text{ note that } a.time = 0 \} \]

  \[ ap \in U \]

  \[ \Rightarrow \{ \text{definition of trace; } a \text{ is a substitution, so } \overline{a} = a \} \]

  \[ a\overline{p} \in U \]

  \[ \Rightarrow \{ \text{definition of } \setminus a; \text{ note that } a.time = 0 \} \]

  \[ \overline{p} \in U \setminus a \]
CHAPTER 1. INTRODUCTION

• $U \setminus a \subseteq U \setminus a$:

\[
\begin{align*}
\Rightarrow \quad & \{ \text{definition of } \setminus a; \text{note that } a.\text{time} = 0 \} \\
\Rightarrow \quad & \{ \text{definition of trace} \}
\end{align*}
\]

\[
\begin{align*}
\Rightarrow \quad & \{ \exists u, q : u \text{ is a sequence of } \tau \text{ at time } 0 \land \overline{q} = p : uaq \in U \} \\
\Rightarrow \quad & \{ \text{from Observation 9, page 20, part}(2), aUq \in U \} \\
\Rightarrow \quad & \{ \exists u, q : u \text{ is a sequence of } \tau \text{ at time } 0 \land \overline{q} = p : uaq \in U \} \\
\Rightarrow \quad & \{ \text{definition of } U \setminus a; \text{note that } a.\text{time} = 0 \} \\
\Rightarrow \quad & \{ \exists u, q : u \text{ is a sequence of } \tau \land \overline{q} = p : uq \in U \setminus a \} \\
\Rightarrow \quad & \{ \exists u = \epsilon, \overline{q} = p \} \\
\Rightarrow \quad & p \in U \setminus a
\end{align*}
\]

**Corollary 1** For any substitution $a$ at time 0, $\langle\langle a.f \rangle\rangle = \langle\langle f \rangle\rangle \setminus a$.

Proof:

\[
\begin{align*}
\langle\langle f \rangle\rangle \setminus a \\
= \{ \text{definition of trace} \}
\end{align*}
\]

\[
\begin{align*}
\{ [f] \setminus a \\
= \{ \text{from Observation 9, page 20, } [f] \text{ is substitution independent; applying Lemma 3, page 20} \}
\end{align*}
\]

\[
\begin{align*}
\{ uq \in U \setminus a \implies \overline{uq} \in U \setminus a \} \\
\{ \exists u, q : u \text{ is a sequence of } \tau \land \overline{q} = p : uq \in U \setminus a \} \\
\end{align*}
\]

\[
\begin{align*}
\{ \exists u = \epsilon, \overline{q} = p \} \\
p \in U \setminus a
\end{align*}
\]

1.6 Identities

In this section, we list certain identities over arbitrary expressions (i.e., with or without free variables), some of them similar to the laws of Kleene algebra. Proofs of the identities, using strong bisimulation, are given below. Other identities such as $f >x> \text{let}(x) \equiv f$, can also be proved using weak bisimulation.

Below, “$f$ is $x$-free” means that $x$ is not a free variable of $f$.

1. $f \mid 0 \sim f$
2. $f \mid g \sim g \mid f$
3. $f \mid (g \mid h) \sim (f \mid g) \mid h$
4. $f >x> (g >y> h) \sim (f >x> g) >y> h$, if $h$ is $x$-free.
5. $0 >x> f \sim 0$
6. \((f | g) >x> h \sim f >x> h | g >x> h\)

7. \((f | g) <x< h \sim (f <x< h) | g\), if \(g\) is \(x\)-free.

8. \((f >y> g) <x< h \sim (f <x< h) >y> g\), if \(g\) is \(x\)-free.

9. \((f <x< g) <y< h \sim (f <y< h) <x< g\),
   if \(g\) is \(y\)-free and \(h\) is \(x\)-free.

10. \(0 <x< b \sim b \gg 0\), where \(b\) is a site call or handle.

Proof:

1. \(f \mid 0 \sim f\).
   The only subexpression is \(f\). Subexpression \(0\) has no transition.
   \[
   f \xrightarrow{t,a} f' \Rightarrow \{\text{Sym1}\}
   f \mid 0 \xrightarrow{t,a} f' \mid 0^i
   \Rightarrow \{\text{definition of } 0^i\}
   f \mid 0 \xrightarrow{t,a} f' \mid 0
   \]
   And,
   \[
   f \xrightarrow{t,a} f'
   \]
   Assumed: \(f' \mid 0 \sim f'\).

2. \(f \mid g \sim g \mid f\).
   First, we consider the transitions of \(f\).
   \[
   f \xrightarrow{t,a} f' \Rightarrow \{\text{Sym1}\}
   f \mid g \xrightarrow{t,a} f' \mid g^i
   \]
   \[
   f \xrightarrow{t,a} f' \Rightarrow \{\text{Sym2}\}
   g \mid f \xrightarrow{t,a} g^i \mid f'
   \]
   Assumed: \(f' \mid g^i \sim g^i \mid f'\)
   The derivation with \(g\)'s transition is symmetric.

3. \(f \mid (g \mid h) \sim (f \mid g) \mid h\). We consider the transitions of \(f\), \(g\) and \(h\) in turn.
   \[
   (a) \ (\text{Transition of } f: f \xrightarrow{t,a} f')
   \]
\[
\begin{align*}
& f \xrightarrow{t,a} f' \\
& \Rightarrow \quad \{\text{Sym1}\} \\
& f \mid (g \mid h) \xrightarrow{t,a} f' \mid (g \mid h)^t \\
& \Rightarrow \quad \{\text{definition of } (g \mid h)^t\} \\
& f \mid (g \mid h) \xrightarrow{t,a} f' \mid (g' \mid h^t)
\end{align*}
\]

And,
\[
\begin{align*}
& f \xrightarrow{t,a} f' \\
& \Rightarrow \quad \{\text{Sym1}\} \\
& f \mid g \xrightarrow{t,a} f' \mid g^t \\
& \Rightarrow \quad \{\text{Sym1}\} \\
& (f \mid g) \mid h \xrightarrow{t,a} (f' \mid g') \mid h^t
\end{align*}
\]

Assumed: \( f' \mid (g' \mid h^t) \sim (f' \mid g'') \mid h^t \)

(b) (Transition of \( g \): \( g \xrightarrow{t,a} g' \))
\[
\begin{align*}
& g \xrightarrow{t,a} g' \\
& \Rightarrow \quad \{\text{Sym1}\} \\
& g \mid h \xrightarrow{t,a} g' \mid h^t \\
& \Rightarrow \quad \{\text{Sym2}\} \\
& f \mid (g \mid h) \xrightarrow{t,a} f' \mid (g' \mid h^t)
\end{align*}
\]

And,
\[
\begin{align*}
& g \xrightarrow{t,a} g' \\
& \Rightarrow \quad \{\text{Sym2}\} \\
& f \mid g \xrightarrow{t,a} f' \mid g' \\
& \Rightarrow \quad \{\text{Sym1}\} \\
& (f \mid g) \mid h \xrightarrow{t,a} (f' \mid g') \mid h^t
\end{align*}
\]

Assumed: \( f' \mid (g' \mid h^t) \sim (f' \mid g'') \mid h^t \)

(c) (Transition of \( h \): \( h \xrightarrow{t,a} h' \))
\[
\begin{align*}
& h \xrightarrow{t,a} h' \\
& \Rightarrow \quad \{\text{Sym2}\} \\
& g \mid h \xrightarrow{t,a} g' \mid h' \\
& \Rightarrow \quad \{\text{Sym2}\} \\
& f \mid (g \mid h) \xrightarrow{t,a} f' \mid (g' \mid h')
\end{align*}
\]

And,
\[
\begin{align*}
& h \xrightarrow{t,a} h' \\
& \Rightarrow \quad \{\text{Sym2}\} \\
& (f \mid g) \mid h \xrightarrow{t,a} (f \mid g') \mid h' \\
& \Rightarrow \quad \{\text{definition of } (f \mid g')\} \\
& (f \mid g) \mid h \xrightarrow{t,a} (f' \mid g') \mid h'
\end{align*}
\]

Assumed: \( f' \mid (g' \mid h') \sim (f' \mid g'') \mid h' \)
4. \( f > x > (g > y > h) \sim (f > x > g) > y > h \), provided \( h \) is \( x \)-free.

Only the transitions of \( f \) have corresponding transitions in \( f > x > (g > y > h) \).

And, only the transitions of \( f \) have corresponding transitions in \( f > x > g \),
and hence in \( (f > x > g) > y > h \). Therefore, we consider only the transitions
of \( f \), publications and non-publications.

(a) \( (f \xrightarrow{t,lm} f') \)

\[
\begin{align*}
  f & \xrightarrow{t,lm} f' \\
\Rightarrow & \{ \text{Seq1V} \} \\
  f & > x > (g > y > h) \xrightarrow{t,lm} f' > x > (g > y > h) | [m/x].(g > y > h)
\end{align*}
\]

And,

\[
\begin{align*}
  f & \xrightarrow{t,lm} f' \\
\Rightarrow & \{ \text{Seq1V} \} \\
  f & > x > g \xrightarrow{t,lm} f' > x > g | ([m/x].g)
\end{align*}
\]

\[
\begin{align*}
  & \{ \text{Seq1N} \} \\
  & (f > x > g) > y > h \xrightarrow{t,lm} (f' > x > g | [m/x].g) > y > h
\end{align*}
\]

We show \( f' > x > (g > y > h) | [m/x].(g > y > h) \sim (f' > x > g | [m/x].g) > y > h \)

\[
\begin{align*}
  f' & > x > (g > y > h) | [m/x].(g > y > h) \\
\Rightarrow & \{ \text{substitution distributes} \} \\
  & (f' > x > (g > y > h) | (m/x).g) > y > (m/x.h) \\
\Rightarrow & \{ h \text{ is } x \text{-free} \} \\
  f' & > x > (g > y > h) | (m/x).g) > y > h
\end{align*}
\]

And,

\[
\begin{align*}
  & \{ \text{distributivity law} \} \\
  & (f' > x > g | [m/x].g) > y > h \\
\sim & \{ Associativity \} \\
  f' & > x > (g > y > h) | (m/x).g) > y > h
\end{align*}
\]

(b) \( (f \xrightarrow{t,a} f', a \neq l,m) \)

\[
\begin{align*}
  f & \xrightarrow{t,a} f' \\
\Rightarrow & \{ \text{Seq1N} \} \\
  f & > x > (g > y > h) \xrightarrow{t,a} f' > x > (g > y > h)
\end{align*}
\]

And,

\[
\begin{align*}
  f & \xrightarrow{t,a} f' \\
\Rightarrow & \{ \text{Seq1N} \} \\
  f & > x > g \xrightarrow{t,a} f' > x > g \\
\Rightarrow & \{ \text{Seq1N} \} \\
  (f > x > g) > y > h \xrightarrow{t,a} (f' > x > g) > y > h
\end{align*}
\]

Assumed: \( f' > x > (g > y > h) \sim (f' > x > g) > y > h \), given \( h \) is \( x \)-free.
Corollary: \( f \gg (g \gg y > h) \sim (f \gg g) > y > h \)

5. \( 0 \gg f \sim 0, 0 \gg x > f \sim 0. \)

Only transitions of \( 0 \gg f \) and \( 0 \gg \) correspond to those of \( 0, \) and \( 0 \) has no transition.

6. \( (f \mid g) > x > h \sim f \gg x > h \mid g \gg x > h. \)

We consider only the transitions of \( f \) and \( g \) because transitions of \( h \) do not have corresponding transitions for either expression. By symmetry and the commutativity of \( \mid \) we need consider only the transitions of \( f. \)

(a) \( (f \xrightarrow{t,a} f', a \neq !m) \)

\[
\begin{align*}
& f \xrightarrow{t,a} f' \\
\Rightarrow & \{\text{Sym1}\} \\
& f \mid g \xrightarrow{t,a} f' \mid g^t \\
\Rightarrow & \{\text{Seq1N}\} \\
& (f \mid g) > x > h \xrightarrow{t,a} (f' \mid g^t) > x > h \\
\end{align*}
\]

And,

\[
\begin{align*}
& f \xrightarrow{t,a} f' \\
\Rightarrow & \{\text{Seq1N}\} \\
& f > x > h \xrightarrow{t,a} f' > x > h \\
\Rightarrow & \{\text{Sym1}\} \\
& f > x > h \mid g > x > h \xrightarrow{t,a} f' > x > h \mid (g > x > h)^t \\
\Rightarrow & \{\text{definition of } (g > x > h)^t\} \\
& f > x > h \mid g > x > h \xrightarrow{t,a} f' > x > h \mid g^t > x > h \\
\end{align*}
\]

Assumed: \( (f' \mid g^t) > x > h \sim f' \gg x > h \mid g^t > x > h. \)

(b) \( (f \xrightarrow{t,\tau, m} f') \)

\[
\begin{align*}
& f \xrightarrow{t,\tau, m} f' \\
\Rightarrow & \{\text{Seq1V}\} \\
& f > x > h \xrightarrow{t,\tau} f' > x > h \mid [m/x].h \\
\Rightarrow & \{\text{Sym1}\} \\
& f > x > h \mid g > x > h \xrightarrow{t,\tau} (f' > x > h \mid [m/x].h) \mid (g > x > h)^t \\
\Rightarrow & \{\text{definition of } (g > x > h)^t\} \\
& f > x > h \mid g > x > h \xrightarrow{t,\tau} (f' > x > h \mid [m/x].h) \mid (g^t > x > h) \\
\end{align*}
\]

And,

\[
\begin{align*}
& f \xrightarrow{t,\tau, m} f' \\
\Rightarrow & \{\text{Sym1}\} \\
& f \mid g \xrightarrow{t,\tau, m} f' \mid g^t \\
\Rightarrow & \{\text{Seq1V}\} \\
& (f \mid g) > x > h \xrightarrow{t,\tau} (f' \mid g^t) > x > h \mid [m/x].h \\
\end{align*}
\]
CHAPTER 1. INTRODUCTION

To see \((f' \gg x > h \downarrow [m/x].h) \downarrow g' \gg x > h \sim (f' \downarrow g') \gg x > h \downarrow [m/x].h\)

\((f' \downarrow g') \gg x > h \downarrow [m/x].h\)

\sim \{\text{distributivity}\}

\((f' \gg x > h \downarrow g' \gg x > h) \downarrow [m/x].h\)

\sim \{\text{associativity and commutativity of } \}

\((f' \gg x > h \downarrow [m/x].h) \downarrow g' \gg x > h\)

7. \((f \downarrow g) < x < h \sim (f < x < h) \downarrow g\), provided \(g\) is \(x\)-free.

There are four different kinds of transitions for each of the expressions:

transitions of \(f\), \(g\), publication of \(h\) and non-publication of \(h\).

(a) \(f \xrightarrow{t,a} f'\):

\[
\begin{align*}
& \Rightarrow \{\text{Sym1}\} \\
& f \downarrow g \xrightarrow{t,a} f' \downarrow g'
\end{align*}
\]

\[
\Rightarrow \{\text{Asym2}\}
\]

\[
(f \downarrow g) \lessdot x < h \xrightarrow{t,a} (f' \downarrow g') \lessdot x < h'
\]

And,

\[
\begin{align*}
& \Rightarrow \{\text{Asym2}\} \\
& f \lessdot x < h \xrightarrow{t,a} f' \lessdot x < h'
\end{align*}
\]

\[
\Rightarrow \{\text{Sym1}\}
\]

\[
(f \lessdot x < h) \downarrow g \xrightarrow{t,a} (f' \lessdot x < h') \downarrow g'
\]

Assumed: \((f' \downarrow g') \lessdot x < h' \sim (f' \lessdot x < h') \downarrow g'\)

(b) \(g \xrightarrow{t,a} g'\):

\[
\begin{align*}
& \Rightarrow \{\text{Sym2}\} \\
& f \downarrow g \xrightarrow{t,a} f' \downarrow g'
\end{align*}
\]

\[
\Rightarrow \{\text{Asym2}\}
\]

\[
(f \downarrow g) \lessdot x < h \xrightarrow{t,a} (f' \downarrow g') \lessdot x < h'
\]

And,

\[
\begin{align*}
& \Rightarrow \{\text{Sym2}\} \\
& g \xrightarrow{t,a} g'
\end{align*}
\]

\[
\Rightarrow \{\text{definition of } (f \lessdot x < h')\}
\]

\[
(f \lessdot x < h) \downarrow g \xrightarrow{t,a} (f' \lessdot x < h') \downarrow g'
\]

Since \(g\) is \(x\)-free, so is \(g'\). Assumed: \((f' \downarrow g') \lessdot x < h' \sim (f' \lessdot x < h') \downarrow g'\).

(c) \(h \xrightarrow{t,a} h', a \neq !m\):
\[ h \xrightarrow{t,a} h' \]

\[ \Rightarrow \quad \{ \text{Asym1N} \} \]

\[ (f \mid g) < x < h \xrightarrow{t,a} (f \mid g) < x < h' \]

\[ \Rightarrow \quad \{ \text{definition } (f \mid g)^t \} \]

\[ (f \mid g) < x < h \xrightarrow{t,a} (f^t \mid g^t) < x < h' \]

And,

\[ h \xrightarrow{t,a} h' \]

\[ \Rightarrow \quad \{ \text{Asym1N} \} \]

\[ f < x < h \xrightarrow{t} f^t < x < h' \]

\[ \Rightarrow \quad \{ \text{Sym1} \} \]

\[ (f < x < h) \mid g \xrightarrow{t,a} (f^t < x < h') \mid g^t \]

Given \( g \) is \( x \)-free, and assumed \( (f^t \mid g^t) < x < h' \sim (f^t < x < h') \mid g^t \).

(d) \( h \xrightarrow{t,m} h' \):

\[ h \xrightarrow{t,m} h' \]

\[ \Rightarrow \quad \{ \text{Asym1V} \} \]

\[ (f \mid g) < x < h \xrightarrow{t,T} [m/x].(f \mid g)^t \]

\[ \Rightarrow \quad \{ \text{definition of } (f \mid g)^t \} \]

\[ (f \mid g) < x < h \xrightarrow{t,T} [m/x].(f^t \mid g^t) \]

And,

\[ h \xrightarrow{t,m} h' \]

\[ \Rightarrow \quad \{ \text{Asym1V} \} \]

\[ f < x < h \xrightarrow{t,T} [m/x].f^t \]

\[ \Rightarrow \quad \{ \text{Sym1} \} \]

\[ (f < x < h) \mid g \xrightarrow{t,T} [m/x].f^t \mid g^t \]

To see that \( [m/x].(f^t \mid g^t) \sim [m/x].f^t \mid g^t \), we show they are equal.

\[ [m/x].(f^t \mid g^t) = \{ \text{substitution distributes} \} \]

\[ ([m/x].f^t) \mid ([m/x].g^t) \]

\[ = \{ g \text{ is } x\text{-free, and so is } g^t \} \]

\[ ([m/x].f^t) \mid g^t \]

8. \( (f > y > g) < x < h \sim (f < x < h) > y > g \), provided \( g \) is \( x \)-free.

The transitions of the left side expression correspond to those of \( (f > y > g) \) and \( h \), i.e., of \( f \) and \( h \). Similarly for the right side expression. We consider publication and non-publication transitions of \( f \) and \( h \) separately.

(a) \( (f \xrightarrow{t,a} f', a \neq !m) \)

\[ f \xrightarrow{t,a} f' \]

\[ \Rightarrow \quad \{ \text{Seq1N} \} \]
CHAPTER 1. INTRODUCTION

\[
f > y > g \implies f' > y > g
\]

\[
\Rightarrow \{\text{Asym2}\}
\]

\[
(f > y > g) \implies (f' > y > g) < x < h'
\]

And,

\[
f \implies f'
\]

\[
\Rightarrow \{\text{Asym2}\}
\]

\[
f < x < h \implies f' < x < h'
\]

\[
\Rightarrow \{\text{Seq1N}\}
\]

\[
(f < x < h) \implies (f' < x < h') > y > g
\]

Assumed: \((f' > y) > (f' < x < h') > y > g\).

(b) \((f \xrightarrow{t,m} f')\)

\[
f \xrightarrow{t,m} f'
\]

\[
\Rightarrow \{\text{Seq1V}\}
\]

\[
f > y > g \implies (f' > y > g | [m/y].g)
\]

\[
\Rightarrow \{\text{Asym2}\}
\]

\[
(f > y > g) < x < h \implies (f' > y > g | [m/y].g) < x < h'
\]

And,

\[
f \xrightarrow{t,m} f'
\]

\[
\Rightarrow \{\text{Asym2}\}
\]

\[
f < x < h \implies f' < x < h'
\]

\[
\Rightarrow \{\text{Seq1V}\}
\]

\[
(f < x < h) > y > g \implies (f' < x < h') > y > g | [m/y].g
\]

To see that \((f' > y) > (f' < x < h') > y > g | [m/y].g\)

\[
(f' > y > g | [m/y].g) < x < h'
\]

\[
\sim \{g \text{ is } x-\text{free. So, is } [m/y].g\}
\]

\[
(f' > y > g < x < h') | [m/y].g
\]

\[
\sim \{\text{this law}\}
\]

\[
(f' < x < h') > y > g | [m/y].g
\]

(c) \((h \xrightarrow{t,a} h', a \neq !m)\)

\[
h \xrightarrow{t,a} h'
\]

\[
\Rightarrow \{\text{Asym1N}\}
\]

\[
(f > y > g) < x < h \implies (f > y > g) < x < h'
\]

\[
\Rightarrow \{\text{definition of } (f > y > g)^t\}
\]

\[
(f > y > g) < x < h \implies (f' > y > g) < x < h'
\]

And,

\[
h \xrightarrow{t,a} h'
\]

\[
\Rightarrow \{\text{Asym1N}\}
CHAPTER 1. INTRODUCTION

29

\[ f < x < h \xrightarrow{t,a} f' < x < h' \]
\[ \Rightarrow \{ \text{Seq1N} \} \]
\[ (f < x < h) > y > g \xrightarrow{t,a} (f' < x < h') > y > g \]

Assumed: \((f' > y > g) < x < h' \sim (f' < x < h') > y > g\)

(d) \((h \xrightarrow{t,m} h')\)
\[ h \xrightarrow{t,m} h' \]
\[ \Rightarrow \{ \text{Asym1V} \} \]
\[ (f' > y > g) < x < h \xrightarrow{t,T} [m/x].(f > y > g)^t \]
\[ \Rightarrow \{ \text{definition of (f > y > g)^t} \} \]
\[ (f > y > g) < x < h \xrightarrow{t,T} [m/x].(f' > y > g) \]

And,
\[ h \xrightarrow{t,m} h' \]
\[ \Rightarrow \{ \text{Asym1V} \} \]
\[ f < x < h \xrightarrow{t,T} [m/x].f^t \]
\[ \Rightarrow \{ \text{Seq1N} \} \]
\[ (f < x < h) > y > g \xrightarrow{t,T} ([m/x].f^t) > y > g \]

To see that \([c/x](f' > y > g) \sim ([c/x]f' > y > g):\)
\[ [c/x](f' > y > g) = \{ \text{substitution distributes} \} \]
\[ ([c/x]f') > y > ([c/x]g) \]
\[ = \{ g \text{ is } x\text{-free. So, } [c/x]g = g \} \]
\[ (c[x]f') > y > g \]

9. \((f < x < g) < y < h \sim (f < y < h) < x < g\), provided \(g\) is \(y\)-free and \(h\) is \(x\)-free.

We have to consider the transitions corresponding to those of \(f, g\) and \(h\).
The roles of \(g\) and \(h\) are symmetric; so, we consider only the transitions of \(g\).

(a) \((f \xrightarrow{t,a} f')\)
\[ f \xrightarrow{t,a} f' \]
\[ \Rightarrow \{ \text{Asym2} \} \]
\[ f < x < g \xrightarrow{t,a} f' < x < g' \]
\[ \Rightarrow \{ \text{Asym2} \} \]
\[ (f < x < g) < y < h \xrightarrow{t,a} (f' < x < g') < y < h' \]

And,
\[ f \xrightarrow{t,a} f' \]
\[ \Rightarrow \{ \text{Asym2} \} \]
\[ f < y < h \xrightarrow{t,a} f' < y < h' \]
\[ \Rightarrow \{ \text{Asym2} \} \]
\[ (f < y < h) < x < g \xrightarrow{t,a} (f' < y < h') < x < g' \]
CHAPTER 1. INTRODUCTION

Assumed: \((f^t < x < g^t) \sim (f^t < y < h^t) < x < g^t\)

(b) \((g \xrightarrow{t,a} g', a \neq !m)\)

\[ g \xrightarrow{t,a} g' \]

\[ \Rightarrow \{ \text{Asym1N} \} \]

\[ f < x < g \xrightarrow{t,a} f^t < x < g' \]

\[ \Rightarrow \{ \text{Asym2} \} \]

\[ (f < x < g) < y < h \xrightarrow{t,a} (f^t < x < g') < y < h^t \]

And,

\[ g \xrightarrow{t,a} g' \]

\[ \Rightarrow \{ \text{Asym1N} \} \]

\[ (f < y < h) < x < g \xrightarrow{t,a} (f < y < h)^t < x < g' \]

\[ \Rightarrow \{ \text{definition of } (f < y < h)^t \} \]

\[ (f < y < h) < x < g \xrightarrow{t,a} (f^t < y < h^t) < x < g' \]

Assumed: \((f^t < x < g^t) \sim (f^t < y < h^t) < x < g^t\).

(c) \((g \xrightarrow{t,lm} g')\)

\[ g \xrightarrow{t,lm} g' \]

\[ \Rightarrow \{ \text{Asym1V} \} \]

\[ f < x < g \xrightarrow{\tau} [c/x]f^t \]

\[ \Rightarrow \{ \text{Asym2} \} \]

\[ (f < x < g) < y < h \xrightarrow{\tau} [c/x]f^t < y < h^t \]

And,

\[ g \xrightarrow{t,lm} g' \]

\[ \Rightarrow \{ \text{Asym1V} \} \]

\[ (f < y < h) < x < g \xrightarrow{\tau} [c/x](f < y < h)^t \]

\[ \Rightarrow \{ \text{definition of } (f < y < h)^t \} \]

\[ (f < y < h) < x < g \xrightarrow{\tau} [c/x](f^t < y < h^t) \]

To see that \([c/x]f^t < y < h^t \sim [c/x](f^t < y < h^t)\),

\[ [c/x](f^t < y < h^t) \]

\[ = \{ \text{substitution distributes} \} \]

\[ [c/x]f^t < y < [c/x]h^t \]

\[ \Rightarrow \{ h \text{ is } x\text{-free and so is } h^t \} \]

\[ [c/x]f^t < y < h^t \]

10. \(0 < x < M(v) \Rightarrow M(v) \gg 0\), for any site \(M\) and value \(v\).

The only transition of the constituent expression \(M(v)\) is \(M(v) \xrightarrow{0,\tau} k\).

\[ M(v) \xrightarrow{0,\tau} k \]

\[ \Rightarrow \{ \text{Asym1N} \} \]
\[ 0 < x < M(v) \stackrel{0,\tau}{\rightarrow} 0' < x < ?k \]
\[ \Rightarrow \{ \text{definition of 0'} \} \]
\[ 0 < x < M(v) \stackrel{0,\tau}{\rightarrow} 0 < x < ?k \]

And,
\[ M(v) \stackrel{0,\tau}{\rightarrow} ?k \]
\[ \Rightarrow \{ \text{Seq1N} \} \]
\[ M(v) \gg 0 \stackrel{0,\tau}{\rightarrow} ?k \gg 0 \]

Assumed: \( 0 < x < ?k \sim ?k \gg 0 \)

Corollaries

(a) \( f < x < M(v) \sim f \mid M(v) \gg 0 \), if \( f \) is \( x \)-free
\[ f < x < M(v) \]
\[ \sim \{ \text{proved} \} \]
\[ f \mid (0 < x < M(v)) \]
\[ \sim \{ f \text{ is } x\text{-free} \} \]
\[ f \mid M(v) \gg 0 \]
(b) \( f < x < 0 \sim f \), if \( f \) is \( x \)-free
\[ f < x < 0 \]
\[ \sim \{ \text{from above} \} \]
\[ f \mid (0 \gg 0) \]
\[ \sim \{ \text{proved} \} \]
\[ f \mid 0 \]
\[ \sim \{ \text{proved} \} \]
\[ f \]
Chapter 2

Combinators applied to Executions

In Section 2.1 we characterize the execution sets of the base expressions. In Section 2.2, we define $U \ast V$ where $U$ and $V$ are sets of executions and $\ast$ is any Orc combinator: $|$, $>_x$, and $<_x$. These definitions give the meaning function for each combinator when applied to sets. We prove results about monotonicity (Section 2.2.4, page 37) and distributivity (Section 2.2.5, page 39) of the combinators.

In subsequent sections of this chapter, we prove that for expressions $f$ and $g$, $[f \ast g] = [f] \ast [g]$.

Throughout we assume a fixed environment mapping $\Sigma$ and set of definitions $D$. We denote the set of times by $T$.

2.1 Base Expressions

For variable $x$ and set of sequences $A$, the exclusion of $x$ from $A$, written $A \setminus x$, is defined by

$$A \setminus x \triangleq \{ p \in A \mid \lbrack m/x \rbrack \text{ does not occur in } p \}.$$  

Theorem 8 The following sets characterize the executions of the base Orc expressions:

- $\lbrack 0 \rbrack = D(T)$
- $\lbrack ?k \rbrack = (\cup(t,m) : (t,m) \in k : D(t) \cdot (t, !m) \cdot \lbrack 0 \rbrack ) \cup (\cup \omega : \omega \in k : \lbrack 0 \rbrack )$
- $\lbrack M(m) \rbrack = (\cup k : k \in \Sigma(M,m) : D(0) \cdot (0, \tau) \cdot \lbrack ?k \rbrack )$
- $\lbrack M(x) \rbrack = (\cup t,m : t \in T, m \in V : D(t) \setminus x \cdot (t, [m/x]) \cdot \lbrack M(m) \rbrack )$

Proof:
• $[0] = D(T)$: Only the rule (SUBST) applies to 0, hence every execution is a finite sequence of substitution events with nondecreasing time.

• $[?k] = (\cup(t, m) : (t, m) \in k : D(t) \cdot (t, \! m) \cdot [0],) \cup (\cup \omega : \omega \in k : [0])$: The proof is by mutual inclusion.

$$\Rightarrow$$

- $[?k] \subseteq (\cup(t, m) : (t, m) \in k : D(t) \cdot (t, \! m) \cdot [0],) \cup (\cup \omega : \omega \in k : [0])$.
- Assume $p \in [?k]$. If $p = \epsilon$ the result follows because $\epsilon \in D(t)$ and $\epsilon \in [0]$. Otherwise let $p = a q_t$, so

$$?k \xrightarrow{t,a} f \Rightarrow f'.$$

By the operational semantics, the transition $\xrightarrow{t,a}$ must be due to either rule (RETURN) or (SUBST).

In the (RETURN) case, we have $(t, m) \in k$, $a = \! m$ and $f = 0$.

Since $z \in [0]$, it follows that $(t, \! m) q_t \in (t, \! m)[0]$, and also, since $\epsilon \in D(t)$, that $D(t)(t, \! m) q_t \in (t, \! m)[0]$. In the (SUBST) case, we have

$$?k \xrightarrow{t,[m/x]} ?k' \Rightarrow f'.$$

By Thm. 6 on page 15 we have $q_t \in [\epsilon k]$, and so by induction we have either $q_t \in (\cup(t, m) : (t, m) \in k : D(t) \cdot (t, \! m) \cdot [0],)$ or $q_t \in (\cup \omega : \omega \in k : [0])$. In either case prepending a substitution event at time $t$ preserves the inclusion because $a D(t) \subseteq D(t)$ and $a[0] \subseteq [0]$. Hence, for some $(t, m) \in k$, $p \in D(t) \cdot (t, \! m) \cdot [0]$. If $p = \epsilon$, then the result follows by induction on the length of $p$.

Otherwise, assume $p \in (\cup(t, m) : (t, m) \in k : D(t) \cdot (t, \! m) \cdot [0],)$. Since $\omega \in k$, the only rule that applies to $?k$ is (SUBST), and $?k' = ?k$ for any $t$. The result follows by induction on the length of $p$.

Suppose $p = q(t, \! m) r_t$, where $r \in [0]$ and $p \in D(t)$; the other cases follow by prefix-closure of execution sets. Otherwise $p \neq \epsilon$.

By the operational semantics, otherwise, $q = a q_s$ and $p = a p'_s$, where $a$ is a substitution event at time $s$, for some $s \leq t$, and $p' = q_t(t, \! m) r_t$. By induction, $p'_s \in [\epsilon k]$, and so by Thm 6 on page 15 $p' \in [?k']$. Finally,

$$?k \xrightarrow{a} ?k' \xrightarrow{p'_s} \Rightarrow f'.$$

• $[M(m)] = (\cup k : k \in \Sigma(M, m) : D(0) \cdot (0, \tau) \cdot [?k])$: The proof is by mutual inclusion.
\[ \text{Consider } p \in [M(m)]. \text{ If } p = \epsilon \text{ the result follows because } \epsilon \in D(0). \text{ Otherwise } \]
\[ M(m) \xrightarrow{t,a} f \Rightarrow f'. \]

By the operational semantics, the transition \( t,a \) must be due to either rule (CALL) or (SUBST).

In the (CALL) case, we have
\[ M(m) \xrightarrow{0,\tau} ?k \Rightarrow f'. \]
Since \( q \in [?k] \), it follows that \((0,\tau)q \in (0,\tau)[?k]\), and hence \((0,\tau)q \in D(0)(0,\tau)[?k]\).

In the (SUBST) case, we have
\[ M(m) \xrightarrow{0,[m/x]} M(m) \Rightarrow f'. \]
Note that the substitution must occur at time 0 because \( M(m)^t = \bot \) for \( t > 0 \). By induction, \( q \in (\cup k : k \in \Sigma(M,m) : D(0) \cdot (0,\tau) \cdot [?k]) \).

The result follows because \((0,[m/x])D(0) \subseteq D(0)\).

- \((\cup k : k \in \Sigma(M,m) : D(0) \cdot (0,\tau) \cdot [?k]) \subseteq [M(m)]\): Consider \( p \in (\cup k : k \in \Sigma(M,m) : D(0) \cdot (0,\tau) \cdot [?k]) \).

Assume \( p = q(0,\tau)r \), where \( q \in D(0) \) and \( r \in [?k] \); the other cases follow by prefix-closure of execution sets. If \( p = \epsilon \), the result follows by prefix-closure of execution sets. Otherwise \( p \neq \epsilon \). If \( q = \epsilon \), then \( p = (0,\tau)r \). Since \( r \in [?k] \), we have by rule (CALL)
\[ M(m) \xrightarrow{0,\tau} ?k \Rightarrow . \]

Otherwise \( q = aq' \) and \( p = ap' \), where \( a \) is a substitution event at time 0. By induction we have \( p' \in [M(m)] \), and the result follows from
\[ M(m) \xrightarrow{a} M(m) \xrightarrow{p'}. \]

- \([M(x)] = (\cup t,m : t \in T, m \in V : D(t) \setminus x \cdot (t,[m/x]) \cdot [M(m)]_t)\): The proof is by mutual inclusion.

