Observing "True" Concurrency

by

Lalita Jategaonkar

S.B., Electrical Engineering and Computer Science
Massachusetts Institute of Technology
(1989)
S.M., Electrical Engineering and Computer Science
Massachusetts Institute of Technology
(1989)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1993

Signature of Author..
Department of Electrical Engineering and Computer Science
July 30, 1993

Certified by..
Albert R. Meyer
Hitachi America Professor of Engineering
Thesis Supervisor

Accepted by..
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Observing "True" Concurrency

by
Lalita Jategaonkar

Submitted to the Department of Electrical Engineering and Computer Science
on July 30, 1993, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In concurrent process theory, processes are often modeled by state machines and Petri Nets. Algebraic process theories based on state machines, exemplified by Milner's CCS and Hoare's CSP, have been more fully developed than Net-based theories, but are inadequate for modeling "true" concurrency concepts such as non-atomic actions, action refinement, locality of actions, and multithreadedness. We introduce an action refinement operator and present some "fully abstract" semantics for "true" concurrency. We show that these semantics are decidable for finite-state concurrent processes and characterize their computational complexity.

Thesis Supervisor: Albert R. Meyer

Title: Hitachi America Professor of Engineering

Acknowledgments

First and foremost, my deepest thanks to Albert R. Meyer, without whom this thesis would not have been possible. His constant guidance, encouragement, and advice about both technical and non-technical matters has been invaluable to me during my time at MIT. It is impossible for me to distill in a few sentences the innumerable things I have learned from him over the last several years, so I will say only that I feel very lucky to have had such a wonderful thesis advisor.

My thanks to John Mitchell for introducing me to the area of programming language theory, and for his continuing encouragement over the years. I would also like to thank my thesis readers, Nancy Lynch and David McAllester, for their helpful comments and their ideas about further directions for this research.

My thanks to Bell Labs for the generous fellowship that made this thesis possible. I would also like to thank David MacQueen for his continued help and attention over the years.

Special thanks to Larry Stockmeyer for making available to us his unpublished notes on the lower bound for deciding history-preserving bisimulation of finite-state processes, and to Alex Rabinovich for his very helpful presentation of the material to us.

In addition, I am grateful to a large number of people for many helpful technical discussions about the material in this thesis, including Bard Bloom, Steve Brookes, Uffe Engberg, Rob van Glabbeek, Ursula Goltz, Roberto Gorrieri, Matthew Hennessy, Radhakrishnan Jagadeesan, Astrid Kiehn, Alain Mayer, Mogens Nielsen, Vaughan Pratt, Wolfgang Reisig, Roberto Segala, Boris Trakhtenbrot, Frits Vaandrager, Walter Vogler, David Wald.

I would like to thank David Jones for his very patient help with Latex.

My heartfelt thanks to my friends for making my stay at MIT a very enjoyable one. I would like to thank my parents-in-law, K. and Sarada Jagadeesan, for their constant encouragement. I would like to thank my husband, Radhakrishnan Jagadeesan, for bringing me so very much happiness, and for his love, encouragement, and optimism.

My warmest thanks to my parents, Vasanti and Arun Jategaonkar, for their never-ending patience, support, encouragement, and strength, without which I would never even have begun, much less completed, this thesis.

Comments on Joint Research

Much of this thesis is based on joint work with Albert R. Meyer. The results in Chapter 3, Chapter 5, and in the first section of Chapter 2 appeared previously in a joint paper with Meyer [23]. The definitions and ideas presented in those chapters are joint work; however, the proofs of the main theorems in those chapters are largely my own work.

The results in Chapter 6 appeared previously in another joint paper with Meyer [24]. The definitions, ideas, and proofs given in Section 6.2 are joint work, as are the proofs of Theorem 6.3.4, Theorem 6.4.1, and Theorem 6.4.3. The proof of Theorem 6.3.6, the other results in Section 6.4, and the proofs of Theorems 6.5.1 and 6.5.3 are my own work.

The second section of Chapter 2 is largely a synthesis and adaptation of results in the literature.

Chapter 4 is entirely my own work.

Dedication

To my parents, Arun and Vasanti Jategaonkar.

Contents

Chapter 1

Introduction

In concurrent process theory, processes are often represented by state machines and Petri Nets. State machines, by definition, have no explicit representation of concurrency, and they identify concurrent actions with sequential, interleaved actions. Process theories based on state machines, exemplified by Milner's CCS [30] and Hoare's CSP [21], typically have associated combinators for composing large processes from smaller components, compositional techniques for reasoning about processes through reasoning about their components, sound and complete techniques for reasoning about process equivalence, and algorithms for deciding equivalence of finite-state processes. These elegant properties have led to automatic verification techniques and tools such as model checkers.