- \([M(x)] \subseteq (\cup t,m : t \in T, m \in V : D(t) \setminus x \cdot (t,[m/x]) \cdot [M(m)]_t)\): Consider \( p \in [M(x)] \). If \( p = \epsilon \) the result follows because \( \epsilon \in D(t) \) for all times \( t \). Otherwise
\[ M(x) \xrightarrow{t,a} f \Rightarrow f'. \]

By the operational semantics, this transition must be by rule (SUBST).
CHAPTER 2. COMBINATORS APPLIED TO EXECUTIONS

35

We call \( p \) and subsequent algebraic manipulations. Let

2.2.1 Meaning of Symmetric Composition

2.2 Meanings of Execution Combinators

2.2.1 Meaning of Symmetric Composition

We define \( b \) means that \( a \) is a base event and \( a.time \leq b.time \).

\[
\begin{align*}
\epsilon & \mid v = \{\epsilon\}, & u & \mid \epsilon = \{\epsilon\}, \\
a u & \mid b v = [a \asymp b \rightarrow a(u \mid v)] \cup [a \leq b \rightarrow a(u \mid b v)] \cup [b \leq a \rightarrow b(a u \mid v)]
\end{align*}
\]

If the substitution is to a variable \( y \neq x \), then

\[
M(x) \xrightarrow{t,\{m/y\}} M(x) \xrightarrow{q} f'.
\]

By induction, \( q \in \langle \cup t, m : t \in T, m \in V : D(t)\{x\} \cdot [M(m)]_t \rangle \). Then for some \( s \in T \) and value \( m, q \in D(s)\{x\} \cdot [M(m)]_s \). Since \( q_t \in D(s + t)\{x\} \cdot (s + t, [m/y]) \cdot [M(m)]_{s + t} \). The result follows because \( (t, [m/y])D(s + t)\{x\} \subseteq D(s + t)\{x\} \).

Otherwise we have

\[
M(x) \xrightarrow{t,\{m/x\}} M(m) \xrightarrow{q} .
\]

Then \( q \in [[M(m)]] \) and so \( (t, [m/x])q_t \in (t, [m/x])[[M(m)]]_t \), from which the result follows.

\[
- (\cup t, m : t \in T, m \in V : D(t)\{x\} \cdot [M(m)]_t) \subseteq [[M(x)]]:
\]

Consider \( p \in \langle \cup t, m : t \in T, m \in V : D(t)\{x\} \cdot (t, [m/x]) \cdot [M(m)]_t \rangle \).

So, for some \( t \in T \) and value \( m, p \in D(t)\{x\} \cdot (t, [m/x]) \cdot [M(m)]_t \).

Assume \( p = q(t, [m/x])r_t \), where \( q \in D(t)\{x\} \) and \( r \in [[M(m)]] \); the other cases follow by prefix-closure of execution sets. If \( p = \epsilon \), the result follows by prefix-closure of execution sets. Otherwise, \( p \neq \epsilon \).

If \( q = \epsilon \), then by \( r \in [[M(m)]] \) and rule (Subst),

\[
M(x) \xrightarrow{t,\{m/x\}} M(m) \xrightarrow{r} .
\]

Otherwise, \( q = (s, [m/y])q' \) and \( p = (s, [m/y])p' \), where \( s \leq t \) and \( y \neq x \). By induction, \( p' \in [[M(x)]] \), and the result follows from

\[
M(x) \xrightarrow{s,\{m/y\}} M(x) \xrightarrow{p'} .
\]

2.2 Meanings of Execution Combinators

2.2.1 Meaning of Symmetric Composition

We introduce guarded set, a notational device, which simplifies our definition and subsequent algebraic manipulations. Let \( p \) be a predicate and \( S \) a set. Then

\[
[p \rightarrow S] = \begin{cases} S & \text{if } p \\
\emptyset & \text{otherwise} \end{cases}
\]

We call \( [p \rightarrow S] \) a guarded set, and predicate \( p \) its guard.

We define merge over two sequences that yields a non-empty set of sequences. The merge of \( u \) and \( v \), written as \( u \mid v \) is defined by the following two rules. Henceforth, \( a \simeq b \) means that \( a \) and \( b \) are identical substitution events, and \( a \preceq b \) means that \( a \) is a base event and \( a.time \leq b.time \).

\[
\begin{align*}
\epsilon & \mid v = \{\epsilon\}, & u & \mid \epsilon = \{\epsilon\}, \\
a u & \mid b v = [a \simeq b \rightarrow a(u \mid v)] \cup [a \preceq b \rightarrow a(u \mid b v)] \cup [b \leq a \rightarrow b(a u \mid v)]
\end{align*}
\]
We define \( \mid \) to be coercive so that

\[
U \mid V = (\cup u, v : u \in U, v \in V : u \mid v)
\]

Therefore \( \mid \) distributes over set union, and observation 2, as well as observations 3 and 4, page 7, apply.

### 2.2.2 Meaning of Sequential Composition

In this section, we deal with expressions of the form \( f \sev x \sev g \); variable \( x \) will be treated specially in this section. We write \textit{own-substitution} for a substitution to \( x \) and \textit{other-substitution} for any other substitution, i.e., made to a variable other than \( x \).

We define \( p \sev x \sev V \), for sequence \( p \) and set \( V \), as a set of sequences.

\[
\begin{align*}
\epsilon \sev x \phi &= \phi, \\
\epsilon \sev x V &= \{\epsilon\}, \text{for } V \neq \phi \\
ap \sev x V &= \begin{cases} 
  a(p \sev x V) & \text{if } c_1(a) \quad (SCD1) \\
  a(p \sev x V') & \text{if } c_2(a) \quad (SCD2) \\
  (t, \tau)(p \sev x V \mid V'') & \text{if } c_3(a) \quad (SCD3)
\end{cases}
\end{align*}
\]

where

- \( c_1(a) \) is “\( a \) is a non-publication base event or an own-substitution”,
- \( c_2(a) \) is “\( a \) is an other-substitution \((t, b)\)”; here \( V' = V \setminus (0, b) \),
- \( c_3(a) \) is “\( a \) is publication \((t, \!m)\)”; here \( V'' = V \setminus (0, [m/x]) \).

Coerce the definition for set \( U \):

\[
U \sev x V = (\cup u : u \in U : u \sev x V)
\]

The form of coercion for sequential composition is different from that for merge. Merge is defined with two sequences as arguments, whereas sequential composition has a set as a second argument.

### 2.2.3 Meaning of Asymmetric Composition

In this section, we deal with expressions of the form \( f \lsev x \lsev g \); variable \( x \) will be treated specially in this section. We write \textit{own-substitution} for a substitution to \( x \) and \textit{other-substitution} for any other substitution, i.e., made to a variable other than \( x \).

**Constrained Partial and Full Merge**

Let \( a \approx_x b \) mean that \( a \) and \( b \) are identical other-substitutions. As before, \( a \preceq b \) means that \( a \) is a base event and \( a.time \leq b.time \). Let \( b \preceq_x a \) denote that: (either \( b \) is a base event or an own-substitution) and \( b.time \leq a.time \). Define \textit{partial merge}, \( \lsem \), an extension of merge, over a pair of sequences.
CHAPTER 2. COMBINATORS APPLIED TO EXECUTIONS

\[ \epsilon \mid \_ = \{ \epsilon \} \]

\[ a u \mid b v = \{ a \approx \_ b \rightarrow a(u) \} \cup \{ a \preceq b \rightarrow a(u) \} \cup \{ b \preceq \_ a \rightarrow b(u) \} \]

Coerce the definition to sets \( U \) and \( V \):

\[ U \mid \_ V = (\cup u, v : u \in U, v \in V : u \mid \_ v) \]

Next, we define full merge, using a notation similar to guarded sets. Let \( \langle p \rightarrow S \rangle \) be set \( S \) if \( p \) is true and the empty set, \( \phi \), if \( p \) is false. Note that, unlike the guarded sets of Section 3.2, page 76, the default value here is the empty set, not \( \{ \epsilon \} \). Therefore, \( \langle p \rightarrow S \rangle \cup \langle \text{false} \rightarrow S' \rangle = \langle p \rightarrow S \rangle \).

Constrained full merge of \( u \) and \( v \), written as \( u \! + \! x \! v \), is a set of sequences defined as follows.

\[ u \! + \! x \! v = \begin{cases} \{ u \} & \text{if } u \text{ contains no substitution event} \\ \phi & \text{otherwise} \end{cases} \]

\[ \epsilon \! + \! x \! v = \begin{cases} \{ v \} & \text{if } v \text{ contains no other-substitution} \\ \phi & \text{otherwise} \end{cases} \]

\[ au \! + \! x \! bv = \langle a \approx \_ b \rightarrow a(u) \rangle \cup \langle a \preceq b \rightarrow a(u) \rangle \cup \langle b \preceq \_ a \rightarrow b(u) \rangle \]

Coerce the definition to sets \( U \) and \( V \):

\[ U \! + \! x \! V = (\cup u, v : u \in U, v \in V : u \! + \! x \! v) \]

Definition of Asymmetric Composition

Define

\[ d_0(u, v) \triangleq u \text{ and } v \text{ have the same sequence of other-substitutions} \]

\[ d_1(u, v) \triangleq u \text{ has no own-substitution and } v \text{ has no publication} \]

\[ d_2(u, v) \triangleq \begin{cases} u \text{ is of the form } u'(t, [m/x])u'', v \text{ is of the form } v'(t, !m)v'', \text{ and} \\ d_0(u', v'), d_1(u', v') \end{cases} \]

We now define \( u \prec x \prec v \).

\[ u \prec x \prec v = \begin{cases} u \mid \_ v & \text{if } d_1(u, v) \\ \{ u \! + \! x \! v \} & \text{if } d_2(u, v) \\ \phi & \text{otherwise} \end{cases} \]

\[ U \prec x \prec V = (\cup u, v : u \in U, v \in V : u \prec x \prec v) \]

2.2.4 Monotonicity

Theorem 9 For any Orc combinator \( \ast \), \( U \ast V \) is \( \phi \) if either \( U \) or \( V \) is \( \phi \). Further, suppose \( U \subseteq U' \) and \( V \subseteq V' \). Then, \( U \ast V \subseteq U' \ast V' \).

Proof: The first part follows from the definitions of meaning functions where \( \ast \) is either \( | \) or \( \prec x \prec \). Combinator \( \succ x \succ \) is coercive in its left argument; so, \( \phi \succ x \succ V = \phi \). To show \( U \succ x \succ \phi = \phi \), we show \( p \succ x \succ \phi = \phi \), for any \( p \). This can be proved by applying induction on the length of \( p \).
For the next part, combinator \( \mid \) is coercive in both of its arguments; so, \( U \subseteq U' \) and \( V \subseteq V' \) implies \( U \mid V \subseteq U' \ast V' \). Similar remarks apply for \(<x< . Combinator \( >x> \) is coercive in its left argument; so, \( U \subseteq U' \) implies \( U >x> V \subseteq U' >x> V \). Next, we prove that \( V \subseteq W \) implies \( U >x> V \subseteq U >x> W \).

If \( U = \phi \), from the definition, \( U >x> V = \phi = U >x> W \). For \( U \neq \phi \), let \( p \in U \). We show \( p >x> V \subseteq p >x> W \). Proof is by induction on the length of \( p \).

For \( p = \epsilon \), the result follows by inspection of the definition. Next, we prove \( ap >x> V \subseteq ap >x> W \). Consider three cases:

- **c1(a):**
  \[
  \begin{align*}
  ap >x> V & \\
  &= \{ \text{definition of meaning function} \} a(p >x> V) \\
  &\subseteq \{ V \subseteq W; \text{inductively, } p >x> V \subseteq p >x> W \} a(p >x> W) \\
  &= \{ \text{definition of meaning function} \} ap >x> W
  \end{align*}
  \]

- **c2(a):**
  \[
  \begin{align*}
  ap >x> V & \\
  &= \{ \text{definition of meaning function} \} a(p >x> V') \\
  &\subseteq \{ V \subseteq W; \text{from the definition of } \{0, b\} \text{ from Section 1.3.1, page 6, } V' \subseteq W' \} a(p >x> W') \\
  &= \{ \text{definition of meaning function} \} ap >x> W
  \end{align*}
  \]

- **c3(a):**
  \[
  \begin{align*}
  ap >x> V & \\
  &= \{ \text{definition of meaning function} \} a(p >x> V | (V''))_t) \\
  &\subseteq \{ V \subseteq W; \text{inductively, } p >x> V \subseteq p >x> W \} a(p >x> W | (V''))_t) \\
  &= \{ \text{from the definition of } \{0, b\} \text{ from Section 1.3.1, page 6, } V'' \subseteq W'' \} a(p >x> W | (W''))_t) \\
  &= \{ \text{definition of meaning function} \} ap >x> W
  \end{align*}
  \]
2.2.5 Distributivity

Lemma 4 Let $V_0 \subseteq V_1 \cdots$. Then, $(\cup i :: p >x> V_i) = p >x> (\cup i :: V_i)$, for any $p$.

Proof:
A note on notation: we use subscripts $i$, $j$, $k$ and $n$ (in addition to a 0 and 1) to designate sets such as $V_0$. At the same time, we will write $V_t$, for time-shift of $V$. And, they will be combined a few times as in $(V_i)_t$; we use context to differentiate the two usages.

Proof is by mutual inclusion.

- $(\cup i :: p >x> V_i) \subseteq p >x> (\cup i :: V_i)$: For any $i$,
  
  $p >x> V_i 
  \subseteq \{ V_i \subseteq (\cup i :: V_i); >x> \text{ is monotonic from Theorem 9, page 37}\}$
  $p >x> V$

Therefore, $(\cup i :: p >x> V_i) \subseteq p >x> (\cup i :: V_i)$.

- $p >x> (\cup i :: V_i) \subseteq (\cup i :: p >x> V_i)$:
  
  If $(\cup i :: V_i) = \phi$ then, $V_i = \phi$, for all $i$. So, $p >x> (\cup i :: V_i) = \phi = (\cup i :: p >x> V_i)$. Assume, henceforth, that $(\cup i :: V_i) \neq \phi$. Then, there is some $V_j$, $V_j \neq \phi$. The proof is by induction on the length of $p$.

  If $p = \epsilon$, then $p >x> (\cup i :: V_i) = \{ \epsilon \}$ and $p >x> V_j = \{ \epsilon \} \subseteq (\cup i :: p >x> V_i)$.

  Next, we show that $ap >x> (\cup i :: V_i) \subseteq (\cup i :: ap >x> V_i)$, for any $ap$. We consider three cases, depending on $a$.

  Case 1) $c_1(a)$:

  \[
  ap >x> (\cup i :: V_i) = \{ c_1(a) \}
  a(p >x> (\cup i :: V_i)) \subseteq \{ \text{induction on } p >x> (\cup i :: V_i) \}
  a(\cup i :: p >x> V_i) = \{ \text{concatenation distributes over set union} \}
  (\cup i :: a(p >x> V_i)) = \{ c_1(a) \}
  (\cup i :: ap >x> V_i)
  \]

  Case 2) $c_2(a)$:

  \[
  ap >x> (\cup i :: V_i) = \{ c_2(a) \}
  a(p >x> (\cup i :: V_i)) = \{ \text{removal operator distributes over set union} \}
  a(p >x> (\cup i :: V_i)) \subseteq \{ \text{induction on } p >x> (\cup i :: V_i) \}
  \]


\[ a(\forall i :: p > x > V'_i) \]
\[ = \{ \text{concatenation distributes over set union} \} \]
\[ (\forall i :: a(p > x > V'_i)) \]
\[ = \{ \text{removal operator distributes over set union} \} \]
\[ (\forall i :: ap > x > V_i) \]

Case 3) \( c_3(a) \):

\[ ap > x > (\forall i :: V_i) \]
\[ = \{ c_3(a) \} \]
\[ a(p > x > (\forall i :: V_i) | ((\forall i :: V_i')_k) \]
\[ = \{ \text{time-shift distributes over set union} \} \]
\[ a(p > x > (\forall i :: V_i) | (\forall i :: (V''_i)_k)) \]
\[ \subseteq \{ \text{induction on } p > x > (\forall i :: V_i); \} \]
\[ \text{merge and concatenation are monotonic wrt set union} \]
\[ a(\forall i :: p > x > V_i) | (\forall i :: (V''_i)_k) \]

Now it is sufficient to show that for any \( q \in (\forall i :: p > x > V_i) \) and \( r \in (\forall i :: (V''_i)_k) \), \( a(q | r) \subseteq (\forall i :: ap > x > V_i) \).

Since \( q \in (\forall i :: p > x > V_i) \), for some \( j, q \in p > x > V_j \).
Since \( r \in (\forall i :: (V''_i)_k) \), for some \( k, r \in (V''_i)_k \).
Let \( n = \max(j, k) \). Then \( V_j \subseteq V_n \) and \( V_k \subseteq V_n \).

\[ q \in p > x > V_j \]
\[ \Rightarrow \{ V_j \subseteq V_n \text{ and } > x > \text{ is monotonic from Theorem 9, page 37} \} \]
\[ q \in p > x > V_n \]

Similarly, from \( r \in (V''_k)_k \) and \( V_k \subseteq V_n \), we get \( r \in (V''_n)_k \). Then,

\[ a(q | r) \]
\[ \subseteq \{ q \in p > x > V_n; \} \]
\[ \text{merge and concatenation are monotonic wrt set union} \]
\[ a(p > x > V_n | r) \]
\[ \subseteq \{ r \in (V''_n)_k \} \]
\[ a(p > x > V_n | (V''_n)_k) \]
\[ = \{ c_3(a) \} \]
\[ ap > x > V_n \]
\[ \subseteq \{ \text{set theory} \} \]
\[ (\forall i :: ap > x > V_i) \]

Next, we establish that every Orc combinator distributes over set union in its left argument and in its right argument under a certain condition. Below \( * \) is any Orc combinator: \( |, > x > \text{ or } < x < \).

**Theorem 10** For any \( U \) and \( V \),

1. (Left Distributivity) \( (\forall i :: P_i * V) = (\forall i :: P_i) * V \), for a family of sets \( P_i \).
2. (Right Distributivity) \((\cup i :: U \ast Q_i) = U \ast (\cup i :: Q_i)\), for a sequence of sets \(Q_i\), where \(Q_0 \subseteq Q_1 \subseteq \cdots\).

Proof: Left Distributivity follows from the definitions of the combinators over sets.

\[
(\cup i :: P_i \ast V) = \{\text{expanding } P_i \ast V\}
(\cup i :: (\cup p : p \in P_i : p \ast V)) = \{\text{set theory}\}
(\cup p : p \in (\cup i :: P_i) \ast V) = \{\text{definition of coercion}\}
(\cup i :: P_i) \ast V
\]

Right distributivity for \(\cup\) and \(<x<\) follow similarly, because they are coercive in both arguments. Now, we show that \((\cup i :: U > x > Q_i) = U > x > (\cup i :: Q_i)\), for a sequence of sets \(Q_i\), where \(Q_0 \subseteq Q_1 \subseteq \cdots\).

\[
(\cup i :: U > x > Q_i) = \{\text{expanding } U > x > Q_i\}
(\cup i :: (\cup p : p \in U : p > x > Q_i)) = \{\text{set theory}\}
(\cup p : p \in U : (\cup i :: p > x > Q_i)) = \{\text{definition of coercion}\}
U > x > (\cup i :: Q_i)
\]

We note that right distributivity holds for the combinators \(\cup\) and \(<x<\) for arbitrary sets \(Q_i\)s; the additional condition \(Q_0 \subseteq Q_1 \subseteq \cdots\) is not required. This condition is needed only for \(> x >\), as we outline below.

We show that \(U > x > (V \cup W) \neq U > x > V \cup U > x > W\), in general; in fact, \(ap > x > (V \cup W) \neq (ap > x > V) \cup (ap > x > W)\) when \(c_3(a)\) holds. Let \(V_1 = p > x > V\) and \(V_2 = (V''')t\), and \(W_1\) and \(W_2\) are similarly defined.

\[
ap > x > (V \cup W)
= \{c_3(a)\}
ap > x > (V \cup W) \cup ((V \cup W)''')t
\geq \{p > x > (V \cup W) \geq p > x > V \cup p > x > W\}
ap > x > V \cup p > x > W \cup ((V \cup W)''')t
= \{V_1 = p > x > V\text{ and } V_2 = (V''')t; \text{ similarly for } W_1\text{ and } W_2\}
ap(V_1 \cup W_1) \cup (V_2 \cup W_2)
= \{\text{coercion}\}
ap(V_1 | V_2) \cup (V_1 | W_2) \cup (W_1 | V_2) \cup (W_1 | W_2)
\]

And,
\[ (ap > x > V) \cup (ap > x > W) \]
\[ = \{c_3(a)\} \]
\[ a(p > x > V \mid (V''_t)) \cup a(p > x > W \mid (W''_t)) \]
\[ = \{V_1 = p > x > V \text{ and } V_2 = (V''_t); \text{ similarly for } W_1 \text{ and } W_2\} \]
\[ a(V_1 \mid V_2) \cup a(W_1 \mid W_2) \]
\[ = \text{rewriting} \]
\[ a((V_1 \mid V_2) \cup (W_1 \mid W_2)) \]

Thus, in general, \( ap > x > (V \cup W) \supseteq (ap > x > V) \cup (ap > x > W) \).

### 2.3 Characterization of Symmetric Composition

The goal of this section is to show that \([ f \mid g ] = [ f ] \mid [ g ]\). We prove this result in two parts: \([ f \mid g ] \subseteq [ f ] \mid [ g ]\) and \([ f ] \mid [ g ] \subseteq [ f \mid g ]\). For the proof we employ the operational semantics of \(\mid\), from Section 1.2, page 2, and the meaning function given in Section 2.2.1, page 35.

We note some preliminary facts.

1. \( a \simeq b \equiv b \simeq a \).
2. \( a \simeq b \Rightarrow \neg (a \preceq b), a \simeq b \Rightarrow \neg (b \preceq a) \).
3. It is possible for \( a \simeq b, a \preceq b \) and \( b \preceq a \) to be all false at the same time.

**Lemma 5** \((u \mid v)_t = u_t \mid v_t\).

Proof: Apply the definition of \(\mid\) to both sides. Note that \(a_t \simeq b_t \equiv a \simeq b, a_t \preceq b_t \equiv a \preceq b\). The result follows by applying induction.

**2.3.1** \([ f \mid g ] \subseteq [ f ] \mid [ g ]\)

**Theorem 11** \([ f \mid g ] \subseteq [ f ] \mid [ g ]\)

Proof: Given \( p \in [ f \mid g ]\), we show that \( p \in [ f ] \mid [ g ]\). Proof is by induction on the length of \( p \).

- **p = \epsilon:** then, \( p \in \{ \epsilon \} \mid \{ \epsilon \} \subseteq [ f ] \mid [ g ]\), since \( \{ \epsilon \} \subseteq [ f ]\), and \( \{ \epsilon \} \subseteq [ g ]\).
- **p = aq_t,** where \( a \) is base: Given \( p \in [ f \mid g ]\), without loss in generality, assume that \( f \xrightarrow{a} f' \) and \( f' \mid g_t \xrightarrow{y_t} \).

Case 1) \( q = \epsilon\): Since \( g_t \neq \bot\), there is some \( y \in [ t' ]\), that is, \( y_t \in [ g ]\).

\[ \in \{ q = \epsilon \} \]
\[ a(\epsilon \mid y_t) \]
\[ \subseteq \{ a \text{ is a base event at time } t; \text{ use definition of } \mid \} \]
\[a \in [y_t] = \{a = a\}\]
\[a \in [y_t] \subseteq \{\text{from } f \xrightarrow{a} , a \in \llbracket f \rrbracket; \text{ we have } y_t \in \llbracket g \rrbracket.\}\]

Case 2) \(q \neq \epsilon\): From \(f' \mid g' \xrightarrow{q} \), inductively, \(q \in x \mid y\), where \(x \in \llbracket f' \rrbracket\) and \(y \in \llbracket g' \rrbracket\). Since \(q \neq \epsilon\), we get \(y \neq \epsilon\).

\[aq_t \in \{q \in x \mid y\}\]
\[\Rightarrow \text{ {distribute time-shift} }\]
\[aq_t \in \{a(x_t \mid y_t)\}\]
\[\Rightarrow \text{ {a.time} } = t \leq (y_t).time, \text{ and } a \text{ is base}\]
\[a(x_t \mid y_t) \subseteq \{f \xrightarrow{a} f' \xrightarrow{\delta} , a.time = t; \text{ so } f \xrightarrow{a} \}, \text{ or } ax_t \in \llbracket f \rrbracket\]
\[y \in \llbracket g' \rrbracket, \text{ so, } y_t \in \llbracket g \rrbracket.\}\]

\[\llbracket f \rrbracket | \llbracket g \rrbracket \subseteq \llbracket f | g \rrbracket\]

**Theorem 12** \([f] | [g] \subseteq \llbracket f | g \rrbracket\)

**Proof:** Let \(p \in \llbracket f \rrbracket | \llbracket g \rrbracket\); we show that \(p \in \llbracket f | g \rrbracket\). Proof is by induction on the length of \(p\).

- \(p = \epsilon\): The execution set of every expression, hence \([f | g]\), contains \(\epsilon\).
- \(p = aq_t\), where \(a\) is a base event:
  \[aq_t \in \llbracket f \rrbracket | \llbracket g \rrbracket\]
  \[\Rightarrow \text{ {assume that } a \text{ is an event from } \llbracket f \rrbracket; definition of | }\]
The proof is similar for the first two cases in the definition (i.e., when \(a \cdot \epsilon \varepsilon V\)).

**Lemma 6**

For any \(p \cdot V\), \(p > x > V = (p > x > V)_t\), where \(a \cdot V\)

\[
aq \in au | v_t, \text{ where } au \in [f], \ v_t \in [g] \\
\Rightarrow \{a \text{ is a base event; so, } au | v_t = a(u_t | v_t)\} \\
aq \in a(u_t | v_t), \text{ where } au \in [f], \ v_t \in [g] \\
\Rightarrow \{u_t | v_t = (u | v)_t\} \\
q \in u | v, \text{ and } au \in [f], \ v_t \in [g] \\
\Rightarrow \{au \in [f] \text{ means } f \xrightarrow{a} f', \text{ for some } f', \\
v_t \in [g] \text{ means } g' \xrightarrow{v} \} \\
q \in [f'] | [g'] \\
\Rightarrow \{\text{induction}\} \\
q \in [f' | g'] \\
\Rightarrow \{f \xrightarrow{a} f' \text{ implies } f | g \xrightarrow{a} f' | g', \text{ and } f' | g' \xrightarrow{g}\} \\
aq \in [f | g] \\
\bullet p = aq_t, \text{ where } a \text{ is a substitution event:}

\[
aq \in [f] | [g] \\
\Rightarrow \{a \text{ is a substitution event, use definition of merge}\} \\
au \in [f], \ av \in [g], \text{ where } q \in u | v \\
\Rightarrow \{\text{rewriting}\} \\
f \xrightarrow{a} f', \text{ for some } f', \text{ and } g \xrightarrow{a} g', \text{ for some } g' \\
\Rightarrow \{q \in u | v\} \\
q \in [f'] | [g'] \\
\Rightarrow \{f \xrightarrow{a} f' \text{ and } g \xrightarrow{a} g' \text{ implies } f | g \xrightarrow{a} f' | g'\} \\
f | g \xrightarrow{a} f' | g' \xrightarrow{g} \\
\Rightarrow \{\text{rewriting}\} \\
aq_t \in [f | g] \\
\]

### 2.4 Characterization of Sequential Composition

The goal of this section is to show that \([f > x > g] = [f] > x > [g]\). We prove this result in two parts: 

\([f > x > g] \subseteq [f] > x > [g]\) and \([f > x > g] \subseteq [f] > x > [g]\). For the proof we employ the operational semantics of \(> x >\), from Section 1.2, page 2, and the meaning function given in Section 2.2.2, page 36.

#### 2.4.1 Preliminary Results

**Lemma 6** For any \(p \cdot V\), \(p > x > V = (p > x > V)_t\)

Proof: If \(V = \phi\), the result is trivial. Assume \(V \neq \phi\). Proof is by induction on the length of \(p\).

- \(\epsilon_t > x > V = (\epsilon > x > V)_t\):
- \(\epsilon_t > x > V = \epsilon > x > V = \{\epsilon\} \text{ and } (\epsilon > x > V)_t = \{\epsilon\}_t = \{\epsilon\}.

- \((ap)_t > x > V = (ap > x > V)_t\):
  
  The proof is similar for the first two cases in the definition (i.e., when \(a\) is not
a publication). So, we show just one proof, for the first case in the definition
where \( c_1(a) \) holds.

\[
(a) \quad \text{for } x > V \implies V = \{ \text{definition; note that } c_1(a) \text{ holds, so } c_1(a_t) \text{ holds} \}
\]

\[
(a_t) \quad \text{for } x > V \implies V = \{ \text{induction: } p_t > x > V = (p > x > V)_t \}
\]

(2) \quad \text{for } x > V \implies V = \{ \text{time-shift of } t \text{ distributes over concatenation} \}

\[
(a(p > x > V))_t = \{ \text{definition: } ap > x > V = a(p > x > V), \text{ given } c_1(a) \}
\]

\[
(ap > x > V)_{t_s} = \{ \text{definition, } ap > x > V = (s, \tau)(p > x > V | V'_s) \}
\]

**Observation 10** For \( y \neq x \), \( f > x > g \overset{(t, [m/y])}{\rightarrow} f' > x > g' \), where \( f \overset{(t, [m/y])}{\rightarrow} f' \) and \( g \overset{(0, [m/y])}{\rightarrow} g' \). Proof:

\[
f > x > g
\]

\[
\overset{(t, [m/y])}{\rightarrow} \{ \text{definition of substitution} \}
\]

\[
[m/y].(f > x > g)\overset{t}{\rightarrow}
\]

\[
= \{ \text{definition of } (f > x > g)^t \}
\]

\[
[m/y].(f^t > x > g)
\]

\[
= \{ \text{substitution rules, note that } y \neq x \}
\]

\[
[m/y].(f^t) > x > [m/y].g
\]

\[
= \{ f \overset{(t, [m/y])}{\rightarrow} \}
\]

\[
[m/y].(f^t) = f', \quad g \overset{(0, [m/y])}{\rightarrow} [m/y].g = g'
\]

\[
f' > x > g'
\]
CHAPTER 2. COMBINATORS APPLIED TO EXECUTIONS

Observation 11 \( f > x > g \xrightarrow{(t,[m/x])} f' > x > g \), where \( f \xrightarrow{(t,[m/z])} f' \).

Proof:

\[
\begin{align*}
& f > x > g \\
& \xrightarrow{(t,[m/c])} \text{definition of substitution} \\
& \quad [m/x].(f > x > g)^t \\
& \quad \{ \text{definition of } (f > x > g)^t \} \\
& \quad [m/x].(f^t > x > g) \\
& \quad \{ \text{substitution rules} \} \\
& \quad [m/x].(f^t) > x > g \\
& \quad \{ f \xrightarrow{(t,[m/c])} [m/x].(f^t) = f' \} \\
& f' > x > g
\end{align*}
\]

2.4.2 \([ f > x > g ] \subseteq [ f ] > x > [ g ]\)

Theorem 13 \([ f > x > g ] \subseteq [ f ] > x > [ g ]\)

Proof: Given \( p \in [ f > x > g ] \), we show that \( p \in [ f ] > x > [ g ] \). Proof is by induction on the length of \( p \).

- \( p = \epsilon \): We have \( \epsilon \in [ f ] \) and \( \{ \epsilon \} \subseteq [ g ] \). Therefore, \( \{ \epsilon \} = \epsilon > x > \{ \epsilon \} \subseteq [ f ] > x > [ g ] \).

- \( p = aq \), where \( a \) is an other-substitution:

\[
\begin{align*}
& aq \in [ f > x > g ] \\
& \Rightarrow \{ \text{operational semantics of } [ f > x > g ] \} \\
& \Rightarrow \{ \text{from Observation 10, page 45, } f \xrightarrow{a} f', g \xrightarrow{(0,b)} g' \text{, where } a = (t,b) \} \\
& \Rightarrow \{ f \xrightarrow{a} f', g \xrightarrow{(0,b)} g' \text{, where } a = (t,b) \text{ and } f' > x > g' \} \\
& \Rightarrow \{ \text{induction on } q \in [ f' > x > g' ] \} \\
& \Rightarrow \{ q \in [ f' ] > x > [ g' ] \} \\
& \Rightarrow \{ \text{rewriting} \} \\
& \Rightarrow \{ aq \in a([ f' ] > x > [ g' ]). \}_t \\
& \Rightarrow \{ \text{sublemma below} \} \\
& \Rightarrow \{ f \xrightarrow{a} f' \}; \text{from Observation 6, page 8, } a[ f' ]_t \subseteq [ f ] \} \\
& \Rightarrow \{ f \xrightarrow{a} f'; \text{from Observation 6, page 8, } a[ f' ]_t \subseteq [ f ] \} \\
& \Rightarrow \{ aq \in [ f ] > x > [ g ] \}
\end{align*}
\]

**Sublemma:** We show that \( a([ f' ] > x > [ g' ]). \) = \( (a[ f' ])_t > x > [ g ] \), given \( g \xrightarrow{(0,b)} g' \), and \( a = (t,b) \).

\[
\begin{align*}
& (a[ f' ])_t > x > [ g ] \\
& \Rightarrow \{ \text{from definition (SCD2), since } c_2(a) \}
\end{align*}
\]
\[ a([f']_t > x > ([g] \setminus (0, b))) \]

\[ = \{ g \stackrel{(0, b)}{\rightarrow} g' \}. \text{ So, } [g] \setminus (0, b) = [g'] \] from Observation 6, page 8
\[ a([f']_t > x > [g']) \]
\[ \Rightarrow \{ \text{from Lemma 6, page 44, } [f']_t > x > [g'] = ([f'] > x > [g'])_t \} \]
\[ = \{ g \stackrel{(0, b)}{\rightarrow} g'. \text{ So, } [g] \setminus (0, b) = [g'] \]
\[ a([f'] > x > [g'])_t \]

- \( p = aq_t \), where \( a \) is an own-substitution:

\[ aq_t \in [f > x > g] \]
\[ \Rightarrow \{ \text{from Observation 11, page 46} \}
\[ f > x > g \stackrel{a}{\rightarrow} f' > x > g \stackrel{\stackrel{a}{\rightarrow}}{\Rightarrow} , \text{ where } f \stackrel{a}{\rightarrow} f' \]
\[ \Rightarrow \{ \text{induction on } q \in [f'] > x > [g] \} \]
\[ q \in [f'] > x > [g] \]
\[ \Rightarrow \{ \text{rewriting} \}
\[ aq_t \in a([f'] > x > [g])_t \]
\[ \Rightarrow \{ \text{from Lemma 6, page 44, } ([f'] > x > [g])_t = [f']_t > x > [g] \}
\[ aq_t \in a([f']_t > x > [g]) \]
\[ = \{ \text{from definition (SCD1), since } c_1(a) \}
\[ aq_t \in (a[f']_t)_{> x} [g] \]
\[ \Rightarrow \{ f \stackrel{a}{\rightarrow} f'; \text{ so, } a[f']_t \subseteq [f] \}
\[ aq_t \in [f] > x > [g] \]

- \( p = aq_t \), where \( a \neq \tau \) and \( a \) is not a substitution:

Since \( aq_t \in [f > x > g] \), \( a \) can not be a publication. Hence, \( c_1(a) \) holds. Also, \( a \neq \tau \) means

\[ f \stackrel{a}{\rightarrow} f', \text{ and } f > x > g \stackrel{a}{\rightarrow} f' > x > g \stackrel{\stackrel{a}{\rightarrow}}{\Rightarrow} \]

The proof is similar to the case where \( a \) is an own-substitution.

- \( p = aq_t \), where \( a = (t, \tau) \):

If \( a \) is not due to a publication, that is, \( f \stackrel{a}{\rightarrow} f' \) so that \( f > x > g \stackrel{a}{\rightarrow} f' > x > g \), the proof is similar to the case where \( a \) is an own-substitution.

If \( a \) is due to a publication \((t, !_m)\), i.e., \( f \stackrel{(t, !_m)}{\rightarrow} f' \),

\[ f \stackrel{(t, !_m)}{\rightarrow} f' \]
\[ \Rightarrow \{ \text{operational semantics} \}
\[ f > x > g \stackrel{(t, \tau)}{\rightarrow} f' > x > g \ | [m/x]; g \]
\[ \Rightarrow \{ p = aq_t \in [f > x > g] \text{ and } a = (t, \tau) \}
\[ q \in [f'] > x > g \ | [m/x]; g \]
\[ \Rightarrow \{ \text{from theorems Theorem 11, page 42 and Theorem 12, page 43:} \]
\[ [f'] > x > g \ | [m/x]; g = [f'] > x > g \ | [m/x]; g \]
\( q \in [f' \triangleright x > g] \mid [[m/x].g] \)

\( \Rightarrow \) {rewriting}

\( q \in u \mid v, \) where \( u \in [f' \triangleright x > g], \) and \( v \in [[m/x].g] \)

\( \Rightarrow \) {induction on \( u \in [f' \triangleright x > g] \)}

\( u \in [f' \triangleright x > g], q \in u \mid v, v \in [[m/x].g] \)

\( \Rightarrow \) {\( q \in u \mid v. \) So \( \mathcal{q}_t \in (u \mid v)_t = u_t \mid v_t \)}

\( aq_t \in a(u_t \mid v_t), v \in [[m/x].g] \)

\( \Rightarrow \) {from Observation 6, page 8, \([m/x].g = [g] \setminus (0, [m/x])\)}

\( aq_t \in a(u_t \mid v_t), v \in ([g] \setminus (0, [m/x])) \)

\( \Rightarrow \) {\( a = (t, \tau), u \in [f' \triangleright x > g] \)}

\( aq_t \in (t, \tau)(([f'] \triangleright x > [g]) \setminus (m/x)) \}

\( \Rightarrow \) {from Lemma 6, page 44, \(([f'] \triangleright x > [g]) \setminus (m/x)) = ([f'] \triangleright x > [g]) \}

\( \Rightarrow \) {from definition (SCD3) given \( c_3(a) \)}

\( aq_t \in ((t, \! m)([f'] \setminus t) \triangleright x > [g]) \}

\( \Rightarrow \) {from \( f \xrightarrow{(t, \! m)} f', (t, \! m)\{[f'] \setminus t) \subseteq [f] \)}

\( aq_t \in [f] \triangleright x > [g] \)

2.4.3 \([f] \triangleright x > [g] \subseteq [f \triangleright x > g] \)

**Theorem 14** \([f] \triangleright x > [g] \subseteq [f \triangleright x > g] \)

**Proof:** For \( p \in [f] \triangleright x > [g], \) we show \( p \in [f \triangleright x > g]. \) If \( p = \epsilon, \) the result follows from \( \epsilon \in [f \triangleright x > g]. \)

Let \( p = aq_t. \) From definition (SCD1–SCD3), \( a \) can not be a publication. We consider four cases: (1) \( a \) is an own-substitution, (2) \( a \) is an other-substitution, and (3) \( a \) is not a substitution and \( a \neq (t, \tau), \) for any \( t, \) and (4) \( a = (t, \tau), \) for some \( t. \)

**Case 1:** \( a \) is an own-substitution: So, \( c_1(a) \) holds.

\( aq_t \in [f] \triangleright x > [g] \)

\( \Rightarrow \) {from definition (SCD1), since \( c_1(a) \)}

\( aq_t \in a(r_t \triangleright x > [g]) \), where \( ar_t \in [f] \)

\( \Rightarrow \) {simplify and rewrite}

\( q_t \in r_t \triangleright x > [g], \) and \( f \xrightarrow{a} f' \xrightarrow{r} \)

\( \Rightarrow \) {from Lemma 6, page 44, \( r_t \triangleright x > [g] = (r \triangleright x > [g])_t \)}

\( q_t \in (r \triangleright x > [g])_t, \) and \( f' \xrightarrow{r} \)

\( \Rightarrow \) {simplify}

\( q \in r \triangleright x > [g], \) and \( f' \xrightarrow{r} \)

\( \Rightarrow \) {\( r \in [f'] \)}

\( q \in ([f'] \triangleright x > [g]) \)

\( \Rightarrow \) {induction}

\( q \in [f' \triangleright x > g] \)

\( \Rightarrow \) {from operational semantics and Observation 11, page 46}

\( f \xrightarrow{\cdot} f' \) and \( c_1(a) \) implies \( f \triangleright x > g \xrightarrow{a} f' \triangleright x > g \)
CHAPTER 2. COMBINATORS APPLIED TO EXECUTIONS

\[ f \xrightarrow{a} f' \xrightarrow{a} f'' \xrightarrow{a} g \]

Case 2) \( a \) is an other-substitution: So, \( c_2(a) \) holds. Let \( a = (t, b) \) and \( a_0 = (0, b) \).

\[ aq_t \in [f] \xrightarrow{a} g \]

\[ \Rightarrow \text{from definition (SCD2), since } c_2(a) \]

\[ aq_t \in a(\tau_i \xrightarrow{a} ([g] \setminus a_0)), \text{ where } a \tau_i \in [f] \]

\[ \Rightarrow \text{Let } g \xrightarrow{a} g'; \text{ then, from Observation 6, page 8, } ([g] \setminus a_0) = [g'] \]

\[ aq_t \in a(\tau_i \xrightarrow{a} [g']), \text{ where } a \tau_i \in [f] \text{ and } g \xrightarrow{a} g' \]

\[ \Rightarrow \text{simplify and rewrite} \]

\[ q_t \in \tau_i \xrightarrow{a} [g'], f \xrightarrow{a} f' \xrightarrow{r} \text{ and } g \xrightarrow{a} g' \]

\[ \Rightarrow \{r \in [f']\} \]

\[ q \in [f'] \xrightarrow{a} [g'], f \xrightarrow{a} f' \xrightarrow{r} \text{ and } g \xrightarrow{a} g' \]

\[ \Rightarrow \text{from Observation 10, page 45, } f \xrightarrow{a} f' \xrightarrow{r} g' \]

\[ \Rightarrow \text{rewrite} \]

\[ aq_t \in [f] \xrightarrow{a} g \]

Case 3) \( a \) is not a substitution and \( a \neq (t, \tau) \), for any \( t \): So, \( c_1(a) \) holds and the proof is similar to that for Case (1).

Case 4) \( a = (t, \tau) \), for some \( t \): Then, the third case in the definition, (SCD3), applies.

\[ aq_t \in [f] \xrightarrow{a} [g] \]

\[ \Rightarrow \{a = (t, \tau)\} \]

\[ \{t, \tau\} q_t \in [f] \xrightarrow{a} [g] \]

\[ \Rightarrow \text{from definition (SCD3)} \]

\[ \{t, \tau\} q_t \in (t, \tau)(\tau_i \xrightarrow{a} [g] \mid ([g] \setminus (0, [m/x]))), \text{ where} \]

\[ f \xrightarrow{(t, \tau_m)} f' \xrightarrow{r} \text{ for some } m \]

\[ \Rightarrow \text{let } g \xrightarrow{(0, [m/x])} g'' \text{; from Observation 6, page 8, } ([g] \setminus (0, [m/x])) = [g''] \}

\[ q_t \in (\tau_i \xrightarrow{a} [g] \mid [g''] \mid \tau), f \xrightarrow{(t, \tau_m)} f' \xrightarrow{r} \text{ and } g \xrightarrow{(0, [m/x])} g'' \]

\[ \Rightarrow \text{from Lemma 6, page 44, } \tau_i \xrightarrow{a} [g] = (\tau \xrightarrow{a} [g]) \}; \text{ simplify} \}

\[ q \in (\tau \xrightarrow{a} [g] \mid [g'']), f \xrightarrow{(t, \tau_m)} f' \xrightarrow{r} \text{ and } g \xrightarrow{(0, [m/x])} g'' \]

\[ \Rightarrow \{r \in [f']\} \]

\[ q \in [f'] \xrightarrow{a} [g] \mid [g''], f \xrightarrow{(t, \tau_m)} f' \xrightarrow{r} \text{ and } g \xrightarrow{(0, [m/x])} g'' \]

\[ \Rightarrow \text{definition of symmetric composition} \]

\[ q \in u \mid v, \text{ where } u \in [f'] \xrightarrow{a} [g] \text{ and } v \in [g''] \]

\[ \Rightarrow \text{induction on } u \in [f'] \xrightarrow{a} [g] \} \]
CHAPTER 2. COMBINATORS APPLIED TO EXECUTIONS

50

$q \in u \mid v$, where $u \in [f' > x > g]$ and $v \in [g'']$

$\Rightarrow$  {rewrite}

$q \in [f' > x > g] \mid [g'']$

$\Rightarrow$  {theorems on Merge: $[f' > x > g] \mid [g''] = [f' > x > g \mid g'']$}

$q \in [f' > x > g \mid g'']$

$\Rightarrow$  {given $f \xrightarrow{(t, m)} f'$}

$f > x > g \xrightarrow{(t, \tau)} f' > x > g \mid [m/x].g$

$\Rightarrow$  {[$m/x].g = g''$}

$f > x > g \xrightarrow{(t, \tau)} f' > x > g \mid g''$

$\Rightarrow$  {rewrite}

$q \in [f' > x > g \mid g'']$

$\Rightarrow$  {rewrite}

$(t, \tau)q \in [f > x > g]$

2.5 Characterization of Asymmetric Composition

The goal of this section is to show that $[f < x < g] = [f] < x < [g]$. We prove this result in two parts: $[f < x < g] \subseteq [f] < x < [g]$ and $[f < x < g] \subseteq [f] < x < [g]$. For the proof we employ the operational semantics of $< x <$, from Section 1.2, page 2, and the meaning function given in Section 2.2.3, page 36.

2.5.1 Preliminary Results

We often write $ap_t$ to mean that $a$ is an event at time $t$, (and $p_t$ is the execution following $a$). It is possible for $p$ to be $\epsilon$.

Observation 12 $f < x < g \xrightarrow{p} \implies f < x < g \xrightarrow{p} c$, where $c$ is an other-substitution, and $c.time = p.time$.

Proof: From the operational semantics.

Observation 13 $(ap_t).time = t + p.time$

Proof: For $p = \epsilon$, $(ap_t).time = t = t + p.time$.

$p \neq \epsilon$, $(ap_t).time = t + p.time$.

Observation 14 $(ap_t).time - (aq_t).time = p.time - q.time$.

$(ap_t).time - (q_t).time = p.time - q.time$, provided $q \neq \epsilon$.

Proof: The first part follows directly from Observation 13, page 50. The second part follows similarly, by noting that $(q_t).time = t + q.time$, for $q \neq \epsilon$.

Observation 15 Let
Then, $T - u.time = T' - u'.time$ and $T - v.time = T' - v'.time$.

Proof: The two proofs are similar; we prove only the first one.

\[
T = \max(u.time, v.time).
\]

\[
T = \max(u.time + u'.time, v.time + v'.time).
\]

\[
T' = \max(u.time + v.time).
\]

Therefore, $T - u.time = T' - u'.time$.

**Observation 16** \( \epsilon \in u +_\times v \iff u = \epsilon \land v = \epsilon \)

Proof: \( \epsilon \in u +_\times v \) by application of the base rule only, because the inductive rule creates items starting with an item. The result follows by considering the base rule.

**Lemma 7** \( p \in u +_\times v \Rightarrow d_0(u, v) \land p.time = \max(u.time, v.time) \).

Proof: Proof is by induction on the combined lengths of \( u \) and \( v \).

1. \( u = \epsilon \) and \( v = \epsilon \): then, \( d_0(u, v) \). And, \( p = \epsilon \), so, \( p.time = 0 = \max(u.time, v.time) \).
2. \( u = \epsilon \) and \( v \neq \epsilon \). Since \( u +_\times v \neq \phi \), \( v \) contains no other-substitution. So, \( d_0(u, v) \). And, \( p = v \), so, \( p.time = v.time = \max(u.time, v.time) \).
3. If \( u \neq \epsilon \) and \( v = \epsilon \): Since \( u +_\times v = \phi \), \( u \) contains no substitution. So, \( d_0(u, v) \).
4. If \( u \neq \epsilon \) and \( v \neq \epsilon \): We rename the terms and consider \( cp \in au +_\times bv \). We will show that \( d_0(au, bv) \) and \( (cp).time = \max(au.time, bv.time) \).

Case 1) \( a \approx_b b, a = b = c \) and \( p \in u +_\times v \). Inductively, \( d_0(u, v) \); so, \( d_0(au, bv) \).

Let \( a.time = b.time = c.time = t \). And, \( u = u', v = v' \), \( p = p' \). From \( p \in u +_\times v \), we get \( p'_i \in u'_i +_\times v'_i \), or \( p'_i \in (u' +_\times v')c_i \), or \( p' \in u' +_\times v' \).

\[
\max(au.time, bv.time) = \max((au'_i).time, (bv'_i).time) = \{a.time = t, b.time = t; \text{from Observation 13, page 50}\}.
\]
\begin{align*}
\max(t + u'.time, t + v'.time) \\
= \{\text{arithmetic}\} \\
= t + \max(u'.time, v'.time) \\
= \{\text{from } p' \in u' + v', \text{ inductively, } p'.time = \max(u'.time, v'.time)\} \\
= t + p'.time \\
= \{a.time = t, \text{ from Observation 13, page } 50\} \\
= (ap').time \\
= \{p = p', a = c\} \\
= (ep).time
\end{align*}

Case 2) \(a \preceq b, \ c = a\) and \(p \in u + bv\): Inductively, \(d_0(u, bv)\); so, \(d_0(au, bv)\), because \(a\) is base, from \(a \preceq b\). Let \(a.time = c.time = t\). And, \(u = u', bv = (b'v')_t, p = p'\). From \(p \in u + bv\), we get \(p'_t \in u' + (b'v')_t, \) or \(p'_t \in (u' + b'v')_t, \) or \(p' \in u' + b'v'.\)

\begin{align*}
\max(au.time, bv.time) \\
= \{u = u', bv = (b'v')_t\} \\
= \max((au').time, (b'v')_t.time) \\
= \{a.time = t, \text{ from Observation 13, page } 50\} \\
= \{\text{arithmetic}\} \\
= t + \max(u'.time, (b'v').time) \\
= \{\text{from } p' \in u' + b'v', \text{ inductively, } p'.time = \max(u'.time, (b'v').time)\} \\
= t + p'.time \\
= \{a.time = t, \text{ from Observation 13, page } 50\} \\
= (ap').time \\
= \{p = p', a = c\} \\
= (ep).time
\end{align*}

Case 3) \(b \preceq a, \ c = b\) and \(p \in au + v\): Similar to Case (2).

The prefix-closure of \(u\), written as \(u^*\), is the set of prefixes of \(u\). Formally,

\[
\epsilon^* = \{\epsilon\}, \\
(au)^* = \{\epsilon\} \cup au^*
\]

Note that \(\{\epsilon\} \subseteq u^*\), for all \(u\). Therefore, \((au)^* = \{\epsilon\} \cup au^*\) holds (vacuously) even when \(a = \epsilon\). Set \(U\) is prefix-closed if \(u^* \subseteq U\), for every \(u\) in \(U\).

**Lemma 8** \(u, v\) is prefix-closed.

Proof: We first observe that \(\epsilon \in (u, v)\), for any \(u\) and \(v\). If either \(u\) or \(v\) is empty, the result follows from definition. For \(au\), \(bv\), \(\neg((a \simeq b) \land (a \preceq b))\); so, at least one of these conditions is false, and the corresponding guarded set contributes \(\{\epsilon\}\).

The proof of prefix-closure is by induction on the combined lengths of \(u\) and \(v\). For empty \(u\) or empty \(v\), the result is obvious.
CHAPTER 2. COMBINATORS APPLIED TO EXECUTIONS

53

Case 2.1) \( a \approx b \rightarrow a(u|_v v) \cup [a \leq b \rightarrow a(u|_v b v)] \cup [b \leq a \rightarrow b(a(u|_v v))]^* \)

= \{prefix-closure distributes over set union and guarded sets\}

\[ \{ a \approx b \rightarrow (a(u|_v v))^* \} \cup [a \leq b \rightarrow (a(u|_v b v))^*] \cup [b \leq a \rightarrow (b(a(u|_v v)))^*] \]

= \{expand prefix-closure\}

\[ \{ a \approx b \rightarrow \{ \epsilon \} \cup a(u|_v v)^* \} \cup [a \leq b \rightarrow \{ \epsilon \} \cup a(u|_v b v)^*] \cup [b \leq a \rightarrow \{ \epsilon \} \cup b(a(u|_v v))^*] \]

= \{guarded set property\}

\[ \{ \epsilon \} \cup [a \approx b \rightarrow a(u|_v v)^*] \cup [a \leq b \rightarrow a(u|_v b v)^*] \cup [b \leq a \rightarrow b(a(u|_v v))^*] \]

= \{induction\}

\[ \{ \epsilon \} \cup [a \approx b \rightarrow a(u|_v v)] \cup [a \leq b \rightarrow a(u|_v b v)] \cup [b \leq a \rightarrow b(a(u|_v v))] \]

= \{definition of \( au|_v bv \)\}

\[ \{ \epsilon \} \cup (au|_v bv) \]

= \( \{ \epsilon \in (au|_v bv) \} \)

= \( au|_v bv \)

Lemma 9 Suppose \( p \in u|_v v \). Then there are prefixes \( u' \) of \( u \) and \( v' \) of \( v \) such that

1. \( p \in u' + v' \),

2. \( u'.time \leq v.time \) and \( v'.time \leq u.time \)

Proof: Proof is by induction on the length of \( p \).

- \( p = \epsilon \): Let \( u' = v' = \epsilon \). Then, \( p \in u' + v' \). Also, \( u'.time = 0 \leq v.time \) and \( v'.time = 0 \leq u.time \).

- \( p \neq \epsilon \): We rename the terms to get \( cp \in au|_v bv \). We consider the three cases in the inductive definition of \( au|_v bv \).

Case 1) \( a \approx c \) and \( p \in u|_v v \):

Inductively, for some prefix \( u' \) of \( u \) and \( v' \) of \( v \), we have \( p \in u' + v' \), \( u'.time \leq v.time \) and \( v'.time \leq u.time \). Then, \( u' \) is a prefix of \( au \) and \( bv' \) of \( bv \). From \( a \approx c \) and \( p \in u' + v' \), \( cp \in au|_v bv' \). Further, from \( u'.time \leq v.time \) and \( a = b \), \( (au').time \leq (bv').time \). Symmetrically, \( (bv').time \leq (au').time \).

Case 2) \( a \leq c \) and \( p \in u|_v v \):

Inductively, for some prefix \( u' \) of \( u \) and \( w' \) of \( bv \), we have \( p \in u' + w' \), \( u'.time \leq (bv).time \) and \( w'.time \leq u.time \). We consider two cases, (2.1) \( w' = \epsilon \) and (2.2) \( w' \neq \epsilon \).

Case 2.1) \( w' = \epsilon \): From \( p \in u' + w', p = u' \). Then, \( cp = ap = au|_v (au|_v w') \), and \( au' \) is a prefix of \( au \) and \( w' \) of \( bv \). Next we show \( (au').time \leq (bv).time \) and \( w'.time \leq (au).time \). The latter one is trivial since \( w' = \epsilon \).

To show \( (au').time \leq (bv).time \), consider two cases.

- \( u' = \epsilon \): \( (au').time = a.time \) \{from \( a \leq b \} \leq b.time \leq (bv.time) \).
$u' \neq \epsilon$: $(au').time = u'.time \{ \text{given} \} \leq (bv.time)$.

Case 2.2) $w' \neq \epsilon$: Since $w'$ is a prefix of $bv$, $w' = bv'$ for some prefix $v'$ of $v$.

\[
cp \in \{ c = a, p \in u' +_a w' = u' +_a bv' \} \\
a(u' +_a bv') \subseteq \{ a \preceq b; \text{apply definition of } +_a \} \\
u' +_a bv'
\]

To show $(au').time \leq (bv).time$, consider two cases.

$u' = \epsilon$: $(au').time = a.time \{ \text{from } a \preceq b \} \leq b.time \leq (bv.time)$.

$u' \neq \epsilon$: $(au').time = u'.time \{ \text{given} \} \leq (bv.time)$.

To show $(bv').time \leq (au).time$:

$(bv').time = u'.time \{ \text{given} \} \leq u.time \leq (au).time$.

Case 3) $b \preceq x_a b, c = b$ and $p \in au|_a v$:

Similar to case (2).

Lemma 10 $uc +_a vc \subseteq uc|_a vc$, where $c$ is an other-substitution.

Proof: We prove the result using induction on the combined length of $u$ and $v$.

- $u = \epsilon$ and $v = \epsilon$: $uc +_a vc = c +_a c = c(\epsilon +_a \epsilon) = \{ c \}$, and $uc|_a vc = c|_a \epsilon = \{ \epsilon, c \}$.

- $u = \epsilon$ and $v = bv'$:

\[
uc +_a vc = \{ u = \epsilon \text{ and } v = bv' \} \\
c +_a bv'c \subseteq \{ \text{since } c \text{ is an other-substitution, } \neg(c \preceq b); \text{ from definition of } +_a \} \\
\langle c \approx, b \rightarrow c(\epsilon +_a v'c) \rangle \cup \langle b \preceq x_a c \rightarrow b(c +_a v'c) \rangle \\
= \{ v'c \text{ contains an other-substitution; so, } \epsilon +_a v'c = \phi \} \\
\langle b \preceq x_a c \rightarrow b(c +_a v'c) \rangle \\
\subseteq \{ \text{apply induction} \} \\
\langle b \preceq x_a c \rightarrow b(c|_a v'c) \rangle \\
\subseteq \{ \text{definitions of the two kinds of guarded sets} \} \\
\langle b \preceq x_a c \rightarrow b(c|_a v'c) \rangle \\
\subseteq \{ \text{definition of } |_a \} \\
c|_a bv'c \\
= \{ u = \epsilon \text{ and } v = bv' \} \\
uc|_a vc
\]

- $v = \epsilon$ and $a = au'$: The proof is similar to the previous case.
\[ \begin{align*} \text{Lemma 12} \\
&\text{u}c + _+ v
\end{align*} \]

\[ \begin{align*}
\{ v = \epsilon \text{ and } a = au' \} \\
\{ \text{since } c \text{ is an other-substitution, } c \prec\prec_{a} a; \text{ from definition of } \prec\prec \}
\end{align*} \]

\[ \begin{align*}
\{ a \approx_{x} c \rightarrow c(u'c + _+ \epsilon) \} \cup \{ a \lessdot_{x} c \rightarrow a(u'c + _+ c) \}
\end{align*} \]

\[ \begin{align*}
\{ u'c \text{ contains a substitution; so, } u'c + _+ \epsilon = \phi \}
\end{align*} \]

\[ \begin{align*}
\{ \text{apply induction} \}
\end{align*} \]

\[ \begin{align*}
\{ \text{definitions of the two kinds of guarded sets} \}
\end{align*} \]

\[ \begin{align*}
\{ \text{definition of [ ]} \}
\end{align*} \]

\[ \begin{align*}
\{ v = \epsilon \text{ and } a = au' \}
\end{align*} \]

\[ \begin{align*}
u|_{v}c
\end{align*} \]

\[ \begin{align*}
\bullet u = au' \text{ and } v = bv':
\end{align*} \]

\[ \begin{align*}
au'c + _+ bv'c
\end{align*} \]

\[ \begin{align*}
\{ \text{definition of } + \}
\end{align*} \]

\[ \begin{align*}
\{ a \approx_{x} b \rightarrow a(u'c + _+ v'c) \} \cup \{ a \lessdot_{x} b \rightarrow a(u'c + _+ c) \} \cup \{ b \lessdot_{x} a \rightarrow b(u'c + _+ v'c) \}
\end{align*} \]

\[ \begin{align*}
\{ \text{induction} \}
\end{align*} \]

\[ \begin{align*}
\{ a \approx_{x} b \rightarrow a(u'c + _+ v'c) \} \cup \{ a \lessdot_{x} b \rightarrow a(u'c + _+ c) \} \cup \{ b \lessdot_{x} a \rightarrow b(u'c + _+ v'c) \}
\end{align*} \]

\[ \begin{align*}
\{ \text{definitions of the two kinds of guarded sets} \}
\end{align*} \]

\[ \begin{align*}
\{ a \approx_{x} b \rightarrow a(u'c + _+ v'c) \} \cup \{ a \lessdot_{x} b \rightarrow a(u'c + _+ c) \} \cup \{ b \lessdot_{x} a \rightarrow b(u'c + _+ v'c) \}
\end{align*} \]

\[ \begin{align*}
\text{Lemma 11} \text{ Let } c \text{ be an other-substitution and } pc \in u + _+ v. \text{ Then, } u = u'c \text{ and } v = v'c, \text{ for some } u' \text{ and } v'.
\end{align*} \]

Proof: Proof is by induction on the length of \( p \).

\[ \begin{align*}
\bullet p = \epsilon: \text{ Then, } c \in u + _+ v. \text{ This can not be derived by the base rule, because } c \text{ is an other-substitution. } \text{So, } c \in u + _+ v \text{ is derived by the inductive rule. We examine each term in that rule.} \\
\text{a } \approx_{x} b: \text{ Then, } a = b = c \text{ and } \epsilon \in u + _+ v. \text{ From Observation 16, page 51, } u = \epsilon \wedge v = \epsilon. \text{ Hence, } au = c \text{ and } bv = c, \text{ as required.} \\
\text{a } \lessdot_{x} b: \text{ Since } a \text{ is a base event, } a \neq c. \text{ Therefore, this term can not derive } c. \\
b \lessdot_{x} a: \text{ Since } b \text{ is base or own-substitution, } b \neq c. \text{ Therefore, this term can not derive } c.
\end{align*} \]

\[ \begin{align*}
\bullet p \neq \epsilon: \text{ Let } pc = dp'c. \text{ As before, the base rule can not be used for deriving } pc. \text{ In the inductive rule, } p'c \text{ has to be generated by either } u + _+ v, \text{ au } + _+ v \text{ or } u + _+ bv. \text{ Inductively, in each case, } p'c \in u'c + _+ v'c, \text{ for some } u' \text{ and } v'.
\end{align*} \]

\[ \begin{align*}
\text{Lemma 12} \begin{align*}
(u + _+ v)t = u + _+ vt.
\end{align*} \]
CHAPTER 2. COMBINATORS APPLIED TO EXECUTIONS

56

Proof: Apply the definition of \(|\) to both sides. Note that \(a_t \approx \_ b_t \equiv a \approx \_ b\), \(a_t \preceq b_t \equiv a \preceq b\) and \(b_t \preceq \_ a_t \equiv b \preceq \_ a\). The result follows by applying induction.

2.5.2 \([f < x < g] \subseteq [f] < x < [g]\)

Lemma 13 Let \(f < x < g \Rightarrow h\), where the publication rule, (ASYM2V), was not used in forming \(q\). Then,

(*) there exist \(f \overset{u}{\Rightarrow} f'\) and \(g \overset{v}{\Rightarrow} g'\), such that

(*) \(d_1(u, v)\),

(*) \(q \in u + x, v\), and

(*) \(h = f'' < x < g''\), where \(f'' = (f')^q.time - u.time\), \(g'' = (g')^q.time - v.time\)

Proof: Proof is by induction on the length of \(q\).

\(\bullet\) \(q = \epsilon\): Then \(f < x < g \Rightarrow f < x < g\). So, \(h = f < x < g\). Let \(u = \epsilon, v = \epsilon\), \(f' = f, g' = g\). Then, \(q.time = u.time = v.time = 0\). Now,

1. \(f \overset{u}{\Rightarrow} f'\) and \(g \overset{v}{\Rightarrow} g'\), from \(f \overset{\epsilon}{\Rightarrow} f\), \(g \overset{\epsilon}{\Rightarrow} g\)
2. \(d_1(u, v)\), from \(d_1(\epsilon, \epsilon)\),
3. \(q \in u + v\), from \(\epsilon \in \epsilon + \epsilon\)
4. \(h = (f')^q.time - u.time < x < (g')^q.time - v.time\), from \(h = f < x < g = f^0 < x < g^0\)

\(\bullet\) \(q = ap_t\) where \(a\) is an other-substitution: Then, \(f \overset{a}{\Rightarrow} f_1, g \overset{a}{\Rightarrow} g_1\) and \(f_1 < x < g_1 \Rightarrow h\). Applying induction on \(f_1 < x < g_1 \Rightarrow h\), we get

1. there exist \(f_1 \overset{u'}{\Rightarrow} f_2\), and \(g_1 \overset{v'}{\Rightarrow} g_2\), such that
2. \(d_1(u', v')\),
3. \(p \in u' + x'\)
4. \(h = f_3 < x < g_3\) where \(f_3 = (f_2)^p.time - u'.time\), \(g_3 = (g_2)^p.time - v'.time\)

Let \(u = au', v = av', f' = f_2, g' = g_2\).

Now, we show the required items under (*).

(*) \(f \overset{u}{\Rightarrow} f'\) and \(g \overset{v}{\Rightarrow} g'\):

\(f \overset{u}{\Rightarrow} f'\); \(f \overset{a}{\Rightarrow} f_1 \overset{u'}{\Rightarrow} f_2 = f'\). Since \(u = au', f \overset{u}{\Rightarrow} f'\).
\(g \overset{v}{\Rightarrow} g'\): similarly.

(*) \(d_1(u, v)\): From \(d_1(u', v')\) and that \(a\) is an other-substitution.

(*) \(q \in u + x\):
\[ q = \{ \text{definition of } q \} \]
\[ a \in \{ p \in u' + v' \text{ implies } p_t \in u'_t + v'_t \} \]
\[ a(u'_t + v'_t) \subseteq \{ a \text{ is an other-substitution; apply definition of } + \} \]
\[ a(u'_t + v'_t) = \{ u = au'_t, v = av'_t \} \]
\[ u + v \]

\[ (*4) \quad h = (f')^{q\text{.time} - u\text{.time}} < x < (g')^{q\text{.time} - v\text{.time}}. \]

We are given
\[ h = f_3 < x < g_3 \text{ where } f_3 = (f_2)^{p\text{.time} - u'\text{.time}}, g_3 = (g_2)^{p\text{.time} - v'\text{.time}}. \]

Since \( f' = f_2, g' = g_2 \), it is sufficient to show that
\[ q\text{.time} - u\text{.time} = p\text{.time} - u'\text{.time} \]
\[ q\text{.time} - v\text{.time} = p\text{.time} - v'\text{.time} \]

Since \( q = ap_t \), \( u = au'_t \) and \( v = av'_t \), the results follow from Observation 14, page 50.

\[ \bullet q = ap_t, \text{ where } a \text{ is own-substitution:} \]
Then, \( f < x < g \xrightarrow{a} f_1 < x < g_1 \xrightarrow{p} h. \)

From the definition of own-substitution, \( f_1 = f^t \) and \( g \xrightarrow{a} g_1. \)

Applying induction on \( f_1 < x < g_1 \xrightarrow{p} h \), we get:

1. there exist \( f_1 \xrightarrow{u'} f_2 \) and \( g_1 \xrightarrow{v'} g_2 \), such that
2. \( d_1(u', v') \),
3. \( p \in u' + v' \)
4. \( h = f_3 < x < g_3 \text{ where } f_3 = (f_2)^{p\text{.time} - u'\text{.time}}, g_3 = (g_2)^{p\text{.time} - v'\text{.time}} \)

Case 1) Suppose \( u' \neq c \): Let \( u = u'_t, v = av'_t, f' = f_2, g' = g_2. \) We show the required items under \((*)\).

\[ (*1) \quad f \xrightarrow{u} f' \text{ and } g \xrightarrow{v} g': \]
\[ f \xrightarrow{u'} f': \text{ Given} \]
\[ f_1 \xrightarrow{u} f_2 \]
\[ \Rightarrow \{ f_1 = f^t \} \]
\[ f^t \xrightarrow{v'} f_2 \]
\[ \Rightarrow \{ \text{time-shift} \} \]
\[ f \xrightarrow{u'_t} f_2 \]
\[ \Rightarrow \{ u = u'_t, f' = f_2 \} \]
\[ f \xrightarrow{v} f' \]
CHAPTER 2. COMBINATORS APPLIED TO EXECUTIONS

58

g \Rightarrow g': Given

\[ \begin{align*}
g & \Rightarrow g_1 \Rightarrow g_2 \\
\Rightarrow & \{\text{rewriting}\} \\
g & \Rightarrow g_2 \\
\Rightarrow & \{av_1' = v, g_2 = g'\} \\
g & \Rightarrow g'
\end{align*} \]

(*2) \( d_1(u,v) \): Since \( u = u'_t \), and \( u'_t \) contains no own-substitution, from \( d_1(u'_t, v') \), we conclude that \( u \) has no own-substitution. And, \( v = av'_t \) contains no publication, because \( v' \) does not contain any, from \( d_1(u'_t, v') \).

(*3) \( q \in u + v \):

\[ \begin{align*}
q &= \{\text{definition of } q\} \\
ap_t &= \{p \in u' + v' \text{ implies } p_t \in u'_t + v'_t\} \\
&\subseteq \{a \text{ is an own-substitution at time } t; \text{ apply definition of } +\} \\
u'_t + av'_t &= \{u = u'_t, v = av'_t\} \\
u + v
\end{align*} \]

(*4) \( h = (f')q.\text{time} - u.\text{time} < x < (g')q.\text{time} - v.\text{time} \):

We are given

\[ h = f_3 < x < g_3 \text{ where } f_3 = (f_2)p.\text{time} - u'.\text{time}, \ g_3 = (g_2)p.\text{time} - v'.\text{time}. \]

Since \( f' = f_2, g' = g_2 \), it is sufficient to show that

\[ \begin{align*}
q.\text{time} - u.\text{time} &= p.\text{time} - u'.\text{time} \\
q.\text{time} - v.\text{time} &= p.\text{time} - v'.\text{time}
\end{align*} \]

Since \( q = ap_t, u = u'_t \) where \( u' \neq \epsilon \), and \( v = av'_t \), the results follow from Observation 14, page 50.

Case 2) Suppose \( u' = \epsilon \): From \( p \in u' + v' \) and \( u' = \epsilon \), we conclude that \( p = v' \) and \( v' \) has no other-substitution. We are given \( f_1 \xrightarrow{u'=\epsilon} f_2 \), so \( f_1 = f_2 \), and \( g_1 \xrightarrow{v'=p} g_2 \).

Let \( u = \epsilon, v = av'_t = ap_t = q, f' = f, \) and \( g' = g_2 \). We show the required items under (*).

(*1) \( f \xrightarrow{v} f' \) and \( g \xrightarrow{v} g' \):

\[ f \xrightarrow{v} f' \]: Follows from \( f \xrightarrow{u'=\epsilon} f = f' \)

\[ g \xrightarrow{v} g' \]: Given

\[ \begin{align*}
g & \Rightarrow g_1 \Rightarrow g_2 \\
\Rightarrow & \{\text{rewriting}\}
\end{align*} \]
CHAPTER 2. COMBINATORS APPLIED TO EXECUTIONS

\[ g \xrightarrow{av'} g_2 \]
\[ \Rightarrow \{ av'_t = v, g_2 = g' \} \]
\[ g \xrightarrow{v} g' \]

(*2) \( d_1(u, v) \): Since \( u = \epsilon \), it contains no own-substitution. From \( d_1(u', v') \), there is no publication in \( v' \); further, \( a \) is a substitution; so, \( v \) has no publication.