However, as is well-known, the state machine approach is inherently inadequate for describing action refinement, the operation of refining atomic actions in a concurrent process, which suggests aspects of top-down "modular" development [1, 2, 3, 10, 16, 20, 32, 39, 40, 41, 47] and "changes of granularity" [28, 31]. This limitation is a direct result of the identification of concurrent actions with sequential, interleaved actions. For example, the state-machine representation of the concurrent process $a \mid\mid b$, which can *concurrently* perform an a and b action, is identical to the state-machine representation of the purely sequential process $ab + ba$, which can *sequentially* perform an a and b action in either order. This state-machine is pictured in Figure 1-1.

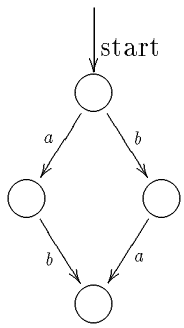
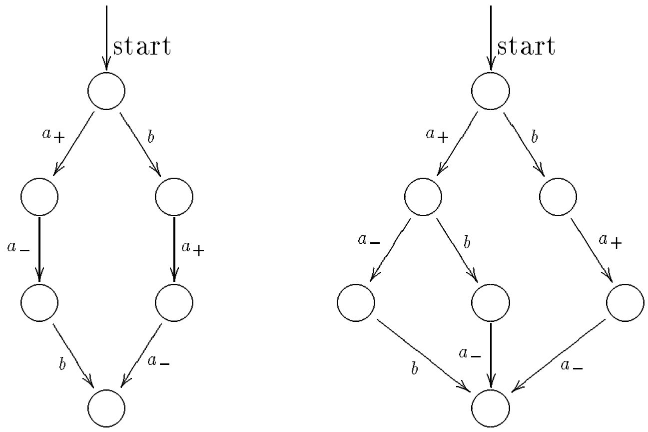


Figure 1-1: State-Machine Representation of $a \mid\mid b$ and $ab + ba$



$split_{(a,a_+,a_-)}(ab + ba)$ $\quad$ $split_{(a,a_+,a_-)}(a \mid\mid b)$

Figure 1-2: After Action Refinement

Now, if the a action in both processes is "refined" or "split" into two actions, a_+ followed by a_-, the resulting processes, $split_{(a,a_+,a_-)}(a \mid\mid b) \stackrel{\text{def}}{=} a_+a_- \mid\mid b$ and $split_{(a,a_+,a_-)}(ab + ba) \stackrel{\text{def}}{=} a_+a_-b + ba_+a_-$ have completely different state-machine representations, pictured in Figure 1-2. Thus, the state machine model is inherently inadequate for describing even the simplest forms of action refinement on concurrent processes. In particular, the state machines of Figure 1-2 are distinguished by all the state-machine based process equivalences in the literature, including bisimulation equivalence (CCS) [30], partial-trace equivalence, failures equivalence (CSP) [8, 9], and Hennessy's Testing-equivalence [19], an elegant experimental justification for partial-traces and failures.

We remark that the operation of refining transitions in a state-machine does model action refinement of *purely sequential* processes. Moreover, trace equivalence, failures equivalence, Testing-equivalence, and intuitively simple variations of bisimulation, notably delay bisimulation and branching bisimulation [12, 13, 43] are sound techniques for reasoning about action refinement on purely sequential processes. However, since all of these equivalences identify concurrent actions with interleaved actions, none of them are sound for reasoning about action refinement on concurrent processes.

Petri Net theory, on the other hand does distinguish "true" concurrency from interleaving by axiomatizing a "causal" partial order on process actions, and is adequate for describing action refinement of concurrent processes. However, Petri Net theory typically does not offer an explanation of how an external observer can detect causality. Hence, in contrast to the state-machine theories, Petri Net theory does not provide complete techniques for reasoning about process equivalence, and compositional reasoning techniques and associated decision procedures are also much less developed.

An important problem is to merge these viewpoints by developing an operational net model for process theories such as CCS and CSP that has a sound and complete justification for distinguishing processes. This requires a precise characterization of which nets are distinguishable by any external observer — who sees only *sequential* behavior — performing action refinement and the CCS/CSP-style operations. To this end, it is useful to develop a "fully abstract" denotational semantics that precisely captures these distinctions. In contrast to purely operational characterizations, which are implicitly quantified over an infinite number of contexts, fully abstract semantics have the advantage that they often lead to a semantic foundation for recursively-defined processes, logical foundations for proving equivalence of (possibly infinite-state) processes, and decision procedures for equivalence of finite-state processes.

Definition 1.0.1 A semantics, $[\![\cdot]\!]$, assigning to any process, P, a meaning, $[\![P]\!]$, is *compositional* for an operator on processes if semantic equality is a congruence for the operator, *i.e.*, the operator preserves semantic equality. We say that a semantics is *adequate* for an equivalence on processes if semantic equality implies process equivalence. Finally, we say that a semantics is *fully abstract* for a process equivalence *with respect to* a set of operators if semantic equality is the coarsest congruence for those operators that is adequate for the equivalence.