(*3) \( q \in u + v \): Given
\[ p = v' \]
\[ \Rightarrow \{ q = ap_t = av'_t = v, \]
\[ v \text{ has no other-substitution, because } v' \text{ has none and } a \text{ is own-substitution} \}
\[ q \in \epsilon + v \]
\[ \Rightarrow \{ u = \epsilon \} \]
\[ q \in u + v \]

(*4) \( h = (f')^{q.time-u.time} <x< (g')^{q.time-v.time} \):
We are given
\[ h = f_3 <x< g_3 \]
where \( f_3 = (f_2)^{p.time-u'.time}, g_3 = (g_2)^{p.time-v'.time} \).
First, we show \( f_3 = (f')^{q.time-u.time} \).

\[
\begin{align*}
& f_3 \\
& = \{ \text{given} \} \\
& \quad (f_2)^{p.time-u'.time} \\
& \quad = \{ \text{from } f_1 \xrightarrow{f} f_2, f_2 = f_1, \text{ and } f_1 = f' \} \\
& \quad (f')^{p.time-u'.time} \\
& \quad = \{ \text{arithmetic} \} \\
& \quad f + p.time - u'.time \\
& \quad = \{ q.time = (ap_t).time = t + p.time, u.time = u'.time = 0; \}
\quad \text{so } t + p.time - u'.time = q.time - u.time \\
& \quad f_q.time - u.time \\
& \quad = \{ f = f' \} \\
& \quad (f')^{q.time-u.time} \\
\end{align*}
\]
Next, we show \( g_3 = (g')^{q.time-v.time} \).

\[
\begin{align*}
& g_3 \\
& = \{ \text{given} \} \\
& \quad (g_2)^{p.time-v'.time} \\
& \quad = \{ g_2 = g' \} \\
& \quad (g')^{p.time-v'.time} \\
& \quad = \{ p.time - v'.time = 0, \text{ from } p = v'; \text{ also, } q.time - v.time = 0, \text{ from } q = v \} \\
& \quad (g')^{q.time-v.time} \\
\end{align*}
\]

\[ q = ap_t, \text{ where } a \text{ is base:} \]
Let \( q \xrightarrow{a} g_1 \) so that \( f <x< g \xrightarrow{a} f' <x< g_1 \xrightarrow{P} h \). The proof for this case is identical to the last case where \( a \) was an own-substitution.
The proof is similar for the case where \( f \xrightarrow{a} f_1 \) so that \( f \prec x \prec g \xrightarrow{a} f_1 \prec x \prec g' \xrightarrow{h} \).

**Theorem 15** \([ f \prec x \prec g ] \subseteq [ f ] \prec x \prec [ g ]\)

Proof: Let \( p \in [ f \prec x \prec g ] \). We show \( p \in [ f ] \prec x \prec [ g ] \). We consider two cases for the execution \( p \) of \( f \prec x \prec g \): (1) rule (ASYM2V) was not used in the execution \( p \), and (2) (ASYM2V) was used.

- **(ASYM2V) was not used in the execution \( p \):** Since \( f \prec x \prec g \xrightarrow{c} \), from Observation 12, page 50, \( f \prec x \prec g = \xrightarrow{c} \), where \( c \) is an other-substitution, and \( c.time = p.time \). Then, applying Lemma 13, page 56,

  1. there exist \( f \xrightarrow{u} f' \) and \( g \xrightarrow{v} g' \), such that
  2. \( d_1(u, v) \),
  3. \( pc \in u +_x v \)

  Now,

  \[
  pc \in u +_x v
  \Rightarrow \{ \text{from Lemma 11, page 55} \}
  pc \in u'c + v'c, \text{ where } u = u'c \text{ and } v = v'c
  \Rightarrow \{ \text{from Lemma 10, page 54, } u'c + v'c \subseteq u|c| v'c = u|c| v \} \]

  \[
  pc \in u|c| v
  \Rightarrow \{ \text{given } d_1(u, v), u \prec x \prec v = u|c| v \}
  \Rightarrow \{ \text{given } f \xrightarrow{u}, u \in [ f ] ; \text{ similarly, } v \in [ g ] \}
  \Rightarrow \{ \text{given } f \xrightarrow{u}, u \in [ f ] \prec x \prec [ g ] \}
  \]

- **(ASYM2V) was used in the execution \( p \):** Then, \( p = q(t, \tau)rt \), where \( f \prec x \prec g \xrightarrow{h} \), and \( t = q.time + s \).

  Applying Lemma 13, page 56, on \( f \prec x \prec g \xrightarrow{h} \),

  1. there exist \( f \xrightarrow{u} f' \) and \( g \xrightarrow{v} g' \), such that
  2. \( d_1(u, v) \),
  3. \( q \in u +_x v \), and
  4. \( h = f'' \prec x \prec g'' \), where \( f'' = (f')^{q\cdot u.time} \), \( g'' = (g')^{q\cdot v.time} \)

  Also, \( g'' \xrightarrow{s,lm} \), and \( [m/x](f'')^s = \).

  Let \( j = u(t, [m/x])rt \), and \( k = v(t, !m) \). We first show that (1) \( j \in [ f ] \), (2) \( k \in [ g ] \), and (3) \( d_2(j, k) \), from which we have an easy proof of \( p \in [ f ] \prec x \prec [ g ] \).
(1) $j \in \llbracket f \rrbracket$: We show

$$f \xRightarrow{u} f' \xrightarrow{(t_1, [m/x])} [m/x].(f'')^s \xRightarrow{r} , \text{ where } t_1 + u.time = t$$

Hence, $j = u(t, [m/x])r_t \in \llbracket f \rrbracket$.

To prove the result, we already have $f \xRightarrow{u} f'$ and $[m/x].(f'')^s \xRightarrow{r}$ . So, we need only prove $f' \xrightarrow{(t_1, [m/x])} [m/x].(f'')^s$.

$$f' \xrightarrow{(t_1, [m/x])} \{\text{application of substitution}\}
\xrightarrow{[m/x].(f')^t_1}
= \{t_1 = t - u.time = \{t = q.time + s\ q.time - u.time + s\}
\xrightarrow{[m/x].(f'q.time - u.time + s)}
= \{f'' = (f')q.time - u.time\}
\xrightarrow{[m/x].(f'')^s}$$

(2) $k \in \llbracket g \rrbracket$: we have to show $g \xRightarrow{v} g'$ and $[m/x].(f'')^s \xRightarrow{r}$. So, we show

$$g' \xrightarrow{s.lm} \{g' = (g')q.time - v.time\}
\xrightarrow{q.time - v.time + s.lm}
= \{from \ t = q.time + s, q.time - v.time + s = t - v.time = t_2\}
\xrightarrow{g' \xrightarrow{t_2.lm}}$$

(3) $d_2(j, k)$: Both $j$ and $k$ are of the required form. We are given $d_1(u, v)$. To see $d_0(u, v)$:

$$q \in u + v
\Rightarrow \{\text{set theory}\}
\Rightarrow u + v \neq \phi
\Rightarrow \{\text{from Lemma 7, page 51}\}
\Rightarrow \{\text{from Lemma 7, page 51}\}
\Rightarrow d_0(u, v)$$

Now, we show that $p \in \llbracket f \rrbracket < x < \llbracket g \rrbracket$.

$$= \{\text{given}\}
\Rightarrow q(t, \tau) r_t
\in \{q \in u + v\}
\Rightarrow (u + v)(t, \tau) r_t
= \{j = u(t, [m/x])r_t, k = v(t, !m) \text{ and } d_2(j, k) \text{ holds}\}
\Rightarrow \{d_2(j, k) \text{ holds}\}$$
\[ j < x < k \subseteq \{ j \in [f], k \in [g] \} \]

\[ [f] < x < [g] \]

### 2.5.3 \([f] < x < [g] \subseteq [f] < x < [g]\)

**Lemma 14** Suppose \( f \xrightarrow{a} f', g \xrightarrow{a} g' \) and \( d_0(u, v) \).

Let \( T = \max(u.time, v.time) \), \( f'' = (f')^{T-u.time} \), \( g'' = (g')^{T-v.time} \).

Then, for any \( p \), where \( p \in u +_v v \), \( f < x < g \Rightarrow f'' < x < g'' \).

**Note:** The lemma does not assert that under the given conditions, \( p \) is an execution of \( f < x < g \). This is because \( f'' \) or \( g'' \) may be \( \perp \). In order to show that \( p \) is an execution of \( f < x < g \), it has to be shown that neither of these expressions is \( \perp \).

**Proof:** Proof is by induction on the combined lengths of \( u \) and \( v \).

- **\( u = \epsilon \) and \( v = \epsilon \):** Then, \( f' = f \) and \( g' = g \). Also, \( u.time = v.time = T = 0 \), and \( f'' = f \) and \( g'' = g \). Since \( p = \epsilon \), we have to show \( f < x < g \Rightarrow f < x < g \), which follows.

- **\( u \neq \epsilon \) and \( v = \epsilon \):** \( p \in u +_v v \) means that \( p = u \) and \( u \) has no substitutions. Given \( u \neq \epsilon \), we may write \( u = au' \). Here \( a \) is a base event because \( u \) has no substitution. Then, \( f \xrightarrow{a} f_1 \xrightarrow{u'} f' \).

\[
f < x < g \\
xrightarrow{a} \{ f \xrightarrow{a} f_1, \ \text{and} \ a \ \text{is a base event} \} \\
xrightarrow{u'} \{ \text{Induction on } f_1 \xrightarrow{u'} f' \ \text{and} \ g' \xrightarrow{a} g'; \ \\
\text{let } T' = \max(u'.time, 0) = u'.time; \ \\
(f')^{T-u'.time} < x < (g')^{T-v'.time} \}
\]

\[
\begin{align*}
T - u.time & = 0 = T' - u'.time; \\
T - v.time & = T = u.time + v.time = t + T', \text{ so,} \\
(g')^{T-v'.time} & = (g')^{T-v.time} \\
(f')^{T-u.time} & < x < (g')^{T-v.time}
\end{align*}
\]

- **\( u = \epsilon \) and \( v \neq \epsilon \):** Similar to the above.

- **\( u \neq \epsilon \) and \( v \neq \epsilon \):** Let \( p = aq \). We consider three cases, (1) \( a \) is base (2) \( a \) is an other-substitution, and (3) \( a \) is an own-substitution.

**Case (1) \( a \) is base:** We have \( p = aq \in u +_v v \), where \( a \) is base. Without loss in generality, assume that \( u = u' \) and \( v = av' \), so that \( f \xrightarrow{u=u'} f', g \xrightarrow{a} g_1 \xrightarrow{v'} g' \).

\[
p \in u +_v v \\
\Rightarrow \{ p = aq, u = u' \ \text{and} \ v = av' \}
\]
CHAPTER 2. COMBINATORS APPLIED TO EXECUTIONS

$$a q_i \in u'_i + a v'_i$$
$$\Rightarrow \{ a \text{ is base; so, } u'_i + a v'_i = a(u'_i + v'_i) = a(u' + v') \}$$
$$a q_i \in a(u' + v')$$
$$\Rightarrow \{ \text{obviously} \}$$
$$q \in u' + v'$$

From \( d_0(u, v) \), we have \( d_0(u', v') \). Let \( T' = \max(u'.time, v'.time) \).

$$f <_x < g$$
$$\overset{a}{\Rightarrow} \{ g \overset{a}{\Rightarrow} g_1 \}$$
$$f' <_x < g_1$$
$$\Rightarrow \{ \text{induction on } f' \Rightarrow f', g_1 \overset{v'}{\Rightarrow} g' \text{ using } g \in u' + v' \text{ and } d_0(u', v') \}$$
$$f_2 <_x < g_2$$, where \( f_2 = (f')^{T' - u'.time}, g_2 = (g')^{T' - v'.time} \)
$$\Rightarrow \{ \text{Using Observation 15, page 50,}$$

\[ T' - u'.time = T - u.time, \quad T' - v'.time = T - v.time \]

$$f_2 <_x < g_2$$, where \( f_2 = (f')^{T' - u'.time}, g_2 = (g')^{T' - v'.time} \)

Case (2) \( a \) is an other-substitution: We have \( p = a q_i \in u + v \), which means \( u = a u'_i, v = a v'_i \) and \( q \in u' + v' \).

Then, \( f \overset{a}{\Rightarrow} f_1 \overset{v'}{\Rightarrow} f', g \overset{a}{\Rightarrow} g_1 \overset{v'}{\Rightarrow} g' \). Let \( T' = \max(u'.time, v'.time) \). From \( d_0(u, v) \), we have \( d_0(u', v') \). Let \( T' = \max(u'.time, v'.time) \).

$$f <_x < g$$
$$\overset{a}{\Rightarrow} \{ a \text{ is an other-substitution; } f \overset{a}{\Rightarrow} f_1 \text{ and } g \overset{a}{\Rightarrow} g_1 \}$$
$$f_1 <_x < g_1$$
$$\Rightarrow \{ \text{induction on } f_1 \Rightarrow f', g_1 \overset{v'}{\Rightarrow} g' \text{ using } q \in u' + v' \text{ and } d_0(u', v') \}$$
$$f_2 <_x < g_2$$, where \( f_2 = (f')^{T' - u'.time}, g_2 = (g')^{T' - v'.time} \)
$$\Rightarrow \{ \text{Using Observation 15, page 50,}$$

\[ T' - u'.time = T - u.time, \quad T' - v'.time = T - v.time \]

$$f_2 <_x < g_2$$, where \( f_2 = (f')^{T' - u'.time}, g_2 = (g')^{T' - v'.time} \)

Case (3) \( a \) is an own-substitution: this case is similar to Case (1) where \( a \) is base.

**Lemma 15** \( u \in \llbracket f \rrbracket, v \in \llbracket g \rrbracket, \) and \( d_1(u, v) \) implies \( u <_x < v \subseteq \llbracket f <_x < g \rrbracket \)

Proof:

\[
p \in u <_x < v
\Rightarrow \{ \text{definition of } u <_x < v \text{ given } d_1(u, v) \}
\]
\[
p \in u |_x v
\Rightarrow \{ \text{from Lemma 9, page 53} \}
\]
\[
p \in u' + v', \text{ where } u' \text{ and } v' \text{ are prefixes of } u \text{ and } v,
\]
\[
u'.time \leq v.time \text{ and } v'.time \leq u.time
\Rightarrow \{ \text{from Lemma 7, page 51, } p \in u' + v' \Rightarrow d_0(u', v') \}
\]
\[
p \in u' + v', \text{ for prefixes } u' \text{ of } u \text{ and } v' \text{ of } v, \text{ and } d_0(u', v'),
\]
CHAPTER 2. COMBINATORS APPLIED TO EXECUTIONS

\[ u' \cdot \text{time} \leq v \cdot \text{time} \text{ and } v' \cdot \text{time} \leq u \cdot \text{time} \]

\[ \Rightarrow \{ u \in [f], v \in [g]; u' \text{ and } v' \text{ are prefixes of } u \text{ and } v \} \]

\[ p \in u' + v', \text{ if } u' \xrightarrow{T} f', g \xrightarrow{T} g', \text{ for some } f' \text{ and } g', \delta_0(u', v') \]

\[ u' \cdot \text{time} \leq v \cdot \text{time} \text{ and } v' \cdot \text{time} \leq u \cdot \text{time} \]

\[ \Rightarrow \{ \text{from Lemma 14, page 62} \} \]

\[ f \xleftarrow{T} x < g \xrightarrow{T} (f')^{\text{time}} \text{ is a prefix of } (g')^{\text{time}}, \]

\[ T = \max(u' \cdot \text{time}, v' \cdot \text{time}), u' \cdot \text{time} \leq v \cdot \text{time} \text{ and } v' \cdot \text{time} \leq u \cdot \text{time} \]

Next, we show that \((f')_{T-u' \cdot \text{time}} \neq \perp \) and \((g')_{T-v' \cdot \text{time}} \neq \perp \); hence, that \(p \in [f < x < g]\), i.e., \( u < x < v \subset [f < x < g] \).

- Given that \( u \in [f], u' \) is a prefix of \( u \), and \( f \xrightarrow{u'} f' \), we have \( f \xrightarrow{T} f' \xrightarrow{T} v \)

where \( u = u'(u'')_{u' \cdot \text{time}} \).

\[ (f')^{u'' \cdot \text{time}} \neq \perp \]

\[ \Rightarrow \{ \text{from } u = u'(u'')_{u' \cdot \text{time}}, u \text{.time} = u' \cdot \text{time} + u'' \cdot \text{time} \} \]

\[ (f')^{u'' \cdot \text{time} - v' \cdot \text{time}} \neq \perp \]

\[ \Rightarrow \{ u' \text{ is a prefix of } u; \text{ so, } u' \cdot \text{time} \leq u \cdot \text{time} \}

Given, \( u' \cdot \text{time} \leq u \cdot \text{time} \). So, \( T = \max(u' \cdot \text{time}, v' \cdot \text{time}) \leq u \cdot \text{time} \).

Hence, \( T - u' \cdot \text{time} \leq u \cdot \text{time} - u' \cdot \text{time} \)

\[ (f')^{T-u' \cdot \text{time}} \neq \perp \]

Similarly, \((g')^{T-v' \cdot \text{time}} \neq \perp \).

**Theorem 16** \([f] < x < [g] \subset [f < x < g]\)

**Proof:** We show

1. \( u \in [f], v \in [g], \text{ and } d_1(u, v) \) implies \( u < x < v \subset [f < x < g] \),
2. \( u \in [f], v \in [g], \text{ and } d_2(u, v) \) implies \( u < x < v \subset [f < x < g] \), and
3. \( u \in [f], v \in [g], \neg d_1(u, v), \text{ and } \neg d_2(u, v) \) implies \( u < x < v \subset [f < x < g] \)

The first case follows from Lemma 15, page 63. The last case is trivial, since \( u < x < v \) in that case is \( \phi \), a subset of any set. So, we prove only the second case.

- \( u \in [f], v \in [g], \text{ and } d_2(u, v) \) implies \( u < x < v \subset [f < x < g] \): Given \( d_2(u, v) \)

we may assume that \( u = u'(t, [m/x])u'' \in [f], v = v'(t, xlm)v'' \in [g] \)

Then \( u < x < v = (u' + x')(t, \tau)u'' \).

We have to show that for any \( p, \) where \( p \in u' + x' \), \( p(t, \tau)u'' \subset [f < x < g] \).
We prove this by showing

\[ f < x < g \xrightarrow{p} f' < x < g' \xrightarrow{T} m/x).f'(t-T) \xrightarrow{u''} \text{, where } T = p \cdot \text{time}, \text{ or} \]

\[ f < x < g \xrightarrow{T} f' < x < g', \text{ for some } f' \text{ and } g', \text{ where} \]

\[ f' < x < g' \xrightarrow{T} m/x).f'(t-T) \xrightarrow{u''} \]

\[ (1) \]

\[ (2) \]

\[ (3) \]
Given $p \in u' + v'$, using Lemma 7, page 51, $p.time = \max(u'.time, v'.time)$. Since $T = p.time$, $T = \max(u'.time, v'.time)$.

Now,

\[ u \in \llbracket f \rrbracket \text{ means } f \xrightarrow{u'} f_1 \xrightarrow{t_1, \llbracket m/x \rrbracket} \llbracket m/x \rrbracket .(f_1)^{t_1} \xrightarrow{u''}, \text{ where} \]

\[ t_1 + u'.time = t. \text{ Also, } (f_1)^{t_1} \neq \bot. \]

\[ v \in \llbracket g \rrbracket \text{ means } g \xrightarrow{v'} g_1 \xrightarrow{t_2, \llbracket m \rrbracket} \llbracket m \rrbracket .(g_1)^{t_2} \neq \bot. \]

(1) $f < x < g \xrightarrow{p} f' < x < g'$: We are given $f \xrightarrow{u'} f_1$, $g \xrightarrow{v'} g_1$. Also, $d_0(u', v')$ follows from $d_2(u, v)$, and $p \in u' + v'$. Applying Lemma 14, page 62, we get

\[ f < x < g \xrightarrow{p} f' < x < g', \text{ where} \]

\[ f' = (f_1)^{T - u'.time}, g' = (g_1)^{T - v'.time} \text{ (Recall } T = \max(u'.time, v'.time)) \]

Next, we show that $f' \neq \bot$ and $g' \neq \bot$. First, from $u = u'(t, \llbracket m/x \rrbracket)u''$, $u'.time \leq t$, and from $v = v'(t, \llbracket m \rrbracket)v''$, $v'.time \leq t$. Therefore, $T = \max(u'.time, v'.time) \leq t$. Now, $t_1 = t - u'.time \geq T - u'.time$. Since $(f_1)^{t_1} \neq \bot$, from (F2), $(f_1)^{T - u'.time} = f' \neq \bot$. Similarly, $g' \neq \bot$.

(2) $f' < x < g' \xrightarrow{p} [m/x].(f')^{T - T}$: From (F2),

\[
\begin{align*}
& g_1 \xrightarrow{t_2, \llbracket m \rrbracket} \\
\Rightarrow & \{ t_2 = t - v'.time = T - v'.time + t - T \} \\
& (g_1)^{T - v'.time} \xrightarrow{T - T, \llbracket m \rrbracket} \\
\Rightarrow & \{ g' = (g_1)^{T - v'.time} \} \\
& (g') \xrightarrow{T - T, \llbracket m \rrbracket} \\
\end{align*}
\]

Hence, from the operational semantics, using (ASYM2V),

\[ f' < x < g' \xrightarrow{p} [m/x].(f')^{T - T} \]

(3) $[m/x].(f')^{T - T} \xrightarrow{u''}$:

\[
\begin{align*}
(f')^{T - T} = & \{ f' = (f_1)^{T - u'.time} \} \\
& (f_1)^{T - u'.time + t - T} \\
= & \{ \text{simplify exponent} \} \\
& (f_1)^{- u'.time} \\
= & \{ t_1 = t - u'.time, \text{ from } (F2) \} \\
& (f_1)^{t_1}
\end{align*}
\]

Given $[m/x].(f_1)^{t_1} \xrightarrow{u'}$, we have $[m/x].(f')^{T - T} \xrightarrow{u''}$. This completes the proof.
Chapter 3

Breadth and Trace Preservation

The goal of this chapter is to show that the traces of $f * g$ can be determined from the traces of $f$ and $g$. Specifically, we show

$$\langle\langle f * g \rangle\rangle = \langle\langle f \rangle\rangle * \langle\langle g \rangle\rangle.$$  \hspace{1cm} (P1)

where $*$ is any orc combinator, $|, >x>$ or $<x<$.

We prove (P1) by first showing, for sets $U$ and $V$,

$$U * V = \overline{U} * \overline{V}.$$  \hspace{1cm} (P2)

Then (P1) follows,

$$\langle\langle f * g \rangle\rangle = \{\text{definition of } \langle\langle f * g \rangle\rangle\}$$

$$= \overline{f * g}$$

$$= \{\text{from Characterization Theorems in Chapter 2, } [f * g] = [f] * [g]\}$$

$$= \overline{f} * \overline{g}$$

$$= \{\text{Use (P2) with } U = [f] \text{ and } V = [g]\}$$

$$= \{\text{from definition, } \langle f \rangle = \overline{f} \text{ and similarly for } g\}$$

$$\langle\langle f \rangle\rangle * \langle\langle g \rangle\rangle$$

The sets $U$ and $V$ in (P2) are not arbitrary, however. In particular, we call set $U$ to be broad (1) if $x \in U$, then $xc \in U$, for any substitution $c$ where $c.time = x.time$, and (2) if $xb \in U$, then $xc \in U$, for any substitution $c$ where $x.time \leq c.time \leq b.time$. The formal definition, given in Section 3.1.1, page 67, is inductively defined to facilitate algebraic manipulations, though it is equivalent to the definition given here. Additionally, we will require that the sets be substitution independent, see Section 1.5.2, page 15. We establish (P2) under these conditions.
Clearly, we have to show that every \[ f \] is broad and substitution independent. The latter result has been proved in Observation 9, page 20. We prove that \[ f \] is broad by induction on the structure of \[ f \]. Base orc expressions are broad, from Lemma 27, page 73. And we show that each combinator preserves breadth, i.e., if \( U \) and \( V \) are broad then so is \( U \ast V \). In this chapter, we discharge both sets of proof obligations: (1) \( U \ast V \) is broad given \( U \) and \( V \) are broad, and (2) \( U \ast V = \overline{U} \ast \overline{V} \), given \( U \) and \( V \) are broad and substitution independent. We prove these results separately for each combinator, after establishing some preliminary results in the following section.

### 3.1 Additional Operators on Sequences

#### 3.1.1 Breadth

The breadth of \( p, \beta(p) \), is the set of sequences that can be generated from \( p \) by applying the substitution rule. Formally,

\[
\begin{align*}
\beta(\epsilon) &= A(0) \\
\beta(ap_t) &= A(t) \cup a(\beta(p))_t, \text{ where } t = \text{a.time}
\end{align*}
\]

Notation: Henceforth, we write \( \beta(p)_t \) for \( (\beta(p))_t \). Note that \( \beta(p)_t \) is different from \( \beta(p_t) \) (see Lemma 20, page 70, for a relationship between the two). We define \( \beta() \) to be coercive, i.e.,

\[
\beta(P) = (\cup p : p \in P : \beta(p)).
\]

Note that \( \beta(\phi) = \phi \).

**Observation 17** \( \beta(\epsilon) \subseteq \beta(p) \), for any \( p \).

**Proof:** \( \beta(\epsilon) = A(0) \), and from the definition of \( \beta() \), \( A(0) \subseteq \beta(p) \), for any \( p \).

**Broad** Define set \( P \) to be broad iff \( P = \beta(P) \).

It follows that if \( P \) and \( Q \) are broad, \( P \cup Q \) is broad:

\[
\beta(P \cup Q) = \beta(P) \cup \beta(Q) = P \cup Q.
\]

Also, that the empty set is broad.

**Lemma 16** \( A(r) \) is broad for any \( r \).

**Proof:** We observe that for \( p \), any finite sequence of substitutions at time 0, \( \beta(p) = A(0) \) (proof is by induction on the length of \( p \)). Therefore, \( \beta(A(0)) = A(0) \), i.e., \( A(0) \) is broad.

Next, observe that any sequence of \( A(r) \) is of the form \( ap_t \), where \( 0 \leq t \leq r \), \( a \) is a substitution at \( t \), and \( p \) is a finite sequence of substitutions at time 0.

\[
\begin{align*}
\beta(ap_t) &= \text{definition of } \beta() \\
&= \{\text{definition of } \beta()\}
\end{align*}
\]
A(t) \cup a\beta(p)_t
= \{p \in A(0). \text{ Hence, } \beta(p) = A(0), \text{ from the the proof above}\}
A(t) \cup A(0)_t
= \{A(0)_t \subseteq A(t)\}
A(t)

We now show that A(r) is broad, i.e., \( \beta(A(r)) = A(r) \).

\[
\beta(A(r)) = \{ \text{coercion} \}
(\cup q : q \in A(r) : \beta(q))
= \{ \text{from above, } \beta(q) = A(t), \text{ where } t = q.time \}
(\cup t : 0 \leq t \leq r : A(t))
= \{ \text{for } t \leq r, A(t) \subseteq A(r) \}
A(r)
\]

**Lemma 17** \( D(t) \) is broad for any \( t \).

Proof: The proof is by induction on the length of \( p \in D(t) \). For \( p = \epsilon \), \( \beta(\epsilon) = A(0) \) and \( A(0) \subseteq D(t) \) by Obs. 7 on page 9. Otherwise \( p = aq_s \), for substitution event \( a \) with \( a.time = s \leq t \).

\[
\beta(aq_s)
= \{ \text{definition of } \beta() \}
A(s) \cup a\beta(q)_s
\subseteq \{ \text{induction on } q \in D(t - s) \}
A(s) \cup A(D(t - s))_s
\subseteq \{ aD(t - s)_s \subseteq D(t) \}
A(s) \cup D(t)
\subseteq \{ A(s) \subseteq A(t) \subseteq D(t) \text{ by Obs. 7} \}
D(t)
\]

**Lemma 18** \( \beta(\beta(p)) = \beta(p) \). So, \( \beta(p) \) is broad.

Proof: Proof is by induction on the length of \( p \). For \( p = \epsilon \), \( \beta(\epsilon) = A(0) \), and \( A(0) \) is broad from Lemma 16, page 67.

\[
\beta(\beta(ap_t))\text{, where } t = a.time
= \{ \text{definition of } \beta() \}
\beta(A(t) \cup a\beta(p)_t)
= \{ \beta() \text{ distributes over union} \}
\beta(A(t)) \cup \beta(a\beta(p)_t)
= \{ \beta(A(t)) = A(t), \text{ from Lemma 16, page 67} \}
A(t) \cup \beta(a\beta(p)_t)
= \{ \text{definition of } \beta() \}
A(t) \cup A(t) \cup a\beta(\beta(p))_t
= \{ \text{induction} \}
A(t) \cup a\beta(\beta(p))_t
= \{ \text{definition of } \beta() \}
\beta(ap_t)
\]
Lemma 19 $p^* \subseteq \beta(p)$.

Proof: Proof is by induction on the length of $p$.

First, we show $\epsilon^* \subseteq \beta(\epsilon)$. $\epsilon^* = \{\epsilon\}$, and $\beta(\epsilon) = A(0)$. And, $\{\epsilon\} \subseteq A(0)$, from the definition of $A(0)$.

Next, we show that $(ap_t)^* \subseteq \beta(ap_t)$, where $t = a.time$.

$$
\begin{align*}
(ap_t)^* &= \{\text{definition of prefix-closure}\} \\
&= \{\epsilon\} \cup a(p_t)^* \\
&= \{(p_t)^* = (p^*)_t, \text{from Lemma 1, page 7}\} \\
&= \{\epsilon\} \cup a(p^*)_t \\
&\subseteq \{\text{\{\epsilon\} \subseteq A(t) and inductively, \ p^* \subseteq \beta(p), so (p^*)_t \subseteq \beta(p)_t\}} \\
&= \{\text{definition of breadth}\} \\
&= \beta(ap_t)
\end{align*}
$$

Corollary 2 $p \in \beta(p)$, and $P \subseteq \beta(P)$.

Proof: $p \in p^*$. From Lemma 19, page 69, $p^* \subseteq \beta(p)$. Therefore, $p \in \beta(p)$. And, $P \subseteq \beta(P)$ follows by applying coercion.

Corollary 3 A broad set is prefix-closed.

Proof: For broad set $P$ and any sequence $q$, we show that $q \in P \Rightarrow q^* \subseteq P$.

$$
\begin{align*}
q \in P \\
\Rightarrow \{\text{apply } \beta(\cdot) \text{ to both sides}\} \\
\beta(q) \subseteq \beta(P) \\
\Rightarrow \{\beta(P) = P, \text{ since } P \text{ is broad}\} \\
\beta(q) \subseteq P \\
\Rightarrow \{q^* \subseteq \beta(q), \text{ from Lemma 19, page 69}\} \\
q^* \subseteq P
\end{align*}
$$

To prove that set $P$ is broad, we can employ any of the following characterizations of a broad set.

Corollary 4 $P$ is broad iff

1. $P = \beta(Q)$, for some set $Q$.
2. $\beta(P) \subseteq P$.
3. for every $p$, where $p \in P$, $\beta(p) \subseteq P$.
4. $P$ is a union of broad sets.
CHAPTER 3. BREADTH AND TRACE PRESERVATION

Proof: (1) If \( P \) is broad, \( P = \beta(P) \), by definition. And, if \( P = \beta(Q) \), for some set \( Q \), then \( \beta(P) = \beta(\beta(Q)) = \beta(Q) = P \); hence \( P \) is broad. (2) From the definition of broad, and Corollary 2, page 69. (3) is a rewriting of (2): \( \beta(P) \subseteq P \) is same as, for all \( p, p \in P, \beta(p) \subseteq P \). (4) If \( P \) is a union of broad sets, say \( P_i \), then \( \beta(P) = \beta(\cup_i P_i) = (\cup_i \beta(P_i)) = (\cup_i P_i) = P \). Conversely, if \( P \) is a broad set then it is a union of broad sets vacuously (to make it non-vacuous, take the union of \( P \) and the empty set, which are both broad).

Lemma 20 For non-empty \( q \), \( \beta(q_s) = A(s) \cup \beta(q)_s \)

Proof: Let \( q = ap_t \), where \( t = a.time \). Then \( q_s = a_s p_{s+t} = bp_{s+t} \), where \( b = a_s \).

\[
\begin{align*}
\beta(q_s) &= \{ q_s = bp_{s+t} \} \\
\beta(bp_{s+t}) &= \{ b.time = a_s.time = a.time + s = s + t ; \text{expand } \beta(bp_{s+t}) \}
\end{align*}
\]

\[
\begin{align*}
A(s + t) &\cup b\beta(p)_{s+t} = \{ A(s + t) = A(s) \cup A(t)_s, \text{from Observation 7, page 9; rewrite } b \}
\end{align*}
\]

And,

\[
\begin{align*}
A(s) &\cup \beta(q)_s \\
\{ q = ap_t \} &= \{ \text{definition of } \beta() \}
\end{align*}
\]

\[
\begin{align*}
A(s) &\cup (A(t) \cup a\beta(p)_t)_s \\
\{ \text{rewriting} \} &= A(s) \cup A(t)_s \cup a_s \beta(p)_{s+t}
\end{align*}
\]

Lemma 21 \( uc \in \beta(u) \), where \( c \) is any substitution and \( c.time = u.time \).

Proof: By induction on the length of \( u \).

- \( u = c \): We write simply \( c \) for the sequence containing just \( c \), in the following proof. We have to show that \( c \in \beta(c) = A(0) \), where \( c \) is any substitution at time 0. This follows from the definition of \( A(0) \).

- \( au_t c_t \in \beta(uu_t) \) where \( c.time = u.time \): Inductively, \( uc \in \beta(u) \), and

\[
\begin{align*}
uc &\in \beta(u) \\
\Rightarrow &\{ \text{time shift applied to both sides} \}
\end{align*}
\]

\[
\begin{align*}
(uc)_t &\in \beta(u)_t \\
\Rightarrow &\{ \text{concatenation applied to both sides} \}
\end{align*}
\]

\[
\begin{align*}
a(uc)_t &\in a\beta(u)_t \\
\Rightarrow &\{ a\beta(u)_t \subseteq \beta(au_t), \text{from definition of } \beta() \}
\end{align*}
\]

\[
\begin{align*}
a(uc)_t &\in \beta(au_t) \\
\Rightarrow &\{ a(uc)_t = au_t c_t \}
\end{align*}
\]

\[
\begin{align*}
a u_t c_t &\in \beta(au_t)
\end{align*}
\]
3.1.2 Visible sequences and Traces

A sequence is visible if it is empty or its last event is non-\(\tau\).

**Lemma 22** For visible sequence \(p\), \(\beta(p) = \beta(\overline{p})\).

Proof: Proof is by induction on the length of \(p\).

\[\beta(\epsilon) = A(0),\] and from Observation 7, page 9, \(A(0) = A(0)\), and
\[\beta(\tau) = \beta(\epsilon) = A(0)\]

Consider \(ap_t\), where \(ap_t\) is visible and \(t = a.time\). Note that \(p\) is visible (\(p\) could be \(\epsilon\)).

\[\beta(ap_t) = \{\text{definition of } \beta()\}
A(t) \cup a\beta(p)_t\]
\[= \{\text{distribute trace over union and concatenation}\}
A(t) \cup (\beta(p))_t\]
\[= \{(\beta(p))_t = (\beta(p))_t, \text{ from Lemma 1, page 7}\}
A(t) \cup \beta(p)_t\]
\[= \{\text{induction using } p \text{ is visible; also use } A(t) = A(t)\}
A(t) \cup \beta(p)_t\]

We show that \(A(t) \cup \overline{\beta(p)}_t = \beta(\overline{ap_t})\). Consider: (1) \(a \neq \tau\), (2) \(a = \tau\).

(1) \(a \neq \tau\): Then \(a = \overline{a}\).

\[A(t) \cup \beta(p)_t\]
\[= \{\text{definition of } \beta() \text{ and } \overline{a}.time = t\}
\beta(\overline{p}_t)\]
\[= \{\overline{p}_t = (\overline{p}_t), \text{ from Lemma 1, page 7}\}
\beta(\overline{p}_t)\]
\[= \{\text{distribute trace over concatenation}\}
\beta(\overline{ap}_t)\]

(2) \(a = \tau\): then \(a = \epsilon\) and \(p\) is non-empty and visible, because \(ap_t\) is visible.

\[A(t) \cup \beta(p)_t\]
\[= \{\overline{a} = \epsilon\}
A(t) \cup \beta(p)_t\]
\[= \{p \text{ is non-empty and visible; so } \overline{p} \text{ is non-empty. from Lemma 20, page 70}\}
\beta((\overline{p})_t)\]
\[= \{\overline{p}_t = (\overline{p}_t), \text{ from Lemma 1, page 7}\}
\beta(\overline{p}_t)\]
\[= \{\overline{a} = \epsilon. \text{ So, } \beta(\overline{ap}_t) = \beta(\overline{ap}_t) = \beta(\overline{ap}_t)\}
\beta(\overline{ap}_t)\]

**Lemma 23** Given that \(U\) is broad, \(\overline{U}\) is broad.
Proof: We show that $\beta(x) \subseteq \overline{U}$ for every $x \in \overline{U}$. Then $\overline{U}$ is broad, from Corollary 4, page 69.

Given that $x \in \overline{U}$ there is $y \in U$ such that $x = y$. We can assume that $y$ is visible (otherwise, $z$, the longest prefix of $y$ that is visible satisfies $x = z$ and $z \in U$, from prefix-closure of $U$). Note that if $x = \epsilon$, then $y = \epsilon$ satisfies the requirements.

\[
\begin{align*}
y \in U \\
\Rightarrow & \quad \{U \text{ is broad; use Corollary 4, page 69}\} \\
\beta(y) \subseteq U \\
\Rightarrow & \quad \{\text{apply trace to both sides}\} \\
\overline{\beta(y)} \subseteq \overline{U} \\
\Rightarrow & \quad \{y \text{ is visible; from Lemma 22, page 71, } \overline{\beta(y)} = \beta(\overline{y})\} \\
\beta(\overline{y}) \subseteq \overline{U} \\
\Rightarrow & \quad \{x = \overline{y}\} \\
\beta(x) \subseteq \overline{U}
\end{align*}
\]

**Event Removal from front of a sequence**

We have defined the removal operator in page 6. Repeating the definition,

\[
\begin{align*}
u \setminus a = & \begin{cases} 
\{v\} & \text{if } u = av_t \text{ where } t = a.\text{time} \\
\emptyset & \text{otherwise}
\end{cases} \\
U \setminus a = & \{v | av_t \in U\}, \text{ where } t = a.\text{time}
\end{align*}
\]

**Lemma 24** Given that $U$ is broad, $U \setminus a$ is broad.

Proof: We show that for any $p, p \in U \setminus a$, $\beta(p) \subseteq U \setminus a$. The result follows from Corollary 4, page 69 part(3).

\[
\begin{align*}
p \in U \setminus a \\
\Rightarrow & \quad \{\text{definition of } U \setminus a\} \\
& \quad ap_t \in U, \text{ where } t = a.\text{time} \\
\Rightarrow & \quad \{U \text{ broad; use Corollary 4, page 69}\} \\
& \quad \beta(ap_t) \subseteq U \\
\Rightarrow & \quad \{a\beta(p)_t \subseteq \beta(ap_t), \text{ from the definition of breadth}\} \\
& \quad a\beta(p)_t \subseteq U \\
\Rightarrow & \quad \{\text{definition of } U \setminus a\} \\
& \quad \beta(p) \subseteq U \setminus a
\end{align*}
\]

**Reducing times in a sequence**

Given set $V$, define $V_{-t}$, for $t \geq 0$, to be

\[V_{-t} = \{v | v_t \in V\}\]

Thus, every sequence in $V$ that starts at or after $t$ has all its event times reduced by $t$, and all other non-empty sequences are discarded. The empty sequence, if it is in $V$, is retained because $\epsilon_t = \epsilon$. 
Observation 18 \( A(t)_{-t} = A(0) \), from definition.

Lemma 25 Given that \( V \) is broad and some \( u_t \in V \), where \( u \neq \epsilon \), \( V_{-t} \) is broad.

Proof: First, note the essential requirement that some sequence \( u_t \), whose first event starts at or after \( t \), is in \( V \). Otherwise, take \( V = A(0) \), which is broad, and choose some positive \( t \). Then \( A(0)_{-t} = \{ \epsilon \} \). But \( \{ \epsilon \} \) is not broad.

The proof follows by showing that for any \( v \), \( v \in V_{-t} \), \( \beta(v) \in V_{-t} \). Proof is by induction on the length of \( v \).

- \( v = \epsilon \): We have to show that \( \beta(\epsilon) = A(0) \subseteq V_{-t} \). Given,
  \[
  \begin{align*}
  u_t & \in V \\
  \Rightarrow & \{ V \text{ is broad} \} \\
  \beta(u_t) & \subseteq V \\
  \Rightarrow & \{ A(t) \subseteq \beta(u_t), \text{ from definition of } \beta(\cdot) \} \\
  A(t) & \subseteq V \\
  \Rightarrow & \{ \text{definition of } V_{-t} \} \\
  A(t)_{-t} & \subseteq V_{-t} \\
  \Rightarrow & \{ A(t)_{-t} = A(0), \text{ from Observation 18, page 73} \} \\
  A(0) & \subseteq V_{-t} 
  \end{align*}
  \]

- \( v \neq \epsilon \):
  \[
  \begin{align*}
  v & \in V_{-t} \\
  \Rightarrow & \{ \text{definition of } V_{-t} \} \\
  v_t & \in V \\
  \Rightarrow & \{ V \text{ is broad} \} \\
  \beta(v_t) & \in V \\
  \Rightarrow & \{ \text{from Lemma 20, page 70, } \beta(v)_t \subseteq \beta(v_t) \} \\
  \beta(v) & \in V \\
  \Rightarrow & \{ \text{definition of } V_{-t} \} \\
  \beta(v) & \in V_{-t} 
  \end{align*}
  \]

3.1.3 Base Expressions are Broad

Lemma 26 \( A(t) \subseteq \llbracket M(x) \rrbracket \), for any time \( t \).

Proof: Consider \( p \in A(t) \). If \( p \) does not contain a substitution to \( x \), then \( p \in \llbracket M(x) \rrbracket \) follows easily from Theorem 8 on page 32. Otherwise, let \( p = q(t, [m/x])r \), where \( q \) has no substitution to \( x \). By Theorem 8 it suffices to show that \( p \in D(t) \setminus x \cdot (t, [m/x]) \cdot \llbracket M(m) \rrbracket_t \). Since \( A(t) \subseteq D(t) \) by Observation 7 on page 9, we have \( A(t) \setminus x \subseteq D(t) \setminus x \). Since \( q \in A(t) \setminus x \) it follows that \( q \in D(t) \setminus x \).

And \( r \in A(t) \subseteq D(t) \), and so \( r \in \llbracket M(m) \rrbracket_t \) by Theorem 8.

Lemma 27 Base expressions are broad.

Proof:
• \( \beta([0]) \subseteq [0] \):

\[
\beta([0]) = \{ \text{Theorem 8 on page 32} \}
\]

\[
= \{ \beta() \text{ is coercive} \}
\]

\[
(\cup t : t \in T : D(t)) \subseteq \{ \text{Lemma 17 on page 68} \}
\]

\[
= \{ \text{Theorem 8 on page 32} \}
\]

\[
\beta([\ldots k]) \subseteq [\ldots k]: \text{We show that, for } p \in [\ldots k], \beta(p) \subseteq [\ldots k]. \text{ By Theorem 8 on page 32, either } p \in [0] \text{ or, for some time } t \text{ and value } m, p \in D(t) \cdot (t, m) \cdot [0]. \text{ The first case follows from the proof above that } \beta([0]) \subseteq [0]. \text{ Otherwise it suffices to consider } p = q(t, !m)r, \text{ where } q \in D(t) \text{ and } r \in [0]; \text{ the other cases follow from Corollary 3 on page 69.}
\]

Suppose \( q = \epsilon \).

\[
\beta((t, !m)r_t)
\]

\[
= \{ \text{definition of } \beta() \}
\]

\[
A(t) \cup (t, !m)\beta(r_t)
\]

\[
\subseteq \{ r \in [0] \text{ by assumption} \}
\]

\[
A(t) \cup (t, !m)\beta([0])_t
\]

\[
\subseteq \{ \text{above} \}
\]

\[
A(t) \cup (t, !m)[0]_t
\]

\[
\subseteq \{ A(t) \subseteq D(t) \text{ by Observation 7 on page 9} \}
\]

\[
D(t) \cup (t, !m)[0]_t
\]

\[
\subseteq \{ D(t) \subseteq [\ldots k] \text{ and } (t, !m)[0]_t \subseteq [\ldots k] \text{ by Theorem 8 on page 32} \}
\]

\[
\subseteq [\ldots k]
\]

Otherwise \( q = aq_s \) and \( p = ap'_s \).

\[
\beta(ap'_s)
\]

\[
= \{ \text{definition of } \beta() \}
\]

\[
A(s) \cup a\beta(p'_s)
\]

\[
\subseteq \{ \text{induction on } p' \in [\ldots k] \}
\]

\[
A(s) \cup a[\ldots k]
\]

\[
\subseteq \{ A(s) \subseteq D(s) \}
\]

\[
D(s) \cup a[\ldots k]
\]

\[
\subseteq \{ D(s) \subseteq [\ldots k] \text{ and } a[\ldots k], \subseteq [\ldots k] \text{ by Theorem 8 on page 32} \}
\]

\[
\subseteq [\ldots k]
\]

• \( \beta([M(m)]) \subseteq [M(m)] \): \text{We show that, for } p \in [M(m)], \beta(p) \subseteq [M(m)]. \text{ By Theorem 8 on page 32, it suffices to consider } p = q(0, \tau)r \in D(0) \cdot (0, \tau) \cdot [\ldots k], \text{ for some } k \in \Sigma(M, m) \text{ where } q \in D(0) \text{ and } r \in [\ldots k]; \text{ the other cases follow from Corollary 3 on page 69.}
Suppose $q = \epsilon$.

$$\beta((0, \tau)r)$$

$$= \{\text{definition of } \beta()\}$$

$$A(0) \cup (0, \tau)\beta(r)$$

$$\subseteq \{r \in [\alpha k], \text{ which is broad}\}$$

$$A(0) \cup (0, \tau)\alpha k$$

$$\subseteq \{A(0) \subseteq D(0) \text{ by Observation 7 on page 9}\}$$

$$D(0) \cup (0, \tau)\alpha k$$

$$\subseteq \{D(0) \subseteq \llbracket M(m) \rrbracket \text{ and } (0, \tau)\alpha k \subseteq \llbracket M(m) \rrbracket \} \text{ by Theorem 8 on page 32}\}$$

$$\llbracket M(m) \rrbracket$$

Otherwise $q = aq'$ and $p = ap'$, where $a.time = 0$.

$$\beta(ap')$$

$$= \{\text{definition of } \beta()\}$$

$$A(0) \cup a\beta(p')$$

$$\subseteq \{\text{induction on } p' \in \llbracket M(m) \rrbracket\}$$

$$A(0) \cup a\llbracket M(m) \rrbracket$$

$$\subseteq \{A(0) \subseteq D(0) \text{ by Observation 7 on page 9}\}$$

$$D(0) \cup a\llbracket M(m) \rrbracket$$

$$\subseteq \{D(0) \subseteq \llbracket M(m) \rrbracket \text{ and } a \subseteq \llbracket M(m) \rrbracket \text{ by Theorem 8 on page 32}\}$$

$$\llbracket M(m) \rrbracket$$

$$\bullet \beta(\llbracket M(x) \rrbracket) \subseteq \llbracket M(x) \rrbracket: \text{ Consider } p \in \llbracket M(x) \rrbracket. \text{ By Theorem 8 on page 8, it suffices to consider } p = q(t, [m/x])r_t, \text{ where } q \in D(t) \setminus x \text{ and } r \in \llbracket M(m) \rrbracket; \text{ the other cases follow from Corollary 3 on page 69.}$$

Suppose $q = \epsilon$.

$$\beta((t, [m/x])r_t)$$

$$= \{\text{definition of } \beta()\}$$

$$A(t) \cup (t, [m/x])\beta(r_t)$$

$$\subseteq \{r \in \llbracket M(m) \rrbracket, \beta(M(m)) \subseteq M(m) \text{ as above}\}$$

$$A(t) \cup (t, [m/x])\llbracket M(m) \rrbracket$$

$$\subseteq \{A(t) \subseteq \llbracket M(x) \rrbracket \text{ by Lemma 26 on page 73, } (t, [m/x])\llbracket M(m) \rrbracket_t \subseteq \llbracket M(x) \rrbracket \text{ by Theorem 8}\}$$

$$\llbracket M(x) \rrbracket$$

Otherwise $q = aq'_s$ and $p = ap'_s$.

$$\beta(ap'_s)$$

$$= \{\text{definition of } \beta()\}$$

$$A(s) \cup a\beta(p'_s)$$

$$\subseteq \{\text{induction on } p'_s \in \llbracket M(x) \rrbracket\}$$

$$A(s) \cup a\llbracket M(x) \rrbracket$$

$$\subseteq \{A(s) \subseteq \llbracket M(x) \rrbracket \text{ by Lemma 26 on page 73, } a\llbracket M(x) \rrbracket_s \subseteq \llbracket M(x) \rrbracket \text{ by Theorem 8}\}$$

$$\llbracket M(x) \rrbracket$$
3.2 Symmetric Composition

We use the definition of $|$ applied to sets, as given in Section 2.2.1, page 35.

3.2.1 Preliminary Results

We use the following algebraic properties of guarded sets.

**Observation 19**
1. $[\text{true} \rightarrow S] = S$, $[\text{false} \rightarrow S] = \{\epsilon\}$.
2. Given that $\epsilon \in S'$, $[\text{false} \rightarrow S] \cup [p' \rightarrow S'] = [p' \rightarrow S']$.
3. Given that $S \subseteq S'$, $[p \rightarrow S] \subseteq [p \rightarrow S']$.
4. Suppose $f(\epsilon) = \{\epsilon\}$. Then, $f[p \rightarrow S] = [p \rightarrow f(S)]$. Thus,

$$\beta([p \rightarrow S]) = [p \rightarrow \beta(S)]$$

**Lemma 28** $|$ is commutative.

Proof: Proof is by induction on the combined length of the arguments. If either $u$ or $v$ is empty, then $u \mid v = \{\epsilon\} = v \mid u$. Now, we show that $bv \mid au = au \mid bv$.

$$bv \mid au = \{\text{definition of } \mid \} \{b \equiv a \mid b(v \mid u)\] \cup [b \preceq a \rightarrow b(v \mid au)] \cup [a \preceq b \rightarrow a(bv \mid u)]
= \{b \equiv a \equiv a \equiv b\} \{b \equiv b \rightarrow a(v \mid u)\] \cup [b \preceq a \rightarrow b(v \mid au)] \cup [a \preceq b \rightarrow a(bv \mid u)]
= \{\text{induction: } v \mid u = u \mid v, v \mid au = au \mid v, bv \mid u = u \mid bv\} \{a \equiv b \rightarrow a(u \mid v)\] \cup [b \preceq a \rightarrow b(au \mid v)] \cup [a \preceq b \rightarrow a(u \mid bv)]
= \{\text{rearranging the terms around set union}\} au \mid bv

**Observation 20** $\epsilon \in (u \mid v)$, for any $u$ and $v$.

Proof: If either $u$ or $v$ is empty, the result follows from definition. For $au \mid bv$, $\neg((a \equiv b) \land (a \preceq b))$; so, at least one of these conditions is false, and the corresponding guarded set contributes $\{\epsilon\}$.

We can prove a much stronger result, that $u \mid v$ is (non-empty and) prefix-closed. We do not need this result in developing the theory.

**Observation 21** $[\text{false} \rightarrow S] \cup [p \rightarrow u \mid v] = [p \rightarrow u \mid v]$

Proof: This follows from $[\text{false} \rightarrow S] = \{\epsilon\}$ and that $u \mid v$ includes $\epsilon$, from (Observation 20, page 76).

This observation allows us to simplify expressions by dropping terms whose guards are false, provided that one of the sets that is retained contains $\epsilon$. 
3.2.2 Symmetric Composition Preserves Breadth

We show that for broad sets $U$ and $V$, $U \mid V$ is broad.

**Lemma 29** Given $s \leq t$, $A(s) \mid A(t) = A(s)$.

Proof: Let $u$ be a sequence of substitutions all at some time $r$ and $v$ a sequence of substitutions all at time $r'$. Prove by induction that $u \mid v$ is the set of common prefixes of $u$ and $v$. Considering the sequences in $A(s)$ and $A(t)$, $A(s) \mid A(t) = A(s)$.

**Corollary 5** $A(t) \mid A(t) = A(t)$.

**Corollary 6** Let $U$ and $V$ be broad sets, and $a \in U$ and $b \in V$, for some $a$ and $b$. Then, $A(t) \subseteq U \mid V$.

Proof:

$$a_t \in U \Rightarrow \{U \text{ broad}\}$$
$$\beta(a_t) \subseteq U$$
$$\Rightarrow \{A(t) \subseteq \beta(a_t), \text{ by the definition of } \beta()\}$$
$$A(t) \subseteq U$$
$$\Rightarrow \{\text{similarly, } A(t) \subseteq V\}$$
$$A(t) \subseteq U, A(t) \subseteq V$$
$$\Rightarrow \{\text{apply } \mid \}$$
$$A(t) \mid A(t) \subseteq U \mid V$$

$$\Rightarrow \{A(t) \mid A(t) = A(t), \text{ from Corollary 5, page 77}\}$$
$$A(t) \subseteq U \mid V$$

**Lemma 30** $(U \mid V) \backslash a = U \backslash a \mid V \backslash a$, where $a$ is a substitution.

Proof: Since $\backslash a$ is coercive, it is sufficient to prove that $(u \mid v) \backslash a = u \backslash a \mid v \backslash a$.

If both $u$ and $v$ do not start with $a$, then $u \mid v$ does not start with $a$, from the definition of $\mid$ and that $a$ is a substitution. Then $(u \mid v) \backslash a = \phi$. Also, at least one of $u \backslash a$ and $v \backslash a$ is $\phi$, so $u \backslash a \mid v \backslash a = \phi$.

If both $u$ and $v$ start with $a$, then $u = ap_t$ and $v = aq_t$, where $t = a.time$.

$$\begin{align*}
(u \mid v) \backslash a &= \{u = ap_t \text{ and } v = aq_t\} \\
(a p_t \mid a q_t) \backslash a &= \{\text{from definition of } \mid, \text{ } a p_t \mid a q_t = a(p_t \mid q_t) = a(p \mid q)_t\} \\
(a(p \mid q)_t) \backslash a &= \{\text{definition of } \backslash a\} \\
p \mid q &= \{u = ap_t \text{ and } v = aq_t, \text{ So, } \{p\} = u \backslash a \text{ and } \{q\} = v \backslash a\} \\
&u \backslash a \mid v \backslash a
\end{align*}$$
Lemma 31  Given $U$ and $V$ broad, a a base event at $t$, $a \in U$, and $b_t \in V$ for some event $b$. Then, $u \in U \setminus a \mid V_{-t} \Rightarrow au_t \in U \mid V$.

Proof: Proof is by case analysis on $u$.

- $u = \epsilon$  We have to show that $a \in U \mid V$.
  
  $a \in U$
  
  $\Rightarrow \{b_t \in V\}$
  
  $a \mid b_t \subseteq U \mid V$
  
  $\Rightarrow \{\text{from definition of } \mid, [a \preceq b_t \rightarrow a(\epsilon \mid b_t)] \subseteq a \mid b_t\}$
  
  $[a \preceq b_t \rightarrow a(\epsilon \mid b_t)] \subseteq U \mid V$
  
  $\Rightarrow \{a \preceq b_t \text{ holds given that } a \text{ is base event at } t; \epsilon \mid b_t = \{\epsilon\}\}$
  
  $a \in U \mid V$

- $u \neq \epsilon$  Given $u \in U \setminus a \mid V_{-t}$, $u \in p \mid q$, where $p \in U \setminus a$, and $q \in V_{-t}$. Neither $p$ nor $q$ is empty because $u$ in non-empty.

  $u \in p \mid q$
  
  $\Rightarrow \{\text{apply time-shift}\}$
  
  $u_t \in p_t \mid q_t$
  
  $\Rightarrow \{\text{concatenation}\}$
  
  $au_t \in a(p_t \mid q_t)$
  
  $\Rightarrow \{\text{let } q = cr. \text{ Then, } a(p_t \mid q_t) = \{a \preceq c_t\} \text{ and } a(p_t \mid c_t r_t) = a(p_t \mid q_t)\}$
  
  $au_t \in a(p_t \mid q_t)$
  
  $\Rightarrow \{p \in U \setminus a \Rightarrow ap_t \in U; q \in V_{-t} \Rightarrow q_t \in V\}$
  
  $au_t \in U \mid V$

Theorem 17  Given that $U$ and $V$ are broad, $U \mid V$ is broad.

Proof: If either of $U$ or $V$ is the empty set, then $U \mid V$ is the empty set, which is broad. Now, assume that both sets are non-empty. We show for any $u$ and $v$, where $u \in U$ and $v \in V$, that $\beta(u \mid v) \subseteq U \mid V$. Then from Corollary 4, page 69, $U \mid V$ is broad.

The proof of $\beta(u \mid v) \subseteq U \mid V$ is by induction on the combined length of $u$ and $v$.

- $u$ or $v$ is empty: Then $u \mid v = \{\epsilon\}$, and we have to show $\beta(\epsilon) \subseteq U \mid V$. From Observation 17, page 67, using $\beta(U) = U$,

  $\beta(\epsilon) \subseteq U$
  
  $\Rightarrow \{\text{similarly with } V\}$
  
  $\beta(\epsilon) \subseteq U, \beta(\epsilon) \subseteq V$
  
  $\Rightarrow \{\text{taking } \mid \} \}$
  
  $\beta(\epsilon) \mid \beta(\epsilon) \subseteq U \mid V$
  
  $\Rightarrow \{\beta(\epsilon) \mid \beta(\epsilon) = \beta(\epsilon), \text{ from Corollary 5, page 77}\}$
  
  $\beta(\epsilon) \subseteq U \mid V$
• $au_t \in U$ and $av_t \in V$, where $a \text{.time} = t$ and $a \simeq b$: we show $\beta(au_t \mid av_t) \subseteq U \mid V$.

$$\begin{align*}
\beta(au_t \mid av_t) \\
= \{\text{definition of } \mid \} \\
\beta(a(u \mid v)_t) \\
= \{\text{definition of } \beta()\} \\
A(t) \cup a\beta(u \mid v)_t
\end{align*}$$

Now, $A(t) \subseteq U \mid V$ follows from Corollary 6, page 77, because $a \in U$, $a \in V$, and $a\text{.time} = t$. We show $a\beta(u \mid v)_t \subseteq U \mid V$.

$$\begin{align*}
au_t \in U, av_t \in V \\
\Rightarrow \{\text{definition}\} \\
u \in U \setminus a, v \in V \setminus a \\
\Rightarrow \{\text{apply } \mid \} \\
u \mid v \subseteq U \setminus a \mid V \setminus a \\
\Rightarrow \{U \setminus a \text{ and } V \setminus a \text{ are broad from Lemma } 24, \text{ page } 72; \text{ apply induction}\} \\
\beta(u \mid v) \subseteq U \setminus a \mid V \setminus a \\
\Rightarrow \{U \setminus a \mid V \setminus a = (U \mid V) \setminus a, \text{ from Lemma } 30, \text{ page } 77\} \\
\beta(u \mid v) \subseteq (U \mid V) \setminus a \\
\Rightarrow \{\text{definition of } (U \mid V) \setminus a\} \\
a\beta(u \mid v)_t \subseteq U \mid V
\end{align*}$$

• $au \in U$ and $bv \in V$, where $\neg(a \simeq b)$: we show $\beta(au \mid bv) \subseteq U \mid V$.

$$\begin{align*}
\beta(au \mid bv) \\
= \{\text{definition of } \mid \text{ given } \neg(a \simeq b)\} \\
\beta([a \preceq b \rightarrow a(u \mid bv)] \cup [b \preceq a \rightarrow b(au \mid v)] \cup \{\epsilon\}) \\
= \{\beta() \text{ distributes over set union and guarded sets}\} \\
[a \preceq b \rightarrow \beta(a(u \mid bv))] \cup [b \preceq a \rightarrow \beta(b(au \mid v))] \cup \beta(\epsilon)
\end{align*}$$

In the earlier proof with $u = \epsilon$ or $v = \epsilon$, we showed $\beta(\epsilon) \subseteq U \mid V$. The remaining two terms are symmetric in $au$ and $bv$, using commutativity of $\mid$. Therefore, it is sufficient to show that $[a \preceq b \rightarrow \beta(a(u \mid bv))] \subseteq U \mid V$.

If $\neg(a \simeq b)$, then $[a \preceq b \rightarrow \beta(a(u \mid bv))] = \{\epsilon\}$, which is trivially in $U \mid V$. Assume $a \preceq b$. We rename the terms as $au_t$ and $bv_t$. Our goal is to show $\beta(a(u_t \mid bv_t)) \subseteq U \mid V$, where $t = a\text{.time}$, $au_t \in U$, $bv_t \in V$.

$$\begin{align*}
\beta(a(u_t \mid bv_t)) \\
= \{\text{distribute time-shift}\} \\
\beta(a(u \mid bv)_t) \\
= \{\text{definition of } \beta()\} \\
A(t) \cup a\beta(u \mid bv)_t
\end{align*}$$

From Corollary 6, page 77, $A(t) \subseteq U \mid V$. The remaining task is to show $a\beta(u \mid bv)_t \subseteq U \mid V$. 
CHAPTER 3. BREADTH AND TRACE PRESERVATION

\[ au \in U, \text{ and } bv \in V \]
\[ \Rightarrow \{ \text{definitions} \} \]
\[ u \in U \setminus a, bv \in V_{-t} \]
\[ \Rightarrow \{ U \setminus a \text{ is broad, from Lemma 24, page 72,} \]
\[ \text{given } bv \in V, \text{ and } V \text{ broad, from Lemma 25, page 73, } V_{-t} \text{ is broad,} \]
\[ \text{apply induction (combined length of } u \text{ and } bv \text{ is less than } au \text{ and } bv) \}
\[ \beta(u \parallel bv) \subseteq U \setminus a \mid V_{-t} \]
\[ \Rightarrow \{ \text{Apply Lemma 31, page 78, for each element in } \beta(u \parallel bv): \}
\[ a \text{ is base, from } a \preceq b, \]
\[ a \in U, \text{ from } au \in U, \text{ and } U \text{ prefix closed,} \]
\[ bv \in V, \text{ from } bv \in V, \text{ and } V \text{ prefix closed} \]
\[ \alpha(bv) \subseteq U \mid V \]

3.2.3 Symmetric Composition Preserves Traces

We show that for broad sets \( U \) and \( V, \overline{U \parallel V} = U \parallel \overline{V} \).

Lemma 32 Let \( u \) and \( v \) be visible. Then, \( \overline{u \parallel v} = \overline{u} \mid \overline{v} \).

Proof: We prove the result by induction on the combined length of \( u \) and \( v \).

- If either \( u \) or \( v \) is \( \epsilon \), both sides are \{ \epsilon \}, from the definition. Next, we take \( au \) and \( bv \) which are both visible, and show that \( \overline{au \parallel bv} = \overline{au} \mid \overline{bv} \). Note that \( u \) and \( v \) are visible, given \( au \) and \( bv \) are visible; either or both of \( u \) and \( v \) may be \( \epsilon \).

\[
\overline{au \parallel bv} = \{ \text{definition of } au \parallel bv \}
\]
\[
\begin{align*}
&= \{ \text{distribute trace over set union, guarded sets and concatenation} \} \\
&= \{ \text{induction. Note that } au, bv, u \text{ and } v \text{ are visible} \} \\
&= \{ \text{distribute trace over concatenation} \} \\
&\Rightarrow \{ \text{condition in at least one of the guarded sets applies and } \epsilon \in \overline{u} \mid \overline{v} \}
\end{align*}
\]

We show that \( \overline{au \parallel bv} = \overline{au} \mid \overline{bv} \) for each of these cases: (1) \( a, b = \tau, \tau \), (2) \( a \neq \tau \) and \( b \neq \tau \) (3) \( a \neq \tau \) and \( b = \tau \), (4) \( a = \tau \) and \( b \neq \tau \).

Case 1) \( a, b = \tau, \tau \): \( \overline{au} \mid \overline{bv} = \overline{au} \mid \overline{v} \)
\[
\overline{au \parallel bv} = \{ \text{use } \neg(a \preceq b) \text{ in } (*) \text{ since } a, b = \tau, \tau; \text{ apply Observation 21, page 76} \}
\]
\[
\begin{align*}
&= \{ \text{condition in at least one of the guarded sets applies and } \epsilon \in \overline{u} \mid \overline{v} \}
\end{align*}
\]
CHAPTER 3. BREADTH AND TRACE PRESERVATION

Case 2) $a \neq \tau$ and $b \neq \tau$:

$$\bar{a}\pi | \bar{b}\nu = \{\text{distribute trace over concatenation}\}$$
$$a\pi | b\nu$$
$$= \{\text{apply definition}\}$$
$$[a \asymp b \rightarrow a(\pi | \nu)] \cup [a \leq b \rightarrow a(\pi | b\nu)] \cup [b \leq a \rightarrow b(a\pi | \nu)]$$

And, this matches (*) given $\pi = a$ and $\nu = b$.

Case 3) $a \neq \tau$ and $b = \tau$:

Then, $\pi|\nu = a\pi | \nu$  \(\text{(L)}\)

And,

$$\bar{a}\mu | \bar{b}\nu = \{\text{use } \neg(a \asymp b) \text{ in (*) since } b = \tau\}$$
$$[a \leq b \rightarrow a(\pi | \nu)] \cup [b \leq a \rightarrow (a\pi | \nu)] \cup \{\epsilon\}$$
$$= \{\text{drop } \{\epsilon\}, \text{ using Observation 21, page 76, on the second term}\}$$
$$[a \leq b \rightarrow a(\pi | \nu)] \cup [b \leq a \rightarrow (a\pi | \nu)] \cup \{\epsilon\}$$  \(\text{(R)}\)

Since $bv$ is visible and $b = \tau$, $v$ is non-empty and visible. Therefore $\nu \neq \epsilon$.

Let $c$ be the first event of $v$. Then, considering $bv$, $b.time \leq c.time$. We show (L) = (R) for 3 cases: (1) $a.time < b.time$, (2) $a.time = b.time$, and (3) $a.time > b.time$.

Case 3.1) $a.time < b.time$:

$$\text{(L)} = a\pi | \nu$$
$$= \{a.time < b.time \leq c.time. \text{ Hence, } \neg(a \asymp c) \text{ and } \neg(c \leq a)\}$$
$$\text{Apply definition of } | \}$$
$$[a \leq c \rightarrow a(\pi | \nu)] \cup \{\epsilon\}$$
$$= \{a.time \leq c.time \text{ means } a \leq c \equiv a \text{ is base }\}$$
$$[a \text{ is base } \rightarrow a(\pi | \nu)] \cup \{\epsilon\}$$

And,

$$\text{(R)} = [a \leq b \rightarrow a(\pi | \nu)] \cup [b \leq a \rightarrow (a\pi | \nu)]$$
$$= \{a.time < b.time. \text{ So, } \neg(b \leq a). \text{ Also, } a \leq b \equiv a \text{ is base }\}$$
$$[a \text{ is base } \rightarrow a(\pi | \nu)] \cup \{\epsilon\}$$
$$= \{\text{from above derivation}\}$$
$$\text{(L)}$$

Case 3.2) $a.time = b.time$:

$$\text{(R)} = [a \leq b \rightarrow a(\pi | \nu)] \cup [b \leq a \rightarrow (a\pi | \nu)]$$
$$= \{\text{from } a.time = b.time, \text{ a } \leq b \equiv a \text{ is base and } b \leq a \text{ holds given } b = \tau\}$$
\[ a \text{ is base } \Rightarrow a(\overline{u} \mid \overline{v}) \cup (a\overline{u} \mid \overline{v}) \]

Case 1) \(a\) is not base:

\[ a \text{ is base } \Rightarrow a(\overline{u} \mid \overline{v}) \cup (a\overline{u} \mid \overline{v}) = \{\epsilon\} \cup (a\overline{u} \mid \overline{v}) = (a\overline{u} \mid \overline{v}) \]

If \(a\) is base:

\[ a \text{ is base } \Rightarrow a(\overline{u} \mid \overline{v}) \cup (a\overline{u} \mid \overline{v}) = a(\overline{u} \mid \overline{v}) \cup (a\overline{u} \mid \overline{v}) \]

\(a.time \leq c.time\) implies, from definition, \(a\overline{u} \mid \overline{v} \supseteq a(\overline{u} \mid \overline{v})\)

In all cases, \(a \text{ is base } \Rightarrow a(\overline{u} \mid \overline{v}) \cup (a\overline{u} \mid \overline{v}) = (a\overline{u} \mid \overline{v})\)

\(\overline{a\overline{u} \mid \overline{v}} = (L)\)

Case 2) \(b.time > c.time\):

\((R) = [a \leq b \rightarrow a(\overline{u} \mid \overline{v})] \cup [b \leq a \rightarrow (a\overline{u} \mid \overline{v})]\)

\[ = \{\neg(a \leq b) \text{ and } b \leq a \text{ hold}\} \]
\[ (a\overline{u} \mid \overline{v}) \cup \{\epsilon\} \]

\[ = \{\text{from Observation 20, page 76, } \{\epsilon\} \subseteq (a\overline{u} \mid \overline{v})\} \]
\[ a\overline{u} \mid \overline{v} = (L) \]

**Lemma 33** Let \(u\) and \(v\) be visible. Then, \(\overline{u} \mid \overline{v} = \overline{\alpha u \mid v}\)

Proof: We prove the result by induction on the combined length of \(u\) and \(v\).

If either \(u\) or \(v\) is \(\epsilon\), both sides are \(\{\epsilon\}\), from the definition. Next, we take \(au\) and \(bv\) which are both visible, and show that \(\overline{au \mid bv} = \overline{\alpha u \mid v}\). Note that \(u\) and \(v\) are visible, given \(au\) and \(bv\) are visible; either or both of \(u\) and \(v\) may be \(\epsilon\).

\[ \overline{au \mid bv} \]
\[ = \{\text{definition of } au \mid bv\} \]
\[ = [a \leq b \rightarrow a(\overline{u} \mid \overline{v})] \cup [b \leq a \rightarrow (a\overline{u} \mid \overline{v})] \]
\[ = \{\text{induction. Note that } au, bv, u \text{ and } v \text{ are visible}\} \]
\[ = [a \leq b \rightarrow (\overline{\alpha u} \mid \overline{v})] \cup [b \leq a \rightarrow (a\overline{u} \mid \overline{v})] \]

We show that \(\overline{au \mid bv} = \overline{\alpha u \mid v}\) for each of these cases: (1) \(a \neq \tau\) and \(b \neq \tau\) (2) \(b = \tau\), (3) and \(a = \tau\).

Case 1) \(a \neq \tau\) and \(b \neq \tau\):

\[ \overline{\alpha u \mid v} \]
\[ = \{\text{apply definition}\} \]
\[ = a\overline{u} \mid \overline{v} \]

And, this matches \((R)\) given \(\overline{a\overline{u} \mid \overline{v}} = a\overline{u} \mid \overline{v}\).

Case 2) \(b = \tau\): Since \(bv\) is visible and \(b = \tau\), it follows that \(v\) is non-empty and visible. Let \(v = cv'\). Then, considering \(bv\), \(b.time \leq c.time\).
CHAPTER 3. BREADTH AND TRACE PRESERVATION

\[
\overline{au} \mid \overline{bv} = \{ b = \tau \} \quad \overline{at} \mid \overline{v} \quad \text{(L1)}
\]

\[
\overline{au} \mid \overline{v} = \{ \text{induction} \} \quad \overline{au} \mid \overline{v} \quad [a \simeq c \rightarrow \overline{at} \mid \overline{v}] \cup [a \preceq c \rightarrow \overline{at} \mid \overline{v}] \cup [c \preceq a \rightarrow \overline{ct} \mid \overline{v}'] \quad \text{(L2)}
\]

We consider two cases and show that \( \text{(R)} \) equals \( \text{(L1)} \) or \( \text{(L2)} \) in each case.

Case 2.1) \( b.\text{time} \leq a.\text{time} \):

\[
\text{(R)} = \{ \neg(a \simeq b) \text{ from } b = \tau, \text{ and } b \preceq a \text{ using } b.\text{time} \leq a.\text{time} \}
\]

\[
[a \preceq b \rightarrow \overline{at} \mid \overline{v}] \cup \{a \} \cup \{\epsilon\}
\]

\[
\text{so, } [a \preceq b \rightarrow \overline{at} \mid \overline{v}] \subseteq [a \preceq c \rightarrow \overline{at} \mid \overline{v}] \cup \{\epsilon\};
\]

\[
\text{from } (\text{L1}, \text{L2}), [a \preceq c \rightarrow \overline{at} \mid \overline{v}] \subseteq \overline{at} \mid \overline{v};
\]

\[
\text{so, } [a \preceq b \rightarrow \overline{at} \mid \overline{v}] \subseteq \overline{at} \mid \overline{v} \cup \{\epsilon\}
\]

\[
\overline{at} \mid \overline{v} \cup \{\epsilon\}
\]

\[
\text{from Observation 20, page 76, } \{\epsilon\} \subseteq \overline{at} \mid \overline{v}
\]

\[
\text{L1}
\]

Case 2.2) \( \neg(b.\text{time} \leq a.\text{time}) \):

\[
\text{(R)} = \{ \text{given } a.\text{time} < b.\text{time}. \text{ So, } \neg(a \simeq b), \neg(b \preceq a) \}
\]

\[
[a \preceq b \rightarrow \overline{at} \mid \overline{v}] \cup \{\epsilon\}
\]

\[
\text{so, } [a \preceq b \rightarrow \overline{at} \mid \overline{v}] \subseteq [a \preceq c \rightarrow \overline{at} \mid \overline{v}] \cup \{\epsilon\};
\]

\[
\text{from } (\text{L1}, \text{L2}), [a \preceq c \rightarrow \overline{at} \mid \overline{v}] \subseteq \overline{at} \mid \overline{v};
\]

\[
\text{so, } [a \preceq b \rightarrow \overline{at} \mid \overline{v}] \subseteq \overline{at} \mid \overline{v} \cup \{\epsilon\}
\]

\[
\overline{at} \mid \overline{v} \cup \{\epsilon\}
\]

\[
\text{L2}
\]

Case 3) \( a = \tau \): Similar to case (2).

Lemma 34 \( u \mid v \subseteq uc \mid vd \), for any events \( c \) and \( d \).

Proof: We prove \( u \mid v \subseteq uc \mid vd \). Similarly, using commutativity, it can be shown that \( u \mid v \subseteq u \mid vd \). Then, \( u \mid v \subseteq uc \mid vd \).

Now, we prove \( u \mid v \subseteq uc \mid v \) by induction on the combined length of \( u \) and \( v \).

- \( u = \epsilon \) or \( v = \epsilon \): Left side is \( \{\epsilon\} \), and the right side, being a merge, contains \( \epsilon \), from Observation 20, page 76.

- \( au \mid bv \subseteq auc \mid bv \):
CHAPTER 3. BREADTH AND TRACE PRESERVATION

84

\( au \mid bv \)

\( = \{ \text{definition of } \mid \} \)

\( [a \simeq b \rightarrow a(u \mid v)] \cup [a \preceq b \rightarrow a(u \mid bv)] \cup [b \preceq a \rightarrow b(au \mid v)] \)

\( \subseteq \{ \text{induction: } u \mid v \subseteq uc \mid v; \text{ similarly for the other terms} \} \)

\( [a \simeq b \rightarrow a(uc \mid v)] \cup [a \preceq b \rightarrow a(uc \mid bv)] \cup [b \preceq a \rightarrow b(uc \mid v)] \)

\( = \{ \text{definition of } \mid \} \)

\( auc \mid bv \)

**Theorem 18** Let \( U \) and \( V \) be broad sets. Then \( U \mid V = U \mid V \).

Proof: The proof is in two parts: \( U \mid V \subseteq U \mid V \), and \( U \mid V \subseteq U \mid V \).

- \( U \mid V \subseteq U \mid V \): For any \( u \) in \( U \) and \( v \) in \( V \). We show \( u \mid v \subseteq U \mid V \).

\( u \mid v \)

\( \subseteq \{ \text{from Lemma 21, page 70, there is } uc \in \beta(u) \subseteq U \text{ and } vd \in V \text{ where } c \text{ and } d \text{ are substitutions} \}

\( = \{ \text{from above} \} \)

\( \subseteq \{ \text{from prefix-closure, } p \in U \text{ and } q \in V \} \)

- \( U \mid V \subseteq U \mid V \): We show that for \( u \) in \( U \) and \( v \) in \( V \), \( u \mid v \subseteq U \mid V \).

Let \( p \) be the longest visible prefix of \( u \) and \( q \) of \( v \). Then \( \pi = p \) and \( \nu = q \).

From prefix-closure, \( p \in U \) and \( q \in V \).

\( \pi \mid \nu \)

\( = \{ \text{from above} \} \)

\( \pi \mid q \)

\( = \{ \text{from Lemma 21, page 70, there is } uc \in \beta(u) \subseteq U \text{ and } vd \in V \text{ where } c \text{ and } d \text{ are substitutions} \}

\( \subseteq \{ \text{from above} \} \)

\( \subseteq \{ \text{from prefix-closure, } p \in U \text{ and } q \in V \} \)

3.3 Sequential Composition

We use the definition of \( \triangleright x \triangleright \) applied to sets, as given in Section 2.2.2, page 36.

3.3.1 Preliminary Results

**Lemma 35** For \( p \) that has no publication and \( V \neq \phi \), \( p \triangleright x \triangleright V = \{ p \} \).

Proof: By induction on the length of \( p \).

**Corollary 7** \( A(t) \triangleright x \triangleright V = A(t) \), for \( V \neq \phi \).

Proof: From above Lemma, because no sequence in \( A(t) \) has a publication.
Notational Simplification Define
\[ \hat{a} = \begin{cases} 
  a & \text{if } a \text{ is not a publication} \\
  (t, \tau) & \text{if } a \text{ is a publication at time } t
\end{cases} \]

Then, replacing \( p \) by \( p_t \) in the definition of sequential composition, and using Lemma 6, page 44, \( ap_t \ >_x V = \hat{a}P_t \), where \( P \) is

\[
(\text{SCD}) \begin{cases} 
  p >_x V & \text{if } c_1(a) \\
  p >_x V' & \text{if } c_2(a) \\
  p >_x V \mid V'' & \text{if } c_3(a)
\end{cases}
\]

3.3.2 Sequential Composition Preserves Breadth

**Theorem 19** For broad sets \( P \) and \( V \), \( P >_x V \) is broad.

**Proof:** We have \( P >_x \phi = \phi \), from Lemma 2.2.4, page 37, and \( \phi \) is broad, vacuously. Assume, henceforth, that \( V \neq \phi \). We prove below that for any sequence \( q \), \( \beta(q) >_x V \) is broad. Then,

\[
P >_x V = \{ \text{\( \beta(P) >_x V \) is broad; so, } P = \beta(P) \}
\]

\[
\beta(P) >_x V = \{ \text{\( \beta(P) = (\cup q \in P: \beta(q)) \)} \\
\uparrow q \in P: \beta(q) >_x V
\]

\[
= \{ \text{distribute } >_x \text{ over set union} \}
\]

Each \( \beta(q) >_x V \) is broad (to be shown) and union of broad sets is broad, from Corollary 4, page 69. Hence the result.

We prove \( \beta(q) >_x V \) is broad by induction on the length of \( q \).

- \( q = \epsilon \):
  \[
  \beta(\epsilon) >_x V = A(0) >_x V = A(0), \text{ from Corollary 7, page 84. And, } A(0) \text{ is broad, from Lemma 16, page 67.}
  \]

- \( q = ap_t \), where \( a.time = t \):
  \[
  \beta(ap_t) >_x V = \{ \text{definition of breadth} \}
  \]

\[
(\beta(t) \cup a\beta(p_t)) >_x V
\]

\[
= \{ \text{distribute } >_x \text{ over set union} \}
\]

\[
(\beta(t) >_x V) \cup (a\beta(p_t) >_x V)
\]

\[
= \{ A(t) >_x V = A(t) \}
\]

\[
A(t) \cup (a\beta(p_t) >_x V)
\]

\[
= \{ \text{replace } p \text{ by } \beta(p) \text{ in (SCD), page 85, to get } U \} \]
\[ A(t) \cup \hat{a}U_t \]
\[(\ast 1) \]
\[= \{ \text{from induction hypothesis} \]
\[= \{ \beta(p) >x> V \text{ is broad, given } V \text{ is broad} \]
\[= \{ \beta(p) >x> V' \text{ is broad, because } V' \text{ is broad from Lemma 24, page 72} \]
\[= \{ \beta(p) >x> V' \mid V'' \text{ is broad, because } V'' \text{ is broad, from Lemma 24, page 72, and} \]
\[= \{ \text{merge of two broad sets is broad, from Theorem 17, page 78;} \]
\[= \{ \text{therefore, } U \text{ is broad; hence } \}
\[= \{ \beta(\hat{a}U_t) \}
\]

It follows from Corollary 4, page 69 Part (1), that \( \beta(ap_t) >x> V \) is broad since it is \( \beta(Q) \), for some \( Q \).

**Lemma 36** \( (U \setminus a >x> V) = (U >x> V) \setminus a \), where \( a \) is an own substitution and \( a.time = t \).

Proof: We show for arbitrary \( u \in U \) that \( (u \setminus a >x> V) = (u >x> V) \setminus a \). The result then follows by coercion.

If \( u = \epsilon \), then
\[ (\epsilon \setminus a >x> V) \]
\[= \{ \text{definition of } \} \]
\[= \{ \epsilon >x> V \} \]
\[= \{ \coersion } \]
\[= \emptyset. \]

and
\[ (\epsilon >x> V) \setminus a \]
\[= \{ \text{definition of } \epsilon >x> V \} \]
\[= \{ \epsilon \} \setminus a \]
\[= \{ \epsilon \} \setminus a \]
\[= \emptyset. \]

Next suppose \( u = bu' \) and \( b \neq a \). Then
\[ (bu' \setminus a >x> V) \]
\[= \{ \text{definition of } \} \]
\[= \{ (\emptyset >x> V), b \neq a \} \]
\[= \{ \coersion } \]
\[= \emptyset. \]

Then if \( b \) is an other-substitution:
\[ (bu' >x> V) \setminus a \]
\[= \{ \text{condition } c_2 \text{ holds} \}
\[= \{ b(u' >x> V \setminus b) \} \setminus a \]
\[= \{ \text{definition of } \} \]
\[= \emptyset. \]
If $b$ is a non-publication base event:

\[(bu' > x > V) \setminus a\]

= \{\text{condition } c_1 \text{ holds}\}

\[(b(u' > x > V)) \setminus a\]

= \{\text{definition of } \setminus, \ b \neq a\}

\[\emptyset.\]

Finally, if $b$ is a publication event \((t, !m)\):

\[((t, !m)u' > x > V) \setminus a\]

= \{\text{condition } c_3 \text{ holds}\}

\[((t, \tau)(u' > x > V \mid (V\setminus a)) \setminus a\]

= \{\text{definition of } \setminus\}

\[\emptyset.\]

Otherwise, suppose $b = a$.

\[(au' \setminus a > x > V)\]

= \{\text{definition of } \setminus\}

\[(u' > x > V)\]

and

\[(au' > x > V) \setminus a\]

= \{\text{condition } c_1 \text{ holds}\}

\[(a(u' > x > V)) \setminus a\]

= \{\text{definition of } \setminus\}

\[(u' > x > V).\]

**Lemma 37** \((U \setminus a > x > V) \setminus a = (U > x > V) \setminus a\), where $a$ is an other substitution and $a.\text{time} = t$.

Proof: We show for arbitrary $u \in U$ that \((u \setminus a > x > V) \setminus a = (u > x > V) \setminus a\). The result then follows by coercion.

If $u = \epsilon$, then

\[\epsilon(a > x > V) \setminus a\]

= \{\text{definition of } \setminus\}

\[\emptyset > x > V \setminus a\]

= \{\text{coersion}\}

\[\emptyset,\]

and

\[(\epsilon > x > V) \setminus a\]

= \{\text{definition of } \epsilon > x > V\}

\[\epsilon \setminus a\]

= \{\text{definition of } \setminus\}

\[\emptyset.\]
Next suppose \( u = bu' \) and \( b \neq a \). Then
\[
(bu'\backslash a > x > V\backslash a)
= \{ \text{definition of } \backslash, \ b \neq a \}
(\emptyset > x > V)
= \{ \text{coersion} \}
\emptyset,
\]

Then if \( b \) is an other-substitution:
\[
(bu' > x > V)\backslash a
= \{ \text{condition } c_2 \text{ holds} \}
(b(u' > x > V\backslash b))\backslash a
= \{ \text{definition of } \backslash, \ b \neq a \}
\emptyset.
\]

If \( b \) is an own-substitution or a non-publication base event:
\[
(bu' > x > V)\backslash a
= \{ \text{condition } c_1 \text{ holds} \}
(b(u' > x > V))\backslash a
= \{ \text{definition of } \backslash, \ b \neq a \}
\emptyset.
\]

Finally, if \( b \) is a publication event \((t, !m)\):
\[
((t, !m)u' > x > V)\backslash a
= \{ \text{condition } c_3 \text{ holds} \}
((t, \tau)((u' > x > V \mid (V\backslash [m/x]))))\backslash a
= \{ \text{definition of } \backslash \}
(\emptyset).
\]

Otherwise, suppose \( b = a \).
\[
(au'\backslash a > x > V\backslash a)
= \{ \text{definition of } \backslash \}
(u' > x > V\backslash a)
\]

and
\[
(au' > x > V)\backslash a
= \{ \text{condition } c_2 \text{ holds} \}
(a(u' > x > V\backslash a))\backslash a
= \{ \text{definition of } \backslash \}
(u' > x > V\backslash a).
\]
3.3.3 Sequential Composition Preserves Traces

**Theorem 20** For broad $P$ and $V$, where $V$ is substitution independent (see page 20), $P \triangleright x \triangleright V = \overline{P \triangleright x \triangleright V}$.

Proof: As in Theorem 19, page 85, it is sufficient to show that for any sequence $q$ and non-empty broad $V$, $\beta(q) \triangleright x \triangleright V = (\beta(q) \triangleright x \triangleright \overline{V})$. Proof is by induction on the length of $q$.

- $q = \epsilon$:
  $\beta(\epsilon) \triangleright x \triangleright V = \{\text{from Corollary 7, page 84}\} \overline{\beta(\epsilon)}$, and
  $(\overline{\beta(\epsilon)} \triangleright x \triangleright V) = \{\beta(\epsilon) = \beta(\epsilon)\} \overline{\beta}(\epsilon) \triangleright x \triangleright \overline{V} = \{\text{Corollary 7, page 84}\} \overline{\beta(\epsilon)}$.

- $q = ap$, where $a.time = t$: We show $\beta(ap) \triangleright x \triangleright V = \overline{\beta(ap)} \triangleright x \triangleright \overline{V}$.
  First, we prove a sublemma.

**Sublemma** Let

$$U = \begin{cases} \beta(p) \triangleright x \triangleright V & \text{if } c_1(a) \\ \beta(p) \triangleright x \triangleright V' & \text{if } c_2(a) \\ \beta(p) \triangleright x \triangleright V | V'' & \text{if } c_3(a) \end{cases}$$

and

$$W = \begin{cases} \overline{\beta(p)} \triangleright x \triangleright V & \text{if } c_1(a) \\ \overline{\beta(p)} \triangleright x \triangleright (V)' & \text{if } c_2(a) \\ \overline{\beta(p)} \triangleright x \triangleright V | (V)' & \text{if } c_3(a) \end{cases}$$

Then $\overline{W} = U$. (*2)

Note: We get $U$ from (SCD), page 85, replacing $p$ by $\beta(p)$. And, $W$ is obtained from $U$ by replacing $\beta(p)$ and $V$ by $\overline{\beta(p)}$ and $\overline{V}$, respectively.

Proof: Observe that

$$\overline{U} = \begin{cases} \overline{\beta(p)} \triangleright x \triangleright V & \text{if } c_1(a) \\ \overline{\beta(p)} \triangleright x \triangleright V' & \text{if } c_2(a) \\ \overline{\beta(p)} \triangleright x \triangleright V | V'' & \text{if } c_3(a) \end{cases}$$

We consider the three cases for $\overline{U}$, as given above, and show that $\overline{U} = \overline{W}$.

(Subcase 1: $c_1(a)$ holds) Applying induction, $\overline{\beta(p)} \triangleright x \triangleright V = \overline{\beta(p)} \triangleright x \triangleright \overline{V}$, or $\overline{U} = \overline{W}$.

(Subcase 2: $c_2(a)$ holds)

$$\overline{U} = \{\text{from the definition of } \overline{U} \text{ in the second case}\}$$
\[
\begin{align*}
\bar{\beta}(p) > x > V' \\
&= \{ \text{\textit{V}'} is broad, from Lemma 24, page 72; apply induction} \}
\bar{\beta}(p) > x > \overline{V'} \\
&= \{ \text{given \textit{V} is substitution independent, from Lemma 3, page 20, } \overline{V'} = \overline{V'} \}
\bar{\beta}(p) > x > \overline{V'} \\
&= \{ \text{definition of } W \}
\end{align*}
\]

(Subcase 3: \(c_3(a) \) holds)

\[
\begin{align*}
\overline{U} \\
&= \{ \text{from the definition of } \overline{U} \text{ in the third case} \}
\bar{\beta}(p) > x > \overline{V} \mid \overline{V''} \\
&= \{ \bar{\beta}(p) > x > \textit{V} \text{ is broad, from Theorem 19, page 85} \}
\bar{\beta}(p) > x > \overline{V} \mid \overline{V''} \\
&= \{ \text{given \textit{V} is substitution independent, from Lemma 3, page 20, } \overline{V''} = \overline{V'} \}
\bar{\beta}(p) > x > \overline{V} \mid \overline{V''} \\
&= \{ \text{induction on the first term} \}
\bar{\beta}(p) > x > \overline{V} \mid \overline{V''} \\
&= \{ \text{definition of } W \}
\overline{W} \\
&= \{ \text{End of Sublemma} \}
\end{align*}
\]

Next, we simplify \(\bar{\beta}(ap_t) > x > \overline{V} \) and \(\bar{\beta}(ap_t) > x > \overline{V} \).

\[
\begin{align*}
\bar{\beta}(ap_t) > x > \overline{V} \\
&= \{ \text{from the derivation in Theorem 19, page 85, see (*)1} \}
\bar{A}(t) \cup a\bar{U}_t, \text{ where } \overline{U} \text{ is as given in the Sublemma} \\
&= \{ \text{distribute trace over set union, concatenation and time-shift;} \}
\bar{A}(t) = A(t); \bar{u} = \bar{\pi} \} \\
A(t) \cup \bar{u}\bar{U}_t \\
&= \{ \text{End of Sublemma} \}
\end{align*}
\]

Next,

\[
\begin{align*}
\bar{\beta}(ap_t) > x > \overline{V} \\
&= \{ \text{expanding } \beta(ap_t) \} \\
\bar{A}(t) \cup a\bar{\beta}(p_t) > x > \overline{V} \\
&= \{ \text{distribute trace over set union; } \bar{A}(t) = A(t) \} \\
(A(t) \cup a\bar{\beta}(p_t)) > x > \overline{V} \\
&= \{ \text{distribute } > x > \text{ over set union; } A(t) > x > \overline{V} = A(t) \} \\
A(t) \cup (a\bar{\beta}(p_t)) > x > \overline{V} \\
&= \{ \text{distribute trace over concatenation and time-shift} \} \\
A(t) \cup ((\bar{\pi} \bar{\beta}(p_t)) > x > \overline{V})
\end{align*}
\]
To complete the proof that $\overline{\beta(ap_t) > x > V} = \overline{\beta(ap_t) > x > V}$, we consider two cases.

Case 1) $a \neq \tau$:

\[
\overline{\beta(ap_t) > x > V} = \{\text{from (*)4}\}
\]
\[
A(t) \cup ((a \beta(p)_t) > x > V)
\]
\[
= \{a \neq \tau, \text{ so } \overline{\alpha} = a\}
\]
\[
A(t) \cup ((\overline{a} \beta(p)_t) > x > \overline{V})
\]
\[
= \{\text{from (SCD), page 85, where } W \text{ is as given in the Sublemma}\}
\]
\[
A(t) \cup \overline{aW_t}
\]
\[
= \{A(t) = A(t), \overline{\alpha} = \overline{a}\}
\]
\[
A(t) \cup \overline{aW_t}
\]
\[
= \{\overline{W} = \overline{U}, \text{ from (*)2}\}
\]
\[
A(t) \cup \overline{aU_t}
\]
\[
= \{\text{from (*)3}\}
\]
\[
\overline{\beta(ap_t) > x > V}
\]

Case 2) $a = \tau$:

\[
\overline{\beta(ap_t) > x > V} = \{\text{from (*)4}\}
\]
\[
A(t) \cup ((\overline{a} \beta(p)_t) > x > V)
\]
\[
= \{A(t) = A(t), \overline{a} = a\}
\]
\[
A(t) \cup \overline{a(p)_t}
\]
\[
= \{\beta(p)_t > x > V = (\overline{\beta(p)} > x > \overline{V})_t, \text{ from Lemma 6, page 44; distribute time-shift over trace}\}
\]
\[
A(t) \cup (\overline{\beta(p)}_t > x > \overline{V})_t
\]
\[
= \{\text{induction}\}
\]
\[
A(t) \cup (\beta(p)_t > x > V)_t
\]
\[
= \{\text{given } a = \tau, \text{ i.e., } c_1(a), \text{ and definition of } U\}
\]
\[
A(t) \cup U_t
\]
\[
= \{a = \tau\}
\]
\[
A(t) \cup \overline{aU_t}
\]
\[
= \{\text{from (*)3}\}
\]
\[
\overline{\beta(ap_t) > x > V}
\]

3.4 Asymmetric Composition

We use the definition of $<x<$ applied to sets, as given in Section 2.2.3, page 36.

3.4.1 Preliminary Results on Constrained Partial Merge

Observation 22 $\epsilon \in (u|v)$, for any $u$ and $v$. 
Proof: If either \( u \) or \( v \) is empty, the result follows from definition. For \( au_\gamma bv \), \( \neg((a \approx_\gamma b) \land (a \preceq b)) \); so, at least one of these conditions is \( false \), and the corresponding guarded set contributes \( \{\epsilon\} \).

**Observation 23** \([false \rightarrow S] \cup [p \rightarrow u_\gamma v] = [p \rightarrow u_\gamma v] \)

Proof: This follows from \([false \rightarrow S] = \{\epsilon\} \) and that \( u_\gamma v \) includes \( \epsilon \), from (Observation 22, page 91).

This observation allows us to simplify expressions by dropping terms whose guards are \( false \), provided that one of the sets that is retained contains \( \epsilon \).

**Lemma 38** \((u_\gamma v)_t = u_t_\gamma v_t\)

Proof: Apply the definition of \( \gamma \) to both sides. Note that \( a_t \approx_\gamma b_t \equiv a \approx_\gamma b \), \( a_t \preceq b_t \equiv a \preceq b \), and \( \approx_\gamma \). The result follows by applying induction on the combined length of \( u \) and \( v \).

**Lemma 39** \((A(t))_\gamma A(t) = A(t)\)

Proof: Proof is by mutual inclusion.

- \((A(t))_\gamma A(t) \subseteq A(t)\): Let \( u \) be a sequence of substitutions, all at time \( r \), and \( v \) be a sequence of substitutions, all at time \( s \). If the base rule is used in forming \( u_\gamma v \), then \( \epsilon \subseteq A(t) \). If the inductive rule is used, then each term starts with an event at \( min(r, s) \). Using induction, it can be shown that \( u_\gamma v \) then contains sequences of substitutions all at time \( min(r, s) \). Therefore, \( u_\gamma v \subseteq A(t) \).

- \((A(t) \subseteq (A(t))_\gamma A(t)\): Let \( p \in A(t) \). We show \( u \in A(t) \) and \( v \in A(t) \), such that \( p \in u_\gamma v \), \( u.time = v.time = p.time \) and either both \( u \) and \( v \) are empty or neither is. Proof is by induction on the length of \( p \).

Case 1) \( p = c \): Let \( u = v = \epsilon \). All the conditions are met.

Case 2) \( p = aq \): From the definition of \( A(t) \), \( q \in A(t) \).

Case 2.1) \( a \) is an own-substitution:

If \( q = \epsilon \), let \( v = a \), \( u = c \), where \( c \) is an other-substitution at \( a.time \). All the conditions are met.

If \( q \neq \epsilon \), inductively, \( q \in u'_\gamma v' \), where \( u' \in A(t) \), \( v' \in A(t) \), \( u'.time = v'.time = q.time \); further, since \( q \neq \epsilon \) neither \( u' \) nor \( v' \) is empty. Let \( u = u' \) and \( v = av' \). Since \( aq \in A(t) \), \( a.time = q.time \), hence, \( a.time = v'.time \), and \( av' = v \in A(t) \).

\[
\begin{align*}
aq & \quad \in \{q \in u'_\gamma v'\} \\
& \quad \in \{a \text{ is own-substitution, } a.time = u'.time\} \\
& \quad \leq \{u' \gamma av'\} \\
& \quad = \{u = u' \text{ and } v = av'\} \\
& \quad u_\gamma v
\end{align*}
\]
Corollary 8 Let $U$ and $V$ be broad sets, and $a_t \in U$ and $b_t \in V$, for some $a$ and $b$. Then, $A(t) \subseteq U \upharpoonright_a V$.

Proof:

\begin{align*}
a_t & \in U \\
\Rightarrow & \{U \text{ broad}\} \\
\Rightarrow & \{\beta(a_t) \subseteq U\} \\
\Rightarrow & \{A(t) \subseteq \beta(a_t), \text{from the definition of } \beta()\} \\
& A(t) \subseteq U \\
\Rightarrow & \{\text{similarly, } A(t) \subseteq V\} \\
& A(t) \subseteq U, A(t) \subseteq V \\
\Rightarrow & \{\text{apply } \upharpoonright_a\} \\
& A(t) \upharpoonright_a A(t) \subseteq U \upharpoonright_a V \\
\Rightarrow & \{A(t) \upharpoonright_a A(t) = A(t), \text{from Lemma 39, page 92}\} \\
& A(t) \subseteq U \upharpoonright_a V \\
\end{align*}

Lemma 40 ($U \upharpoonright_a V \backslash a = U \backslash a \upharpoonright_a V \backslash a$, where $a$ is an other-substitution.

Proof: Since $\backslash a$ is coercive, it is sufficient to prove that $(u \upharpoonright_a v) \backslash a = u \backslash a \upharpoonright_a v \backslash a$.

If both $u$ and $v$ do not start with $a$, then $u \upharpoonright_a v$ does not start with $a$, from the definition of $\upharpoonright_a$ and that $a$ is an other-substitution. Then $(u \upharpoonright_a v) \backslash a = \phi$. Also, at least one of $u \backslash a$ and $v \backslash a$ is $\phi$, so $u \backslash a \upharpoonright_a v \backslash a = \phi$.

If both $u$ and $v$ start with $a$, then $u = ap_t$ and $v =aq_t$, where $t = a.time$.

\begin{align*}
(u \upharpoonright_a v) \backslash a & = \{u = ap_t \text{ and } v =aq_t\} \\
& = \{\text{from the definition of } \upharpoonright_a, ap_t \upharpoonright_a aq_t = a(p_t \upharpoonright q_t) = a(p \upharpoonright q)\} \\
& = \{a(p \upharpoonright q) \backslash a\} \\
& = \{\text{from the definition of } \backslash a; \text{see Section 3.1.2, page 72}\}
\end{align*}
CHAPTER 3. BREADTH AND TRACE PRESERVATION

\[
p|_s q = \{u = ap_t \text{ and } v = aq_t\}. \text{ So, } \{p\} = u\backslash a \text{ and } \{q\} = v\backslash a
\]

\[u\backslash a, v\backslash a\]

**Lemma 41** Given that \(U\) and \(V\) broad, \(a\) is a base event at \(t\), \(a \in U\), and \(b_t \in V\) for some event \(b\). Then, \(u \in U\backslash a|_s V_{-t} \Rightarrow au_t \in U|_s V\).

**Proof:** Proof is by case analysis on \(u\).

- \(u = \epsilon\): We have to show that \(a \in U|_s V\). Given,
  
  \[a \in U \Rightarrow \{b_t \in V\} \Rightarrow \{\text{apply the definition of } |_s \text{ to } a|_s b_t, \nonumber\]
  
  \[\{a \preceq b_t \rightarrow a(\epsilon|_s b_t) \subseteq a|_s b_t\}; \nonumber\]
  
  \[\{a \preceq b_t \rightarrow a(\epsilon|_s b_t) \subseteq U|_s V\} \Rightarrow \{a \preceq b_t \text{ holds given that } a \text{ is base event at } t; \epsilon|_s b_t = \{\epsilon\}\} \nonumber\]
  
  \[a \in U|_s V\]

- \(u \neq \epsilon\): Given \(u \in U\backslash a|_s V_{-t}\), we have \(u \in p|_s q\), where \(p \in U\backslash a\), and \(q \in V_{-t}\). Neither \(p\) nor \(q\) is empty because \(u\) in non-empty.
  
  \[u \in p|_s q \Rightarrow \{\text{apply time-shift}\} \Rightarrow \{\text{concatenation}\} \Rightarrow \{\text{let } q = cr. \text{ Then, } ap_t|_s q_t = ap_t|_s c_t r_t \supseteq \{a \preceq c_t\} a(p_t|_s c_t r_t) = a(p_t|_s q_t)\} \Rightarrow \{p \in U\backslash a \Rightarrow ap_t \in U; q \in V_{-t} \Rightarrow q_t \in V\} \Rightarrow \{p \in U\backslash a \Rightarrow au_t \in U|_s V\} \Rightarrow \{p \in U\backslash a \Rightarrow au_t \in U|_s V\}\]

**Corollary 9** Given that \(U\) and \(V\) are broad, \(b\) a base event or an own-substitution at \(t\), \(b \in V\), \(a_t \in U\), for some event \(a\). Then, \(u \in U_{-t}|_s V\backslash b \Rightarrow bu_t \in U|_s V\).

**Proof:** Similar to Lemma 41, page 94.

**Theorem 21** Given that \(U\) and \(V\) are broad, \(U|_s V\) is broad.

**Proof:** If either of \(U\) or \(V\) is the empty set, then \(U|_s V\) is the empty set, which is broad. Now, assume that both sets are non-empty. We show for any \(u\) and \(v\), where \(u \in U\) and \(v \in V\), that \(\beta(u|_s v) \subseteq U|_s V\). Then from Corollary 4, page 69, \(U|_s V\) is broad.

The proof of \(\beta(u|_s v) \subseteq U|_s V\) is by induction on the combined length of \(u\) and \(v\).

- \(u = \epsilon\) or \(v = \epsilon\): Then \(u|_s v = \{\epsilon\}\), and we have to show \(\beta(\epsilon) \subseteq U|_s V\). From Observation 17, page 67, using \(\beta(U) = U\),
\[ \beta(e) \subseteq U \]

\[ \Rightarrow \{ \text{similarly with } V \} \]

\[ \beta(e) \subseteq U, \beta(e) \subseteq V \]

\[ \Rightarrow \{ \text{taking } \mid \} \]

\[ \beta(e) \mid \beta(e) \subseteq U \mid \beta V \]

\[ \Rightarrow \{ \beta(e) = A(0); \text{ from Lemma 39, page } 92, \beta(e) \mid \beta(e) = \beta(e) \} \]

\[ \beta(e) \subseteq \mathcal{U} \]

\[ \bullet \text{ } au_t \in U \text{ and } av_t \in V, \text{ where } a.time = t \text{ and } a \approx_v b: \text{ we show } \beta(au_t \mid av_t) \subseteq \mathcal{U} \mid a V. \]

\[ \beta(au_t \mid av_t) = \{ \text{definition of } \mid \} \]

\[ \beta(a(\mid a v_t) i) \]

\[ = \{ \text{definition of } \beta() \} \]

\[ A(t) \cup a \beta(a(\mid a v_t) i) \]

Now, \( A(t) \subseteq \mathcal{U} \mid a V \) follows from Corollary 8, page 93, because \( a \in U, a \in V, \) and \( a.time = t. \) We show \( a \beta(a(\mid a v_t) i) \subseteq \mathcal{U} \mid a V. \)

\[ au_t \in U, av_t \in V \]

\[ \Rightarrow \{ \text{definition} \} \]

\[ u \in U \setminus a, v \in V \setminus a \]

\[ \Rightarrow \{ \text{apply } \mid \} \]

\[ \mid a v_t \subseteq U \setminus a \mid V \setminus a \]

\[ \Rightarrow \{ U \setminus a \text{ and } V \setminus a \text{ are broad from Lemma 24, page } 72; \text{ apply induction} \} \]

\[ \beta(a(\mid a v_t) t) \subseteq \mathcal{U} \setminus a \setminus V \setminus a \]

\[ \Rightarrow \{ a \text{ is other-substitution; so, } U \setminus a \setminus V \setminus a = (\mathcal{U} \setminus V \setminus a), \text{ from Lemma 40, page } 93 \} \]

\[ \beta(a(\mid a v_t) t) \subseteq (\mathcal{U} \setminus V \setminus a) \]

\[ \Rightarrow \{ \text{definition of } (\mathcal{U} \setminus V \setminus a) \} \]

\[ a \beta(\mid a v_t) t \subseteq \mathcal{U} \setminus V \]

\[ \bullet \text{ } au \in U \text{ and } bv \in V, \text{ where } \neg(a \approx_v b): \text{ we show } \beta(au \mid bv) \subseteq \mathcal{U} \setminus V. \]

\[ \beta(au \mid bv) = \{ \text{definition of } \mid \text{ given } \neg(a \approx_v b) \} \]

\[ \beta([a \leq b \rightarrow a(\mid a bv)]) \cup [b \leq \epsilon a \rightarrow b(\mid a v)] \cup \{ \epsilon \} \]

\[ = \{ \beta() \text{ distributes over set union and guarded sets} \} \]

\[ \mid a \leq b \rightarrow \beta(a(\mid a bv))) \cup [b \leq \epsilon a \rightarrow \beta(\mid a v))] \cup \beta(\epsilon) \]

In the earlier proof with \( u = \epsilon \text{ or } v = \epsilon, \) we showed \( \beta(\epsilon) \subseteq \mathcal{U} \setminus V. \) We now show that \([a \leq b \rightarrow \beta(a(\mid a bv))]) \subseteq \mathcal{U} \setminus V \text{ and } [b \leq \epsilon \rightarrow \beta(\mid a v))] \subseteq \mathcal{U} \setminus V. \]

Case 1) \([a \leq b \rightarrow \beta(a(\mid a bv))]) \subseteq \mathcal{U} \setminus V:\]

If \( \neg(a \leq b), \) then \([a \leq b \rightarrow \beta(a(\mid a bv))]) = \{ \epsilon \}, \text{ which is trivially in } \mathcal{U} \setminus V. \)

Assume \( a \leq b. \) We rename the terms as \( au_t \text{ and } bv_t. \) Our goal is to show \( \beta(a(u_t b v_t)) \subseteq \mathcal{U} \setminus V, \text{ where } t = a.time, au_t \in U, b v_t \in V. \)
CHAPTER 3. BREADTH AND TRACE PRESERVATION

\[ \beta(a(u|_t b v_t)) \]
\[ = \{ \text{distribute time-shift} \} \]
\[ \beta(a(u|_t b v_t)_t) \]
\[ = \{ \text{definition of } \beta() \} \]
\[ A(t) \cup a\beta(u|_t b v_t)_t \]

From Corollary 8, page 93, \( A(t) \subseteq U|_s V \) (using \( au_t \in U, b v_t \in V \)). The remaining task is to show \( a \beta(u|_t b v_t)_t \subseteq U|_s V \).

\[ au_t \in U, \text{ and } b v_t \in V \]
\[ \Rightarrow \{ \text{definitions} \} \]
\[ u \in U \backslash a, \text{ and } b v \in V \backslash t \]
\[ \Rightarrow \{ U \backslash a \text{ is broad, from Lemma 24, page 72,} \]
\[ \text{given } b v_t \in V, \text{ and } V \text{ broad, from Lemma 25, page 73, } V \backslash t \text{ is broad,} \]
\[ \text{apply induction (combined length of } u \text{ and } b v \text{ is less than } au_t \text{ and } b v_t) \}
\[ \beta(u|_t b v)_t \subseteq U \backslash a V \backslash t \]
\[ \Rightarrow \{ \text{Apply Lemma 41, page 94, for each element in } \beta(u|_t b v): \}
\[ \text{ } a \text{ is base, from } a \leq b; \text{ also, } a.\text{time} = t, \]
\[ a \in U, \text{ and } au_t \in U, \text{ and } U \text{ is prefix-closed,} \]
\[ b v \in V, \text{ and } b v_t \in V, \text{ and } V \text{ is prefix-closed} \}
\[ a \beta(u|_t b v)_t \subseteq U|_s V \]

Case 2) \( b \leq_s a \Rightarrow \beta(b(a u|_t v)) \subseteq U|_s V \):

The proof is similar to that of Case (1); we include it here for completeness.

If \( \neg (b \leq_s a) \), then \( b \leq_s a \Rightarrow \beta(a(u|_t b v)) = \{ \epsilon \} \), which is in \( U|_s V \), from Observation 22, page 91. Assume \( b \leq_s a \). We rename the terms as \( a u_t \) and \( b v_t \). Our goal is to show \( \beta(b(a u_t|_t v_t)) \subseteq U|_s V \), where \( t = b.\text{time}, a u_t \in U, b v_t \in V \).

\[ \beta(b(a u_t|_t v_t)) \]
\[ = \{ \text{distribute time-shift} \} \]
\[ \beta(b(a u|_t v_t)_t) \]
\[ = \{ \text{definition of } \beta() \} \]
\[ A(t) \cup b \beta(a u|_t v_t)_t \]

From Corollary 8, page 93, \( A(t) \subseteq U|_s V \) (using \( a u_t \in U, b v_t \in V \)). The remaining task is to show \( b \beta(a u|_t v)_t \subseteq U|_s V \).

\[ a u_t \in U, \text{ and } b v_t \in V \]
\[ \Rightarrow \{ \text{definitions} \} \]
\[ a u \in U \backslash t, \text{ and } v \in V \backslash a \]
\[ \Rightarrow \{ \text{given } a u_t \in U, \text{ and } U \text{ broad, from Lemma 25, page 73, } U \backslash t \text{ is broad,} \]
\[ \text{given } V \text{ broad, from Lemma 24, page 72, } V \backslash b \text{ is broad,} \]
\[ \text{apply induction (combined length of } a u \text{ and } v \text{ is less than } a u_t \text{ and } b v_t) \}
\[ \beta(a u|_t v)_t \subseteq U \backslash t|_s V \backslash a \]
\[ \Rightarrow \{ \text{Apply Corollary 9, page 94, for each element in } \beta(a u|_t v): \}
\[ \text{ } b \text{ is base, or own-substitution, from } b \leq_s a, \text{ and } b.\text{time} = t \]
\[ a u \in U, \text{ and } a u_t \in U, \text{ and } U \text{ is prefix-closed,} \]
\[ b \in V, \text{ and } b v_t \in V, \text{ and } V \text{ is prefix-closed} \}
Lemma 42 Let \( u \) and \( v \) be visible. Then, \( \overline{uv} = \overline{u} \overline{v} \)

Proof: We prove the result by induction on the combined length of \( u \) and \( v \).

If either \( u \) or \( v \) is \( \epsilon \), both sides are \( \{ \epsilon \} \), from the definition. Next, we take \( au \) and \( bv \) which are both visible, and show that \( \overline{au} \overline{bv} = \overline{aubv} \). Note that \( u \) and \( v \) are visible, given \( au \) and \( bv \) are visible; either or both of \( u \) and \( v \) may be \( \epsilon \).

\[
\overline{au} \overline{bv} = \{ \text{definition of } \overline{au} \overline{bv} \} \n\]
\[
\{ a \approx b \rightarrow a(u \overline{v}) \} \cup \{ a \preceq b \rightarrow a(u \overline{v}) \} \cup \{ b \preceq_s a \rightarrow b(u \overline{v}) \} \n\]
\[
\{ \text{distribute trace over set union, guarded sets and concatenation} \} \n\]
\[
\{ a \approx b \rightarrow a(u \overline{v}) \} \cup \{ a \preceq b \rightarrow a(u \overline{v}) \} \cup \{ b \preceq_s a \rightarrow b(u \overline{v}) \} \n\]
\[
\{ \text{apply induction. Note that } u, v, au \text{ and } bv \text{ are visible} \} \n\]
\[
\{ a \approx b \rightarrow a(u \overline{v}) \} \cup \{ a \preceq b \rightarrow a(u \overline{v}) \} \cup \{ b \preceq_s a \rightarrow b(u \overline{v}) \} \n\]
\[
\{ \text{distribute trace over concatenation} \} \n\]
\[
\{ a \approx b \rightarrow a(u \overline{v}) \} \cup \{ a \preceq b \rightarrow a(u \overline{v}) \} \cup \{ b \preceq_s a \rightarrow b(u \overline{v}) \} \n\]
\[
\{ \text{in at least one of the guarded sets applies, and } \epsilon \in \{ \overline{u} \overline{v} \} \text{, from Observation 22, page 91} \} \n\]
\[
\overline{u} \overline{v} \n\]

Case 1) \( a, b = \tau, \tau \): \( \overline{au} \overline{bv} = \overline{au} \overline{bv} = \overline{uv} = \overline{uv} \)

\[
\overline{au} \overline{bv} = \{ \text{use } \neg(a \approx b) \text{ in (*) since } a, b = \tau, \tau; \text{ apply Observation 23, page 92} \} \n\]
\[
\{ a \preceq b \rightarrow (u \overline{v}) \} \cup \{ b \preceq_s a \rightarrow (u \overline{v}) \} \n\]
\[
\{ a, b = \tau, \tau, \text{ so } a \preceq b = a.time \preceq b.time, \text{ and } b \preceq_s a \equiv b.time \leq a.time \} \n\]
\[
\{ a.time \preceq b.time \rightarrow (u \overline{v}) \} \cup \{ b.time \preceq a.time \rightarrow (u \overline{v}) \} \n\]
\[
\{ \text{condition in at least one of the guarded sets applies, and } \epsilon \in \{ \overline{u} \overline{v} \} \text{, from Observation 22, page 91} \} \n\]
\[
\overline{u} \overline{v} \n\]

Case 2) \( a \neq \tau \) and \( b \neq \tau \):

\[
\overline{au} \overline{bv} = \{ \text{distribute trace over concatenation} \} \n\]
\[
\overline{au} \overline{bv} = \{ \text{apply definition} \} \n\]
\[
\{ a \approx b \rightarrow a(u \overline{v}) \} \cup \{ a \preceq b \rightarrow a(u \overline{v}) \} \cup \{ b \preceq_s a \rightarrow b(u \overline{v}) \} \n\]

And, this matches (*) given \( u = a \) and \( v = b \).

Case 3) \( a \neq \tau \) and \( b = \tau \):

Then, \( \overline{au} \overline{bv} = \overline{au} \overline{v} \) (L)

And,
CHAPTER 3. BREADTH AND TRACE PRESERVATION

Let 

\[ a \downarrow_{bv} \]

\[ \{ \text{since } b = \tau, \neg(a \approx_{x} b). \text{ Simplify (*)} \} \]

\[ \{ a \leq b \to a(\bar{v}, u)[v] \} \cup \{ b \leq_{x} a \to (a\bar{v}, u)[v] \} \cup \{ \epsilon \} \]

\[ \{ \text{drop } \{ \epsilon \}, \text{ using Observation 23, page 92, on the second term} \} \]

\[ \{ a \leq b \to a(\bar{v}, u)[v] \} \cup \{ b \leq_{x} a \to (a\bar{v}, u)[v] \} \]

(R)

Since \( bv \) is visible and \( b = \tau \), \( v \) is non-empty and visible. Therefore \( \bar{v} \neq \epsilon \).

Let c be the first event of \( \bar{v} \). Then, considering \( bv, b.time \leq c.time \). We show 

\[ (L) = (R) \] for 3 cases: (1) \( a.time < b.time \), (2) \( a.time = b.time \), and (3) \( a.time > b.time \).

Case 3.1) \( a.time < b.time \):

\[ (L) = a\bar{v}[u] \]

\[ \{ a.time < b.time \leq c.time \}. \text{ Hence, } \neg(a \approx_{x} c) \text{ and } \neg(c \leq_{x} a) \]

\[ \text{Apply definition of } \leq \] \( \]

\[ \{ a \leq c \to a(\bar{v}, u)[v] \} \cup \{ \epsilon \} \]

\[ \{ a.time \leq c.time \text{ means } a \leq c \equiv a \text{ is base } \}

\[ \{ a \text{ is base } \to a(\bar{v}, u)[v] \} \cup \{ \epsilon \} \]

And,

\[ (R) = \{ a \leq b \to a(\bar{v}, u)[v] \} \cup \{ b \leq_{x} a \to (a\bar{v}, u)[v] \} \]

\[ \{ a.time < b.time \}. \text{ So, } \neg(b \leq_{x} a). \text{ Also, } a \leq b \equiv a \text{ is base } \]

\[ \{ a \text{ is base } \to a(\bar{v}, u)[v] \} \cup \{ \epsilon \} \]

\[ \{ \text{from above derivation} \} \]

(L)

Case 3.2) \( a.time = b.time \):

\[ (R) = \{ a \leq b \to a(\bar{v}, u)[v] \} \cup \{ b \leq_{x} a \to (a\bar{v}, u)[v] \} \]

\[ \{ \text{from } a.time = b.time, a \leq b \equiv a \text{ is base } \text{ and } b \leq_{x} a \text{ holds, given } b = \tau \}

\[ \{ a \text{ is base } \to a(\bar{v}, u)[v] \} \cup \{ a\bar{v}[u] \} \]

\[ \{ \text{If } a \text{ is not base : } \}

\[ \{ a \text{ is base } \to a(\bar{v}, u)[v] \} \cup \{ a\bar{v}[u] \} = \{ \epsilon \} \cup \{ a\bar{v}[u] \} = (a\bar{v}, u)[v] \]

\[ \text{If } a \text{ is base : } \]

\[ \{ a \text{ is base } \to a(\bar{v}, u)[v] \} \cup \{ a\bar{v}[u] \} = a(\bar{v}, u)[v] \cup \{ a\bar{v}[u] \} \]

\[ a.time \leq c.time \text{ implies, from } |_{x} \text{ definition, } a\bar{v}[u] \supseteq a(\bar{v}, u)[v] \]

In all cases, \[ a \text{ is base } \to a(\bar{v}, u)[v] \} \cup \{ a\bar{v}[u] \} = (a\bar{v}, u)[v] \]

\[ a\bar{v}[u] = (L) \]

Case 3.3) \( a.time > b.time \):

\[ (R) = \{ a \leq b \to a(\bar{v}, u)[v] \} \cup \{ b \leq_{x} a \to (a\bar{v}, u)[v] \} \]

\[ \{ \neg(a \leq b) \text{ and } b \leq_{x} a \} \text{ hold} \]

\[ \{ \epsilon \} \cup \{ a\bar{v}[u] \} \]

\[ \{ \text{from Observation 22, page 91, } \epsilon } \subseteq (a\bar{v}, u)[v] \}

\[ a\bar{v}[u] = (L) \]
Case 4) $a = \tau$ and $b \neq \tau$: Similar to Case (3)

Lemma 43 $u|_x v \subseteq uc|_x vd$, for other substitutions $c$ and $d$.

Proof: We prove $u|_x v \subseteq uc|_x v$. Similarly, it can be shown that $u|_x v \subseteq u|_x vd$.

Now, we prove $u|_x v \subseteq uc|_x v$ by induction on the combined length of $u$ and $v$.

• $u = \varepsilon$ or $v = \varepsilon$: Left side is $\{\varepsilon\}$, and the right side, being a merge, contains $\varepsilon$, from Observation 22, page 91.

• $au|_x bv \subseteq auc|_x bv$:

   $$au|_x bv$$

   = \{definition of $|_x$\} \n
   $$[a \approx b \rightarrow a(u|_x v)] \cup [a \preceq b \rightarrow a(u|_x bv)] \cup [b \preceq a \rightarrow b(au|_x v)]$$

   \[ \subseteq \{\text{induction: } u|_x v \subseteq uc|_x v; \text{ similarly for the other terms} \} \]

   = \{definition of $|_x$\} \n
   $$auc|_x bv$$

   \[ \subseteq \{\text{from } uc \in U, uc \in U; \text{ similarly } vd \in V\} \]

   $U|_x V$

\[ \subseteq \{\text{from above} \} \]

$\overline{u|_x v}$

= \{definition of $|_x$\} \n
$\overline{p|_x \overline{q}}$

= \{definition of $\overline{\cdot}$\} \n
$\overline{p|_x q}$

\[ \subseteq \{p \in U \text{ and } q \in V\} \]

$\overline{U|_x V}$

Theorem 22 Let $U$ and $V$ be broad sets. Then $\overline{U|_x V} = \overline{U|_x V}$.

Proof: The proof is in two parts: $\overline{U|_x V} \subseteq \overline{U|_x V}$, and $\overline{U|_x V} \subseteq \overline{U|_x V}$.

• $\overline{U|_x V} \subseteq \overline{U|_x V}$: For any $u$ in $U$ and $v$ in $V$, we show $u|_x v \subseteq \overline{U|_x V}$.

   Let $p$ be the longest visible prefix of $u$ and $q$ of $v$. Then $\overline{p} = \overline{p}$ and $\overline{v} = \overline{v}$. From prefix-closure, $p \in U$ and $q \in V$.

   $$\overline{u|_x v}$$

   = \{from Lemma 21, page 70, there is $uc \in \beta(u) \subseteq U$ and, similarly, $vd \in V$, where we may pick $c$ and $d$ to be other-substitutions, apply Lemma 43, page 99\} \n
   $$\overline{(uc|_x vd)}$$

   = \{uc and vd are visible; apply Lemma 42, page 97\} \n
   $$\overline{(uc|_x vd)}$$

   \[ \subseteq \{\text{from } uc \in U, \overline{uc} \in U; \text{ similarly } vd \in V\} \]

   $\overline{U|_x V}$

   \[ \subseteq \{\text{from above} \} \]

   $\overline{p|_x \overline{q}}$

   = \{definition of $|_x$\} \n
   $\overline{p|_x q}$

   \[ \subseteq \{p \in U \text{ and } q \in V\} \]

   $\overline{U|_x V}$
3.4.2 Preliminary Results on Constrained Full Merge

Lemma 44 \( \overline{u +_v v} = \overline{u +_v v} \).

Proof: The proof is analogous to Lemma 42, page 97. Proof is by induction on
the combined length of \( u \) and \( v \).

Suppose \( u \) is \( \epsilon \): if \( v \) contains no other-substitution (then neither does \( \overline{v} \)),
\( \overline{u +_v v} = \{ \overline{v} \} \) and \( \overline{\overline{u +_v v}} = \{ \overline{v} \} \). If \( v \) contains an other-substitution, then so does \( \overline{v} \), and \( \overline{u +_v v} = \overline{\overline{v}} = \phi = \overline{\overline{u +_v v}} \). The proof for \( v = \epsilon \) is analogous.

Next, we show that \( \overline{au +_v bv} = \overline{au +_v bv} \).

\[
\begin{align*}
\overline{au +_v bv} &= \{ \text{definition of } \overline{au +_v bv} \} \\
&= \{ \text{distribution of trace over concatenation} \} \\
&= \{ \text{induction} \} \\
&= \{ \text{distribution trace over concatenation} \} \\
&= \{ \text{distribution trace over set union, guarded sets and concatenation} \}
\end{align*}
\]

\( \overline{au +_v bv} = \overline{au +_v bv} \) for each of these cases: (1) \( a, b = \tau, \tau \), (2) \( a \neq \tau \) and \( b = \tau \), (3) \( a = \tau \) and \( b \neq \tau \), (4) \( a = \tau \) and \( b \neq \tau \).

Case 1) \( a, b = \tau, \tau \): Then, \( \overline{au +_v bv} = \overline{au +_v bv} \)

\[
\begin{align*}
\overline{au +_v bv} &= \{ \text{simplify (*)}, noting that } -(a \approx b), \text{ from } a, b = \tau, \tau \} \\
&= \{ \text{given } a, b = \tau, \tau, a \leq b \equiv a.time \leq b.time \text{ and } b \leq a \equiv b.time \leq a.time, \text{ so, at least one of } a \leq b \text{ and } b \leq a \text{ holds} \} \\
&= \{ \text{induction} \} \\
&= \{ \text{distribution trace over concatenation} \} \\
&= \{ \text{distribution trace over set union, guarded sets and concatenation} \}
\end{align*}
\]

Case 2) \( a \neq \tau \) and \( b \neq \tau \):

\[
\begin{align*}
\overline{au +_v bv} &= \{ \text{distribution trace over concatenation} \} \\
&= \{ \text{apply definition} \} \\
&= \{ \text{distribution trace over set union, guarded sets and concatenation} \}
\end{align*}
\]

And, this matches (*) given \( \overline{a} = a \) and \( \overline{b} = b \).

Case 3) \( a \neq \tau \) and \( b = \tau \):

Then, \( \overline{au +_v bv} = \overline{au +_v bv} \) (L)

And,
CHAPTER 3. BREADTH AND TRACE PRESERVATION

\[ au +_x bv \]

\[ = \{ \text{from (*)}, \text{noting that } \neg(a \approx_x b), \text{from } b = \tau \} \]
\[ (a \leq b \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v})) \cup (b \preceq_x a \rightarrow (a\overrightarrow{a} +_x \overrightarrow{v})) \]  
\[ \text{(R)} \]

Since \( bv \) is visible and \( b = \tau \), \( v \) is non-empty and visible. Therefore \( \overrightarrow{v} \neq \epsilon \). Let \( c \) be the first event of \( \overrightarrow{v} \). Then, \( b.time \leq c.time \). We show \( (L) = (R) \) for 3 cases:

1. \( a.time < b.time \),
2. \( a.time = b.time \), and
3. \( a.time > b.time \).

Case 3.1) \( a.time < b.time \):

\[ (L) = a\overrightarrow{a} +_x \overrightarrow{v} \]
\[ = \{ a.time < b.time \leq c.time \}. \text{ Hence, } \neg(a \approx_x c) \text{ and } \neg(c \preceq_x a) \]
\[ \text{Apply definition of } +_x \]
\[ \langle a \leq c \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v}) \rangle \]
\[ = \{ \text{from } a.time < c.time, \ a \preceq c \equiv a \text{ is base} \} \]
\[ \langle a \text{ is base } \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v}) \rangle \]

And,

\[ (R) = \langle a \leq b \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v}) \rangle \cup (b \preceq_x a \rightarrow (a\overrightarrow{a} +_x \overrightarrow{v})) \]
\[ = \{ a.time < b.time. \ So, } \neg(b \preceq_x a) \} \]
\[ \langle a \leq b \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v}) \rangle \]
\[ = \{ a.time < b.time. \ So, } a \preceq b \equiv a \text{ is base } \} \]
\[ \langle a \text{ is base } \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v}) \rangle \]
\[ = \{ \text{above derivation} \} \]

\[ (L) \]

Case 3.2) \( a.time = b.time \):

\[ (R) = \langle a \leq b \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v}) \rangle \cup (b \preceq_x a \rightarrow (a\overrightarrow{a} +_x \overrightarrow{v})) \]
\[ = \{ \text{given } a.time = b.time \text{ and } b = \tau, \ b \preceq_x a \text{ holds} \} \]
\[ \langle a \leq b \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v}) \rangle \cup (a\overrightarrow{a} +_x \overrightarrow{v}) \]
\[ = \{ \text{given } a.time = b.time, a \preceq b \equiv a \text{ is base } \} \]
\[ \langle a \text{ is base } \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v}) \rangle \cup (a\overrightarrow{a} +_x \overrightarrow{v}) \]
\[ = \{ \text{If } a \text{ is not base :} \}
\[ \langle a \text{ is base } \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v}) \rangle \cup (a\overrightarrow{a} +_x \overrightarrow{v}) = a\overrightarrow{a} +_x \overrightarrow{v}, \]
\[ \text{If } a \text{ is base :} \]
\[ \langle a \text{ is base } \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v}) \rangle \cup (a\overrightarrow{a} +_x \overrightarrow{v}) = a\overrightarrow{a} +_x \overrightarrow{v}; \]
\[ a\overrightarrow{a} +_x \overrightarrow{v} = (L) \]

Case 3.3) \( a.time > b.time \):

\[ (R) = \langle a \leq b \rightarrow a(\overrightarrow{a} +_x \overrightarrow{v}) \rangle \cup (b \preceq_x a \rightarrow (a\overrightarrow{a} +_x \overrightarrow{v})) \]
\[ = \{ \neg(a \preceq b) \text{ and } b \preceq_x a \text{ hold} \}
\[ a\overrightarrow{a} +_x \overrightarrow{v} = (L) \]
Case (4) $a = \tau$ and $b \neq \tau$: Similar to Case(3).

Call a sequence pub-free if it has no publication. Formally, $\epsilon$ is pub-free, and $ap$ is pub-free iff $a$ is not a publication and $p$ is pub-free. A set is pub-free if all its sequences are.

**Observation 24** $A(t)$ is pub-free.

Proof: From the definition of $A(t)$.

**Lemma 45** Given $p$ is pub-free, $\beta(p)$ is pub-free.

Proof: By induction on the length of $p$. For $p = \epsilon$, $\beta(\epsilon) = A(0)$ is pub-free, by Observation 24. Next, consider $ap_t$ where $a$ is not a publication and $p$ is pub-free. Then, $\beta(ap_t) = A(t) \cup \beta(p)_t$, where $A(t)$ is pub-free, by Observation 24, and $\beta(p)$ is pub-free by induction hypothesis (then, so is $\beta(p)_t$).

**Lemma 46** Let $V$ be broad and $W$ be its pub-free subset. Then, $W$ is broad.

Proof: We show that for every $v$, where $v \in W$, $\beta(v) \subseteq W$.

\[
\begin{align*}
\forall v \in W & \Rightarrow \{W \subseteq V; v \in W \text{ means } v \text{ is pub-free}\} \\
& \Rightarrow \{V \text{ is broad}\} \\
& \Rightarrow \{v \text{ is pub-free implies } \beta(v) \text{ is pub-free, from Lemma 45, page 102}\} \\
& \Rightarrow \{W \text{ is the pub-free subset of } V\} \\
& \Rightarrow \beta(v) \subseteq W
\end{align*}
\]

**Lemma 47** $\beta(paqt) = \beta(pca) \cup pa\beta(q)_t$, where $t = a.time$, and $c$ is any substitution at time $t$.

Proof: Proof is by induction on the length of $p$.

- $p = \epsilon$: we have to show $\beta(aqt) = \beta(c) \cup a\beta(q)_t$.

\[
\begin{align*}
\beta(c) \cup a\beta(q)_t & = \{c \text{ is at time } t; \text{ from definition of } \beta(), \beta(c) = A(t) \cup c\beta(c)_t\} \\
& = \{cA(0)_t; \text{ $c$ being a substitution, } cA(0)_t \subseteq A(t)\} \\
& = \{\text{definition of } \beta()\} \\
& = \beta(aqt)
\end{align*}
\]

- $\beta(bpaqt) = \beta(bpc) \cup bpa\beta(q)_t$.

Let $s = b.time$, $p'_s = p$, $a'_s = a$, and $r = t - s.$
CHAPTER 3. BREADTH AND TRACE PRESERVATION

\[
\beta(bpaq_t) = \{ \text{transform using } p'_s = p, \ a'_s = a, \text{ and } r = t - s. \} \\
\beta(b(p'a'q)_s) = \{ \text{apply definition of } \beta(), \text{ using } s = b.time \} \\
A(s) \cup b \beta(p'a'q)_s = \{ \text{induction; note } a.time = t, \text{ so } (a'_s).time = t \} \\
\beta(bpd_s) \cup bpa \beta(q)_r + s = \{ \text{rewrite using } p'_s = p; \ r + s = t \} \\
\beta(bpd_s) \cup bpa \beta(q)_r + s
\]

Now, \( c = d_s \) is an arbitrary substitution at \( d.time + s = r + s = t \)

\[\Box\]

Lemma 48 Let \( pa \in U \), where \( U \) is broad. Then \( pc \in U \), where \( c \) is any substitution at \( a.time \).

Proof:

\[pa \in U \Rightarrow \{ U \text{ is broad} \} \Rightarrow \beta(pa) \subseteq U\]

\[\{ \text{from Lemma 47, page 102, with } q = \epsilon, \beta(pa) = \beta(pc) \cup pa \beta(\epsilon)_t \}
\]

\[\{ c.time = a.time \}
\]

\[\beta(pc) \subseteq U, \text{ where } c.time = a.time\]

\[\{ \text{from Corollary 2, page 69, } pc \in \beta(pc) \}
\]

\[\{ pc \in U, \text{ where } c.time = a.time \}

3.4.3 Asymmetric Composition Preserves Breadth

We show that for broad sets \( U \) and \( V \), \( U < x < V \) is broad.

Lemma 49 Let \( U \) and \( V \) be broad and \( V \) pub-free. For any \( p, p \in U|_x V \), there exist \( u \in U \) and \( v \in V \) such that \( d_1(u, v) \) and \( p \in u|_x v \).

Proof: First, we prove a sublemma.

Sublemma Consider sequences \( u \) and \( v \). Let \( u = u'cu'' \) where \( c \) is an own-substitution. Let \( d \) be an other-substitution, where \( c.time = d.time \) and \( d \) does not occur in \( v \). Then, \( u|_x v = u'd|_x v \).

Proof: Proof is by induction on the combined length of \( u' \) and \( v \).

- \( v = \epsilon \): Then \( u|_x v = \{ \epsilon \} = u'd|_x v \).
- \( u' = \epsilon \): We may assume that \( v \neq \epsilon \), i.e., \( v = bv' \).
CHAPTER 3. BREADTH AND TRACE PRESERVATION

\[ u|_x v \]

\[ = \{ u = cu'' \text{ and } v = bv' \} \]

\[ cu''|_x bv' \]

\[ = \{ \text{definition of } [\_] \} \]

\[ [c \approx \_ b \rightarrow c(u''|_x v')] \cup [c \not\leq b \rightarrow c(u''|_x v')] \cup [b \not\leq c \rightarrow b(cu''|_x v')] \]

\[ = \{ \neg(c \approx \_ b), \text{ since } c \text{ is not an other-substitution} \}
\]

\[ \neg(c \approx \_ b), \text{ since } c \text{ is not a base event} \]

\[ [b \not\leq c \rightarrow b(cu''|_x v')] \cup \{ \epsilon \} \]

\[ \{ \text{induction: } cu''|_x v' = d|_x v' \} \]

\[ [b \not\leq c \rightarrow b(d|_x v')] \cup \{ \epsilon \} \]

\[ = \{ b \not\leq c \rightarrow b \not\leq d, \text{ because } c.time = d.time \}
\]

\[ [b \not\leq d \rightarrow b(d|_x v')] \cup \{ \epsilon \} \]

And,

\[ u'd|_x v \]

\[ = \{ u' = \epsilon \text{ and } v = bv' \} \]

\[ d|_x bv' \]

\[ = \{ \text{definition of } [\_] \} \]

\[ [d \approx \_ b \rightarrow d(\epsilon|_x v')] \cup [d \not\leq b \rightarrow d(\epsilon|_x bv')] \cup [b \not\leq d \rightarrow b(d|_x v')] \]

\[ = \{ \neg(d \approx \_ b), \text{ since } d \text{ does not occur in } v \}
\]

\[ \neg(d \approx \_ b), \text{ since } d \text{ is not a base event} \]

\[ [b \not\leq d \rightarrow b(d|_x v')] \cup \{ \epsilon \} \]

\[ = \{ \text{from above proof} \}
\]

\[ u|_x v \]

\[ = \{ u = cu'cu'' = aw \text{ and } v = bv' \} \]

\[ aw|_x bv' \]

\[ = \{ \text{definition of } [\_] \} \]

\[ [a \approx \_ b \rightarrow a(w|_x v')] \cup [a \not\leq b \rightarrow a(w|_x bv')] \cup [b \not\leq a \rightarrow b(aw|_x v')] \]

\[ = \{ \text{induction on every term; recall } w = w'cu''; \text{ so, replace } w \text{ by } w'd \} \]

\[ [a \approx \_ b \rightarrow a(w'd|_x v')] \cup [a \not\leq b \rightarrow a(w'd|_x bv')] \cup [b \not\leq a \rightarrow b(aw'd|_x v')] \]

\[ = \{ \text{definition of } [\_] \} \]

\[ aw'd|_x bv' \]

\[ = \{ u' = aw' \text{ and } v = bv' \} \]

\[ u'd|_x v \]

\[ = \{ u' = aw' \text{ and } v = bv' \} \]

\[ w'd|_x v \]

Now, we are ready to prove the main lemma. Given \( p \in U|_x V \), there exist \( r \) and \( s \), in \( U \) and \( V \), respectively, such that \( p \in r|_x s \). If \( r \) has no own-substitution, then \( d_1(r, s) \), because \( s \) is pub-free (from \( V \) pub-free); thus \( u, v = r, s \).

If \( r \) has a own-substitution \( c \), let \( r = r'cr'' \), where \( r' \) has no own-substitution. Since \( U \) is broad, it is prefix-closed; therefore \( r = r'cr'' \in U \) implies \( r'c \in U \). Using Lemma 48, page 103, (set \( p = r' \) and \( a = c \), \( r'd \in U \), where \( d \) is any substitution at \( c.time \). Choose \( d \) to be an other-substitution that does not occur
in $s$ (this is possible because $s$ has a finite number of substitutions). Let $u = r'd$ and $v = s$. Then, $p \in r)|s = \{ \text{from Sublemma} \} r'd|_s s = u|_v v$. Further, $u$ has no own substitution because $r'$ has none and $d$ is an other-substitution. Also, $s$ is pub-free. So, $d_1(u, v)$.

**Lemma 50** Let $U$ and $W$ be broad and $W$ be pub-free. Then $U <x<v W = U|_x W$.

Proof:

$U|_x W$

= \{ \text{definition of coercion} \}

(\cup u \in U, w \in W : u|_x w)

= \{ \text{from Lemma 49, page 103} \}

(\cup u \in U, w \in W, d_1(u, w) : u|_x w)

= \{ \text{definition of } u <x<w \text{ using } w \text{ is pub-free} \}

(\cup u \in U, w \in W, d_1(u, w) : u <x<w)

= \{ \text{given } w \text{ is pub-free, } \neg d_1(u, w) \text{ implies } u <x<w = \phi \}

(\cup u \in U, w \in W, d_1(u, w) : u <x<w) \cup (\cup u \in U, w \in W, \neg d_1(u, w) : u <x<w)

= \{ \text{set theory} \}

(\cup u \in U, w \in W : u <x<w)

= \{ \text{definition of } <x< \text{ over sets} \}

$U <x<W$

**Theorem 23** For broad sets $U$ and $V$, $U <x<V$ is broad.

Proof: We show that for every $p$, where $p \in U <x<V$, $\beta(p) \subseteq U <x<V$. Let $W$ be the pub-free subset of $V$. We consider two cases: (1) $p \in U <x<W$, and (2) $p \in U <x<V - W$, and show in each case that $\beta(p) \subseteq U <x<V$.

• $p \in U <x<W$: We first show that $U <x<W$ is broad.

  $W$ is the pub-free subset of $V$

  $\Rightarrow$ \{ from Lemma 46, page 102 \}

  $W$ is broad

  $\Rightarrow$ \{ $U$ is broad; from Theorem 21, page 94 \}

  $U|_x W$ is broad

  $\Rightarrow$ \{ $U$ and $W$ are broad, and $W$ is pub-free; from Lemma 50, page 105 \}

  $U <x<W = U|_x W$, and $U|_x W$ is broad

  $\Rightarrow$ \{ obviously \}

  $U <x<W$ is broad

Hence,

$p \in U <x<W$

$\Rightarrow$ \{ $U <x<W$ is broad \}

$\beta(p) \subseteq U <x<W$

$\Rightarrow$ \{ $W \subseteq V$ \}

$\beta(p) \subseteq U <x<V$
CHAPTER 3. BREADTH AND TRACE PRESERVATION

- \( p \in U < x < (V - W) \): Any such \( p \) is generated by \( u \in U \) and \( v \in V \), where \( d_2(u, v) \). So, \( u = u'(t,[m/x])u''_v \) and \( v = v'(t,[m]/v') \), \( d_0(u', v') \) and \( d_1(u', v') \).

\[
\begin{align*}
p &\in (u' +_x v')(t,\tau)u''_u \\
\Rightarrow &\{ \text{apply } \beta() \text{ to both sides} \} \\
\beta(p) &\subseteq \beta((u' +_x v')(t,\tau)u''_u)
\end{align*}
\]

- \( \{ \text{from Lemma 47, page 102,} \} \)
- \( \beta((u' +_x v')(t,\tau)u''_u) = \beta((u' +_x v'c) \cup (u' +_x v')(t,\tau)\beta(u''_t), \) where \( c \) is an other-substitution at time \( t \} \)
- \( \beta(p) \subseteq \beta((u' +_x v'c) \cup (u' +_x v')(t,\tau)\beta(u''_t), \) where \( c \) is an other-substitution at \( t \)

We show that each of \( \beta((u' +_x v'c) \) and \( (u' +_x v')(t,\tau)\beta(u''_t) \) are subsets of \( U < x < V \).

Case 1) \( \beta((u' +_x v'c) \subseteq U < x < V \):

Since \( U \) is broad, and hence prefix-closed, from \( u'(t,[m/x])u''_v \in U \), we have \( u'(t,[m/x]) \in U \), and using Lemma 48, page 103, \( u'c \in U \); similarly, \( v'c \in V \). From \( d_1(u', v') \), \( u' \) has no own-substitution; therefore, \( u'c \) has none either. From \( d_1(u', v') \), \( v' \) is pub-free, and so is \( v'c \). So, \( d_1(u'c, v'c) \). Also, \( v'c \in V \), since \( v'c \in V \) and \( W \) is the pub-free subset of \( V \).

\[
\begin{align*}
\beta(u'c +_x v'c) \\
\subseteq &\{ \text{from Lemma 10, page 54, } u'c +_x v'c \subseteq u'c +_x v'c \} \\
\beta(u'c +_x v'c) \\
\subseteq &\{ u'c \in U; v'c \in W \} \\
\beta(U)_W \\
= &\{ U \text{ is broad,} \} \\
W \text{ is broad, from Lemma 46, page 102,} \\
so, U|_W \text{ is broad, from Theorem 21, page 94} \\
U|_W \\
= &\{ U \text{ and } W \text{ are broad, } W \text{ has no publication; from Lemma 50, page 105} \} \\
U < x < V \\
\subseteq &\{ W \subseteq V \} \\
U < x < V
\end{align*}
\]

Case 2) \( (u' +_x v')(t,\tau)\beta(u''_t) \subseteq U < x < V \):

\[
\begin{align*}
(u' +_x v')(t,\tau)\beta(u''_t) \\
= &\{ d_2(u, v) \text{ implies } d_2(w, v), \text{ where } w = u'(t,[m/x])w', \text{ for any } w', \} \\
use each element of \( \beta(u''_t) \text{ for } w'; \text{ apply definition of } < x < v < v \\
\beta(u'(t,[m/x])\beta(u''_t) < x < v \\
\subseteq &\{ \text{from Lemma 47, page 102, } u'(t,[m/x])\beta(u''_t) \subseteq \beta(u'(t,[m/x])u''_v) = \beta(u) \} \\
\beta(u) < x < v \\
\subseteq &\{ u \in U, v \in V \} \\
\beta(U) < x < V \\
= &\{ U \text{ is broad; so } \beta(U) = U \} \\
U < x < V
\end{align*}
\]
Lemma 51 \((U <x V\backslash a) \subseteq (U <x V)\backslash a\), where \(a\) is an own substitution and \(a.\text{time} = t\).

Proof: We show for arbitrary \(u \in U\) and \(v \in V\) that \((u <x v\backslash a) \subseteq (u <x v)\backslash a\). The result then follows by coercion.

Suppose \(v\) does not begin with \(a\). Then

\[
\begin{align*}
(u <x v\backslash a) & = \{\text{definition of } \backslash\} \\
& = \{u <x \emptyset\} \\
& = \{\text{coersion}\} \\
& = \emptyset.
\end{align*}
\]

Suppose \(d_1(u, v)\) holds. Then

\[
\begin{align*}
(u <x v)\backslash a & = \{\text{definition of } u <x v, d_1(u, v)\} \\
& = \{u|_x v\} \backslash a \\
& = \{\text{if } v \text{ does not begin with } a, \text{ then neither does any execution of } u|_x v\} \\
& = \emptyset.
\end{align*}
\]

Suppose \(d_2(u, v)\) holds. Then

\[
\begin{align*}
(u <x v)\backslash a & = \{\text{definition of } u <x v, d_2(u, v)\} \\
& = \{(u' +_x v')\backslash (t, \tau)u''\backslash a\} \\
& = \{\text{if } v \text{ does not begin with } a, \text{ then neither does any execution of } u' +_x v'\} \\
& = \emptyset.
\end{align*}
\]

Suppose neither \(d_1(u, v)\) nor \(d_2(u, v)\) holds. Then

\[
\begin{align*}
(u <x v)\backslash a & = \{\text{definition of } u <x v, \text{ neither } d_2(u, v) \text{ nor } d_2(u, v)\} \\
& = \emptyset \backslash a. \\
& = \{\text{definition of } \backslash\} \\
& = \emptyset.
\end{align*}
\]

Otherwise assume \(v\) starts with \(a\), and \(v = av'\).

Suppose \(d_1(u, v)\) holds. Then

\[
\begin{align*}
(u <x av'\backslash a) & = \{\text{definition of } \backslash\} \\
& = \{u <x v'\} \\
& = \{\text{definition of } u <x v', d_1(u, av') \text{ implies } d_1(u, v')\} \\
& = u|_x v'
\end{align*}
\]

and
\((u < x < av') \setminus a\)
\[= \{ \text{definition of } u < x < av', \; d_1(u, av') \} \]
\[= \{ \text{definition of } u < x < av' \} \]
\[\subseteq \{ \text{definition of } u \setminus av' \} \]
\[a(u \setminus v') \setminus a \]
\[= \{ \text{definition of } \setminus \} \]
\[u \setminus v' \]

Suppose \(d_2(u, v)\) holds. Then
\[u < x < av' \setminus a \]
\[= \{ \text{definition of } \setminus \} \]
\[u < x < v' \]
\[= \{ \text{definition of } u < x < v', \; d_2(u, av') \text{ implies } d_2(u, v') \} \]
\[= \{ \text{definition of } \setminus \} \]
\[(u'' + v'')(t, \tau)u'' \]

and
\[(u < x < av') \setminus a \]
\[= \{ \text{definition of } u < x < av', \; d_2(u, av') \} \]
\[= \{ \text{definition of } u < x < v', \; d_2(u, av') \text{ implies } d_2(u, v') \} \]
\[= \{ \text{definition of } \setminus \} \]
\[(u'' + v'')(t, \tau)u'' \]

Suppose neither \(d_1(u, v)\) nor \(d_2(u, v)\) holds. Then
\[u < x < av' \setminus a \]
\[= \{ \text{definition of } \setminus \} \]
\[u < x < v' \]
\[= \{ \text{definition of } u < x < v', \; -d_1(u, av') \text{ implies } -d_1(u, v') \text{ and } -d_2(u, av') \text{ implies } -d_2(u, v') \} \]
\[= \{ \text{definition of } \setminus \} \]
\[\emptyset \]

and
\[(u < x < av') \setminus a \]
\[= \{ \text{definition of } u < x < av', \; -d_1(u, av') \text{ and } -d_2(u, av') \} \]
\[= \{ \text{definition of } \setminus \} \]
\[\emptyset \]

Lemma 52 \((U \setminus a < x < V \setminus a) = (U < x < V) \setminus a\), where \(a\) is an other substitution and \(a\text{.time} = t\).

Proof: We show for arbitrary \(u \in U\) and \(v \in V\) that \((u \setminus a < x < v \setminus a) = (u < x < v) \setminus a\). The result then follows by coercion.

Suppose \(u\) does not begin with \(a\). (The case when \(v\) does not begin with \(a\) is similar.) Then
\[ (u|a < x < v|a) \]
\[ = \{ \text{definition of } \} \]
\[ (\emptyset < x < v|a) \]
\[ = \{ \text{coersion} \} \]
\[ \emptyset. \]

Suppose \( d_1(u, v) \) holds. Then

\[ (u < x < v)|a \]
\[ = \{ \text{definition of } u < x < v, d_1(u, v) \} \]
\[ (u|_x v)|a \]
\[ = \{ \text{if } u \text{ does not begin with } a, \text{ then neither does any execution of } u|_x v \} \]
\[ \emptyset. \]

Suppose \( d_2(u, v) \) holds. Then

\[ (u < x < v)|a \]
\[ = \{ \text{definition of } u < x < v, d_2(u, v) \} \]
\[ ((u'+_x v')(t, \tau)u'')|a \]
\[ = \{ \text{if } u \text{ does not begin with } a, \text{ then neither does any execution of } u'+_x v' \} \]
\[ \emptyset. \]

Suppose neither \( d_1(u, v) \) nor \( d_2(u, v) \) holds. Then

\[ (u < x < v)|a \]
\[ = \{ \text{definition of } u < x < v, \text{ neither } d_2(u, v) \text{ nor } d_2(u, v) \} \]
\[ \emptyset|a \]
\[ = \{ \text{definition of } \} \]
\[ \emptyset. \]

Otherwise, assume both \( u \) and \( v \) begin with \( a \), and \( u = au' \) and \( v = av' \). Suppose \( d_1(u, v) \) holds. Then

\[ au'|a < x < av'|a \]
\[ = \{ \text{definition of } \} \]
\[ u' < x < v' \]
\[ = \{ \text{definition of } u' < x < v', d_1(au', av') \text{ implies } d_1(u', v') \} \]
\[ u'|_x v' \]

and

\[ (au' < x < av')|a \]
\[ = \{ \text{definition of } au' < x < av', d_1(au', av') \} \]
\[ (au'|_x av')|a \]
\[ = \{ \text{definition of } au'|_x av' \} \]
\[ a(u'|_x v')|a \]
\[ = \{ \text{definition of } \} \]
\[ u'|_x v' \]
CHAPTER 3. BREADTH AND TRACE PRESERVATION

Suppose \( d_2(u, v) \) holds. Then

\[
au' \backslash a < x < av' \backslash a = \{ \text{definition of } \backslash \} \\
u' < x < v' = \{ \text{definition of } u' < x < v', \ d_2(au', av') \text{ implies } d_2(u', v') \} \\
(u'' + v'')(t, \tau)u''
\]

and

\[
(au' < x < av') \backslash a = \{ \text{definition of } au' < x < av', \ d_2(au', av') \} \\
(au'' + av'')(t, \tau)u'' \backslash a = \{ \text{definition of } au'' + av'' \} \\
a(au'' + av'')(t, \tau)u'' = \{ \text{definition of } \} \\
(u'' + v'')(t, \tau)u''
\]

Suppose neither \( d_1(u, v) \) nor \( d_2(u, v) \) holds. Then

\[
au' \backslash a < x < av' \backslash a = \{ \text{definition of } \backslash \} \\
u' < x < v' = \{ \text{definition of } u' < x < v', \ \neg d_1(au', av') \text{ implies } \neg d_1(u', v') \text{ and } \neg d_2(au', av') \text{ implies } \neg d_2(u', v') \} \\
\emptyset
\]

and

\[
(au' < x < av') \backslash a = \{ \text{definition of } au' < x < av', \ \neg d_1(au', av') \text{ and } \neg d_2(au', av') \} \\
\emptyset \backslash a = \{ \text{definition of } \} \\
\emptyset
\]

3.4.4 Asymmetric Composition Preserves Traces

We show that for broad sets \( U \) and \( V \), \( \bar{U} < x < \bar{V} = \bar{U} < x < \bar{V} \).

**Observation 25** \( d_0(u, v) \equiv d_0(\bar{u}, \bar{v}), \ d_1(u, v) \equiv d_1(\bar{u}, \bar{v}), \) and \( d_2(u, v) \equiv d_2(\bar{u}, \bar{v}) \).

**Proof:** The results follow from the definitions of \( d_0, \ d_1 \) and \( d_2 \).

**Theorem 24** Given that \( U \) and \( V \) are broad, \( \bar{U} < x < \bar{V} = \bar{U} < x < \bar{V} \).

**Proof:** Let \( V = W \cup R \), where \( W \) is the pub-free subset of \( V \) and every sequence in \( R \) has a publication. Then,
U <x< V
= \{\text{definition of } <x< \text{ over sets}\}
\((U <x< W) \cup (U <x< R)\)
= \{\text{distribute trace over set union}\}
U <x< W \cup U <x< R

Similarly, using V = W \cup R, we get
U <x< W \cup U <x< R = U <x< W \cup U <x< R. We show that
U <x< W = U <x< W, and U <x< R = U <x< R.

• U <x< W = U <x< W:
U <x< W
= \{\text{since } U \text{ and } W \text{ are broad, } U \text{ and } W \text{ are broad, from Lemma 23, page 71;}\}
W \text{ is pub-free since } W \text{ is; apply Lemma 50, page 105}\}
\(U \mid_w W\)
= \{\text{from Theorem 22, page 99}\}
\(U \mid_w W\)
= \{\text{simplify}\}
\(U \mid_w W\)
= \{\text{from Lemma 50, page 105}\}
U <x< W

• U <x< R = U <x< R: For any u \in U and w \in R, we first show that
u <x< w = \overline{u} <x< \overline{w} \quad (**)

Case 1) \neg d_2(u, w):
Since w \in R, w has a publication. So, \neg d_1(u, w). From \neg d_1(u, w) and \neg d_2(u, w),
u <x< w = \phi. From Observation 25, page 110, \neg d_1(\overline{u}, \overline{w}) and \neg d_2(\overline{u}, \overline{w}); so,
\overline{u} <x< \overline{w} = \phi.

Case 2) d_2(u, w): Let u = u'(t, [m/x])u'' and w = w'(t, !m)w''. From Observation 25, page 110, d_2(\overline{u}, \overline{w}),

\begin{align*}
\overline{u} <x< \overline{w} & \quad \{\text{from Lemma 44, page 100, } (u' + v')u'' = \overline{u}' + \overline{v}\}'\}
(\overline{u}' + \overline{v}')u'' & \quad \{\text{simplify}\}
(\overline{u}' + \overline{v}')u'' & \quad \{\text{from Lemma 44, page 100, } (u' + v') = \overline{u}' + \overline{v}\}'

\overline{u} <x< \overline{w} & \quad \{\text{in the above derivation, replace } u', v' \text{ and } u'' \text{ by } \overline{u}', \overline{v}' \text{ and } \overline{u}''\} }
\end{align*}
\[
(u' + w') u''
\]
\[
= \{ \text{simplify} \}
\]
\[
(u' + w') u''
\]
\[
= \{ \text{from above derivation} \}
\]
\[
\overline{u < x < w}
\]

Now, we show that \( \overline{U < x < R} = \overline{U < x < R} \), by mutual inclusion.

\begin{itemize}
  \item \( U < x < R \subseteq \overline{U < x < R} \):
    For any \( u \in U \) and \( w \in R \), we show \( \overline{\overline{u < x < w}} \subseteq \overline{U < x < R} \).
    \[
    \overline{u < x < w}
    \]
    \[
    = \{ \text{from (**)} \}
    \]
    \[
    \overline{u < x < w}
    \]
    \[
    \subseteq \{ \text{given } u \in U, \overline{u} \in \overline{U}; \text{ similarly, } \overline{w} \in \overline{R} \}
    \]
    \[
    \overline{U < x < R}
    \]
  
  \item \( \overline{U < x < R} \subseteq U < x < R \):
    For any \( p \in \overline{U} \) and \( q \in \overline{R} \), we show \( \overline{p < x < q} \subseteq \overline{U < x < R} \). Since \( p \in \overline{U} \), there is some \( u \in U \), such that \( \overline{u} = p \); similarly, there is some \( w \in R \), such that \( \overline{w} = q \).
    \[
    \overline{p < x < q}
    \]
    \[
    = \{ \overline{u} = p, \overline{w} = q \}
    \]
    \[
    \overline{u < x < w}
    \]
    \[
    \Rightarrow \{ \text{from (**)} \}
    \]
    \[
    \overline{u < x < w}
    \]
    \[
    \Rightarrow \{ u \in U, w \in R \}
    \]
    \[
    \overline{U < x < R}
    \]
\end{itemize}
Chapter 4

Traces are Denotations

We are finally ready to prove the main result, that the traces of an expression can be generated from the traces of its constituent expressions. Let * denote any Orc combinator, | , >x> or <x<.

**Theorem 25** For any expression \( f \), \( \llbracket f \rrbracket \) is broad.

Proof: The proof is by induction on the structure of the expression. For base expression \( f \), \( \llbracket f \rrbracket \) is broad, from Lemma 27, page 73. For \( f | g \), We have from Section 2.3, page 42

\[
\llbracket f | g \rrbracket = \llbracket f \rrbracket | \llbracket g \rrbracket
\]

\[
\Rightarrow \{\text{inductively, } \llbracket f \rrbracket \text{ and } \llbracket g \rrbracket \text{ are broad; apply Theorem 17, page 78}\}
\]

\[
\llbracket f | g \rrbracket = \llbracket f \rrbracket | \llbracket g \rrbracket \text{ and } \llbracket f \rrbracket | \llbracket g \rrbracket \text{ is broad}
\]

\[
\Rightarrow \{\text{obviously}\}
\]

\[
\llbracket f | g \rrbracket \text{ is broad}
\]

Proofs for the other combinators are similar: for \( >x> \), use the result from Section 2.4, page 44 and apply the breadth preservation Theorem 19, page 85; for \( <x< \), use the result from Section 2.5, page 50 and Theorem 23, page 105.

**Theorem 26** For any Orc combinator *, \( \{f \ast g\} = \{f\} \ast \{g\} \)

Proof:

\[
\{f \ast g\}
\]

\[
= \{\text{definition of trace}\}
\]

\[
\llbracket f \ast g \rrbracket
\]

\[
= \{\text{from the characterization theorems, } \llbracket f \ast g \rrbracket = \llbracket f \rrbracket \ast \llbracket g \rrbracket \}
\]

see Section 2.3, page 42 for Symmetric Composition,

see Section 2.4, page 44 for Sequential Composition,

see Section 2.5, page 50 for Asymmetric Composition.}
4.1 The Denotation of an Orc Expression

A family of functions $\mu_i$, for $i \geq 0$, is defined that maps recursive Orc expressions to trace sets. The denotation for recursive expression $f$ is defined as the least upper bound of the trace sets $\mu_i(f)$, where $\mu_i$ is defined by:

- $\mu_0(f) = \emptyset$
- $\mu_{i+1}(f) = \begin{cases} \{b\} & \text{if } f = b, \text{ a base expression} \\ \mu_{i+1}(g) * \mu_{i+1}(h) & \text{if } f = g * h \\ \mu_i([p/x].g) & \text{if } f = E(p) \text{ and } D(E(x)) = g \end{cases}$

And $\mu(f) = (\bigcup i : i \geq 0 : \mu_i(f))$.

**Lemma 53** $\mu(f) = \mu(f)$.

Proof: By induction on $i$. The base case is trivial, because $\emptyset$ contains no $\tau$ events. Assume $\mu_i(f) = \mu_i(f)$, for an expression $f$. Then $\mu_{i+1}(b) = \{b\} = \{b\}$.

For the combinator case,

$$\mu_{i+1}(f * g) = \{\text{definition of } \mu\}$$

$$\mu_{i+1}(f) * \mu_{i+1}(g) = \{\text{trace is idempotent}\}$$

$$\mu_{i+1}(f) * \mu_{i+1}(g) = \{\text{definition of } \mu\}$$

$$\mu_{i+1}(f * g) = \{\text{definition of } \mu\}$$

For a defined expression $E(p)$, where $E(x) \Delta g$,

$$\mu_{i+1}(E(p)) = \{\text{definition of } \mu\}$$

$$\mu_i([p/x].g) = \{\text{induction on } i\}$$
CHAPTER 4. TRACES ARE DENOTATIONS

\[ \mu_i([p/x].g) = \mu_{i+1}(E(p)) \]

**Lemma 54** For any combinator \(*, \emptyset \ast \emptyset = \emptyset.\)

Proof: By Lemma 29, page 77, \(\emptyset \mid \emptyset = \emptyset.\) By Corollary 7, page 84 \(\emptyset > x > \emptyset = \emptyset.\)

Finally

\[ \emptyset < x < \emptyset \]

\[ = \{\text{Lemma 50, page 105}\} \]

\[ \emptyset \mid \emptyset \]

\[ = \{\text{Lemma 39, page 92}\} \]

**Lemma 55** \(\mu(f)\) is broad.

Proof: We show by induction on \(i\) and the structure of \(f\) that \(\mu_i(f)\) is broad for all \(i\), where \(i \geq 0.\) For \(i = 0,\) the result follows from Lemma 16, page 67. Next, assume for all expressions \(f\) that \(\mu_i(f)\) is broad. We show \(\mu_{i+1}(f)\) is broad.

- \(f = b,\) a base expression: \(\mu_{i+1}(b) = \{b\},\) and \(\{b\}\) is broad by Lemma 27, page 73.
- \(f = g \ast h: \) \(\mu_{i+1}(g \ast h) = \mu_{i+1}(g) \ast \mu_{i+1}(h).\) By structural induction, both \(\mu_{i+1}(g)\) and \(\mu_{i+1}(h)\) are broad. The combinators preserve breadth by Theorem 17, page 78, Theorem 19, page 85 and Theorem 23, page 105. And trace preserves breadth by Lemma 23, page 71.
- \(f = E(p),\) where \(E(x) \Delta g: \) \(\mu_{i+1}(E(p)) = \mu_i([p/x].g),\) which is broad by induction on \(i.\)

**Lemma 56** \(\mu_i(a.(f \mid g)) \subseteq \mu_i(f \mid g) \backslash a,\) for all \(i,\) where \(i \geq 0.\)

Proof: The proof is by induction on \(i\) and the subterm ordering. \(\mu_0(a.(f \mid g)) = \emptyset\) by definition and \(\mu_i(f \mid g) \backslash a = \emptyset \backslash a = \emptyset\) by Observation 7, page 9. Next, assume \(\mu_i(a.(f \mid g)) \subseteq \mu_i(f \mid g) \backslash a.\) We show \(\mu_{i+1}(a.(f \mid g)) \subseteq \mu_{i+1}(f \mid g) \backslash a.\)

\[ \mu_{i+1}(a.(f \mid g)) \]

\[ = \{\text{definition of substitution}\} \]

\[ \mu_{i+1}(a.f \mid a.g) \]

\[ = \{\text{definition of } \mu_{i+1}\} \]

\[ \mu_{i+1}(a.f) \mid \mu_{i+1}(a.g) \]

\[ \subseteq \{\text{subterm induction}\} \]

\[ \mu_{i+1}(f) \backslash a \mid \mu_{i+1}(g) \backslash a \]

\[ \subseteq \{\text{Lemma 30, page 77}\} \]

\[ (\mu_{i+1}(f) \mid \mu_{i+1}(g)) \backslash a \]

\[ \subseteq \{\text{Lemma 3, page 20}\} \]
Lemma 57 \( \mu_i(a.(f \succ g)) \subseteq \mu_i(f \succ g) \setminus a \), for all \( i \), where \( i \geq 0 \).

Proof: The proof is by induction on \( i \) and the subterm ordering. \( \mu_0(a.(f \succ g)) = \emptyset \) by definition and \( \mu_i(f \succ g) \setminus a = \emptyset \) by Observation 7, page 9. Next, assume \( \mu_i(a.(f \succ g)) \subseteq \mu_i(f \succ g) \setminus a \). We show \( \mu_{i+1}(a.(f \succ g)) \subseteq \mu_{i+1}(f \succ g) \setminus a \).

- Case \( a = [m/x] \):
  \[
  \begin{align*}
  \mu_{i+1}(a.(f \succ g)) \\
  = \{ \text{definition of substitution} \} \\
  \mu_{i+1}(a.f \succ g) \\
  = \{ \text{definition of } \mu_{i+1} \} \\
  \mu_{i+1}(a.f) \succ \mu_{i+1}(g) \\
  \subseteq \{ \text{subterm induction} \} \\
  \mu_{i+1}(f) \setminus a \succ \mu_{i+1}(g) \\
  = \{ \text{Lemma 36, page 86} \} \\
  \mu_{i+1}(f) > \mu_{i+1}(g) \setminus a \\
  \subseteq \{ \text{Lemma 3, page 20} \} \\
  \mu_{i+1}(f) > \mu_{i+1}(g) \setminus a \\
  = \{ \text{definition of } \mu_{i+1} \} \\
  \mu_{i+1}(f \succ g) \setminus a
  \end{align*}
  \]

- Case \( a = [m/y], \) where \( y \neq x \):
  \[
  \begin{align*}
  \mu_{i+1}(a.(f \succ g)) \\
  = \{ \text{definition of substitution, } x \neq y \} \\
  \mu_{i+1}(a.f \succ a.g) \\
  = \{ \text{definition of } \mu_{i+1} \} \\
  \mu_{i+1}(a.f) \succ \mu_{i+1}(a.g) \\
  \subseteq \{ \text{subterm induction} \} \\
  \mu_{i+1}(f) \setminus a \succ \mu_{i+1}(g) \setminus a \\
  = \{ \text{Lemma 37, page 87} \} \\
  \mu_{i+1}(f) > \mu_{i+1}(g) \setminus a \\
  \subseteq \{ \text{Lemma 3, page 20} \} \\
  \mu_{i+1}(f) > \mu_{i+1}(g) \setminus a \\
  = \{ \text{definition of } \mu_{i+1} \} \\
  \mu_{i+1}(f \succ g) \setminus a
  \end{align*}
  \]

Lemma 58 \( \mu_i(a.(f \prec g)) \subseteq \mu_i(f \prec g) \setminus a \), for all \( i \), where \( i \geq 0 \).

Proof: The proof is by induction on \( i \) and the subterm ordering. \( \mu_0(a.(f \prec g)) = \emptyset \) by definition and \( \mu_i(f \prec g) \setminus a = \emptyset \) by Observation 7, page 9. Next,
assume \( \mu_i(a.(f <x< g)) \subseteq \mu_i(f <x< g) \setminus a \). We show \( \mu_{i+1}(a.(f <x< g)) \subseteq \mu_{i+1}(f <x< g) \setminus a \).

- Case \( a = [m/x] \):
  \[
  \begin{align*}
  \mu_{i+1}(a.(f <x< g)) &= \{ \text{definition of substitution} \}
  \\
  &= \mu_{i+1}(f <x< a.g) \\
  &= \{ \text{definition of } \mu_{i+1} \}
  \\
  &= \mu_{i+1}(f) <x< \mu_{i+1}(a.g) \\
  \subseteq& \{ \text{subterm induction} \}
  \\
  &= \mu_{i+1}(f)<x<\mu_{i+1}(g) \setminus a \\
  \subseteq& \{ \text{Lemma 51, page 107} \}
  \\
  &= \{ \text{Lemma 3, page 20} \}
  \\
  &= \{ \text{definition of } \mu_{i+1} \}
  \\
  &= \mu_{i+1}(f)<x<\mu_{i+1}(g) \setminus a \\
  \subseteq& \{ \text{subterm induction} \}
  \\
  &= \mu_{i+1}(f)<x<\mu_{i+1}(g) \setminus a \\
  \subseteq& \{ \text{Lemma 52, page 108} \}
  \\
  &= \{ \text{Lemma 3, page 20} \}
  \\
  &= \{ \text{definition of } \mu_{i+1} \}
  \\
  &= \mu_{i+1}(f)<x<\mu_{i+1}(g) \setminus a.
  \end{align*}
\]

- Case \( a = [m/y] \), where \( y \neq x \):
  \[
  \begin{align*}
  \mu_{i+1}(a.f) &= \{ \text{definition of substitution, } x \neq y \}
  \\
  &= \mu_{i+1}(a.f)<x<\mu_{i+1}(a.g) \\
  \subseteq& \{ \text{subterm induction} \}
  \\
  &= \mu_{i+1}(f)\setminus a <x<\mu_{i+1}(g) \setminus a \\
  \subseteq& \{ \text{Lemma 52, page 108} \}
  \\
  &= \{ \text{Lemma 3, page 20} \}
  \\
  &= \{ \text{definition of } \mu_{i+1} \}
  \\
  &= \mu_{i+1}(f)<x<\mu_{i+1}(g) \setminus a \\
  \subseteq& \{ \text{subterm induction} \}
  \\
  &= \mu_{i+1}(f)<x<\mu_{i+1}(g) \setminus a.
  \end{align*}
\]

**Lemma 59** For expression \( f \), substitution \( a \) and \( i \geq 0 \), \( \mu_i(a.f) \subseteq \mu_i(f) \setminus a \).

Proof: By induction on \( i \).

- \( \mu_0(a.f) = \mu_0(f) \setminus a \):
  \[
  \begin{align*}
  \mu_0(a.f) &= \{ \text{definition of } \mu_0 \}
  \\
  &= \emptyset \\
  &= \{ \text{Observation 7, page 9} \}
  \\
  &= \emptyset \setminus a \\
  &= \{ \text{definition of } \mu_{i+1} \}
  \\
  &= \mu_0(f) \setminus a
  \end{align*}
\]
• $\mu_{i+1}(a.f) = \mu_{i+1}(f)\backslash a$: By induction on the structure of $f$.

  – Case $f = b$, a base expression:

  $\mu_{i+1}(a.b)$

  $= \{\text{definition of } \mu_{i}, a.b \text{ is a base expression}\}$

  $\langle a.b \rangle$

  $= \{\text{Corollary 1, page 21}\}$

  $\langle b \rangle \backslash a$

  $= \{\text{definition of } \mu_{i+1}(b)\}$

  $\mu_{i+1}(b) \backslash a$

  – Case $f = g \ast h$:

  $\mu_{i+1}(a.(g \ast h))$

  $= \{\text{definition of substitution}\}$

  $\mu_{i+1}(a.g \ast a.h)$

  $= \{\text{definition of } \mu_{i+1}\}$

  $\mu_{i+1}(a.g) \ast \mu_{i+1}(a.h)$

  $= \{\text{induction on } a.g \text{ and } a.h \text{ as subterms of } a.f\}$

  $\mu_{i+1}(g) \backslash a \ast \mu_{i+1}(h) \backslash a$

  $\subseteq \{\text{Lemma 56, page 115, Lemma 57, page 116 and Lemma 58, page 116}\}$

  $(\mu_{i+1}(g) \ast \mu_{i+1}(h)) \backslash a$

  $= \{\text{definition of } \mu_{i+1}\}$

  $(\mu_{i+1}(g \ast h)) \backslash a$

  – Case $f = E(p)$, where $D(E(x)) = g$: We consider two cases. First, assume the substitution is $[m/p]$:

  $\mu_{i+1}([m/p].E(p))$

  $= \{\text{definition of substitution}\}$

  $\mu_{i+1}(E(m))$

  $= \{\text{definition of } \mu_{i+1}\}$

  $\mu_{i}([m/x].g)$

  $= \{\text{only } x \text{ is free in } g\}$

  $\mu_{i}([m/p].([p/x].g))$

  $= \{\text{induction on } i\}$

  $\mu_{i}([p/x].g) \backslash [m/p]$

  $= \{\text{definition of } \mu_{i+1}\}$

  $\mu_{i+1}(E(p)) \backslash [m/p]$

  Next, assume the substitution is $[m/q]$, where $q \neq p$.

  $\mu_{i+1}([m/q].E(p))$

  $= \{\text{definition of substitution}\}$

  $\mu_{i+1}(E(p))$

  $= \{\text{definition of } \mu_{i+1}\}$

  $\mu_{i}([p/x].g)$

  $= \{\text{only } x \text{ is free in } g\}$

  $\mu_{i}([m/q].[p/x].g)$

  $= \{\text{induction on } i\}$
\[ \mu_i([p/x].g)\{m/q] = \{\text{definition of } \mu_{i+1}\} \mu_{i+1}(E(p))\{m/q] \]

**Lemma 60** For expression \( f \) and substitution event \( a \), \( \mu(a.f) \subseteq \mu(f)\{a]. \)

Proof:

\[
\mu([m/x].f) = \{\text{definition of } \mu\} \\
\quad (\cup i : i \geq 0 : \mu_i([m/x].f)) \\
= \{\text{Lemma 59, page 117}\} \\
\quad (\cup i : i \geq 0 : \mu_i(f)\{m/x]) \\
= \{\text{set theory, definition of } \mu_0(f)\{m/x]\} \\
\quad (\cup i : i \geq 0 : \mu_i([p/x].g)) \\
= \{\text{definition of } \mu\} \\
\quad (\cup i : i \geq 0 : \mu_i([p/x].g) \{m/x]) \\
= \{\mu([p/x].g)] \\
\quad \subseteq \mu([p/x].g) \\
\quad \mu([p/x].g) \\
\]

**Lemma 61** Suppose \( D(E(x)) = g \). Then \( \mu(E(p)) = \mu([p/x].g) \).

Proof:

\[
\mu(E(p)) = \{\text{definition of } \mu\} \\
\quad (\cup i : i \geq 0 : \mu_i(E(p))) \\
= \{\text{set theory, definition of } \mu_0(E(p))\} \\
\quad (\emptyset \cup (\cup i : i \geq 0 : \mu_{i+1}(E(p))) \\
= \{\text{definition of } \mu_{i+1}(E(p))\} \\
\quad (\emptyset \cup (\cup i : i \geq 0 : \mu_i([p/x].g)) \\
= \{\text{definition of } \mu\} \\
\quad (\emptyset \cup \mu([p/x].g) \\
= \{\emptyset \subseteq \mu([p/x].g)\} \\
\quad \mu([p/x].g) \\
\]

**Theorem 27** (Equivalence of Semantics) For expression \( f \), \( \{f\} = \mu(f) \).

Proof: By well founded induction on the product of the subterm ordering on the structure of \( f \) and the usual ordering on the natural numbers.

- \( f = b \), a base expression

\[
\mu(b) = \{\text{definition of } \mu\} \\
\quad (\emptyset \cup (\cup i : i \geq 0 : \mu_{i+1}(b)) \\
= \{\text{definition of } \mu_{i+1}(b)\} \\
\quad (\emptyset \cup (\cup i : i \geq 0 : \{b\}) \\
= \{\emptyset \subseteq \{f\}, \text{ for any expression } f\} \\
\quad \{b\}. \\
\]
• $f = g * h$:

$\{g * h\}$

$= \{\text{Theorem 26, page 113}\}$

$\{g\} * \{h\}$

$= \{\text{induction}\}$

$\mu(g) * \mu(h)$

$= \{\text{definition of } \mu\}$

$($\bigcup : i \geq 0 : \mu_i(g) * \mu_i(h)\)$

$= \{\text{Theorem 10, page 40, monotonicity of } \mu_i\}$

$($\bigcup : i \geq 0 : \mu_i(g) * \mu_i(h)\)$

$= \{\text{definition of } \mu_i\}$

$0_0 \cup ($\bigcup : i \geq 0 : \mu_i(g) * \mu_i(h)\)$

$= \{\text{definition of } \mu, \mu(g * h) = \mu(g * h)\}$

$\mu(g * h)$

\[ E(p), \text{ where } E(x) \triangleright g. \text{ The proof is by mutual inclusion.} \]

$\mu(E(p)) \subseteq \{E(p)\}$: We show, for all $i \geq 0$, that $\mu_i(E(p)) \subseteq \{E(p)\}$. We proceed by induction on $i$.

Suppose $i = 0$ and $u \in \emptyset$. Then $u \in \{E(p)\}$ by definition. Otherwise assume that $\mu_i(E(p)) \subseteq \{E(p)\}$ and consider $u \in \mu_{i+1}(E(p))$.

$u \in \mu_{i+1}(E(p))$

$\Rightarrow \{\text{definition}\}$

$u \in \mu_i([p/x].g)$

$\Rightarrow \{\text{induction on } i\}$

$u \in \{[p/x].g\}$

$\Rightarrow \{\text{by definition, for some } v \text{ such that } \overline{v} = u\}$

$v \in [[p/x].g]$  

$\Rightarrow \{\text{operational semantics, } E(x) \triangleright g\}$

$(0, \overline{\tau})v \in [[E(p)]]$

$\Rightarrow \{\overline{0, \tau}v = \overline{v} = u, [[E(p)]] = \{E(p)\}\}$

$u \in \{E(p)\}$

$\{E(p)\} \subseteq \mu(E(p))$: Consider $u \in \{(E(p))\}$. Let $v \in [[E(p)]]$ such that $\overline{v} = u$. We show $\overline{v} \in \mu(E(p))$, which implies $u \in \mu(E(p))$. The proof proceeds by induction on $v$.

Suppose $v = \epsilon$, so $\overline{v} = \epsilon$. Then the result follows from Lemma 55, page 115, breadth of $\mu$, because $\epsilon$ is in all broad sets. Otherwise $v = av'$. If $a$ is a substitution event, then $t = 0$ because $E(p)^t = \bot$ for $t > 0$.

$av' \in [[E(p)]]$

$\Rightarrow \{\text{operational semantics}\}$

$v' \in [[a.E(p)]]$

$\Rightarrow \{\text{definition of trace}\}$
\[ \overrightarrow{v'} \in \{a.E(p)\} \]
\[ \Rightarrow \{\text{induction on } v'\} \]
\[ \overrightarrow{v'} \in \mu(a.E(p)) \]
\[ \Rightarrow \{\mu(a.E(p)) \subseteq \mu(E(p))\setminus a \text{ by Lemma 60, page 119}\} \]
\[ \overrightarrow{v'} \in \mu(E(p))\setminus a \]
\[ \Rightarrow \{\text{definition of } \setminus\} \]
\[ av' \in \mu(E(p)) \]
\[ \Rightarrow \{av' = \overrightarrow{av}\} \]
\[ av' \in \mu(E(p)) \]

Otherwise, by rule (Def), \( E(p) \xrightarrow{0,\tau} [p/x].g \xrightarrow{v'} \), where \( E(x) \triangleq g \).
So \( v = (0, \tau)v' \).
\[ (0, \tau)v' \in [E(p)] \]
\[ \Rightarrow \{\text{operational semantics}\} \]
\[ v' \in [[p/x].g] \]
\[ \Rightarrow \{\text{definition of trace}\} \]
\[ \overrightarrow{v'} \in ([p/x].g) \]
\[ \Rightarrow \{\text{induction on } v'\} \]
\[ \overrightarrow{v'} \in \mu([p/x].g) \]
\[ \Rightarrow \{\text{Lemma 61, page 119}\} \]
\[ \overrightarrow{v'} \in \mu(E(p)) \]
\[ \Rightarrow \{\overrightarrow{v'} = (0, \tau)v'\} \]
\[ (0, \tau)v' \in \mu(E(p)) \]