1.1 True Concurrency Semantics and Action Refinement on Petri Nets

A starting point for such an investigation is to apply state-machine-based equivalences to Petri Nets. It is well-known [10] that these equivalences, including partial traces, failures, bisimulation, and Testing-equivalence, are not compositional for even very simple forms of action refinement on Petri Nets, including those whose only effect is to "split" actions into two parts.

In a seminal paper [47], Vogler has developed a semantics for labeled, 1-safe Petri Nets that is compositional for certain simple "split" and "choice" refinements, and indeed is fully abstract for failures semantics [9] and Hennessy's MUST-experiments [19]. Furthermore, his semantics supports a full process theory involving CSP-style parallel process composition-with-communication, hiding, deadlock, and divergences (*cf.* [9, 19, 21, 30]). Vogler's semantics is based on "pomset-traces," which are a generalization of ordinary traces, *i.e.*, sequences of visible actions, to multi-sets of actions partially ordered to reflect causality and concurrency. In particular, his semantics consists of "interval pomset-failures": namely, pomset-traces with a certain "interval" order, paired with "failure sets" [8, 9, 21].

Vogler's elegant insight is that pomset-failures are not compositional for split refinements, since these refinements reveal "failure sets" of nets when transitions have "half-fired": that is, when all tokens have been removed from the preset of the transition but no tokens have been added to the postset. Vogler's technical solution is to specify some maximal events of pomset-traces to be "half-fired" and to keep track of the corresponding failure sets. The fully abstract semantics for non-divergent nets is obtained by performing certain closure operations and then restricting to interval orders. This is extended to a fully abstract semantics for divergent nets by additionally keeping track of "divergent" pomset-traces (with half-fired events), performing certain closure operations, and then again restricting to interval orders.

Although Vogler's insight about half-fired transitions is quite elegant, the "half-fired events" in his pomset-failures make the definition of his semantics and his proofs of compositionality quite difficult to understand. Furthermore, as Vogler points out, his "general pomset" semantics for divergent nets, *i.e.*, the intermediate semantics obtained before restricting to interval orders, is not compositional for split and choice refinements, and he states and leaves open [47, 49] the problem of identifying such a semantics that is compositional. As a consequence, his closure operations become rather technically complicated.

Vogler generalizes his simple split and choice refinements to allow a fairly large class of "refinement nets" required to satisfy some structural and behavioral conditions, which are rather technical and quite restrictive. His semantics is compositional with respect to each of the operators corresponding to his refinement nets. Namely, if two nets are equivalent under his semantics, then applying the *same* action refinement ρ to them yields semantically equivalent nets.

However, it is *not* the case that his semantics is compositional for nets as action refinement *operators*. For example, the nets a and $\tau.a$, where τ is the hidden action, are semantically equivalent as operands or *targets* of action refinement, but they behave differently when used as operators refining an action b, *viz.*,

$$\begin{array}{lrcl} & [\![a]\!] & =_{\text{Vogler}} & [\![\tau.a]\!]\,, \\ \text{but} & [\![(b+c)[b{:=}a]]\!] & \neq_{\text{Vogler}} & [\![(b+c)[b{:=}\tau.a]]\!]. \end{array}$$

In this thesis, we simplify and extend Vogler's results in a number of ways. We first present a general class of *Well Terminating (WT) Nets*, which are possibly infinite, safe nets with designated transitions for signaling successful termination. We then present WT Net operators corresponding to the familiar CCS/CSP operations of prefixing ($a.$), restriction ($\backslash a$), hiding ($-a$), renaming ($[f]$), CSP-style sequencing ($;$), non-communicating parallel composition ($||$), CSP-style parallel-composition-with-synchronization ($||_L$), CCS-style parallel-composition-with-hiding ($|$), internal choice, and CCS-style choice ($+_M$). All of our net operations are closely related to the corresponding CCS/CSP operators on labeled transition systems (*lts*'s).

The first main result of this thesis is that rather than keeping track of the technically cumbersome half-fired events, it is sufficient to first simply "duplicate-and-split" all the visible transitions in a net and then take the *ordinary* pomset-failures of the "duplicate-split" net. For divergent nets, one must also keep track of "pomset-divergences": namely, pomsets together with an explicit representation of the possibly concurrent divergences that are enabled. Performing some natural closure operations then yields a "general pomset" semantics, $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$, that is compositional for split refinements, choice refinements, and all of the CCS/CSP operators on WT Nets, and whose restriction, $[\![\cdot]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$, to interval pomsets is fully abstract for MUST-equivalence. We describe a similar fully abstract semantics, $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$, for MAY-equivalence [19] based on "pomset-traces"; the MAY- and MUST-semantics together provide a fully abstract semantics for Testing Equivalence [19].

Our semantics greatly simplify Vogler's representation by avoiding "half-fired" events; furthermore, keeping track of concurrent divergences simplifies the closure operations and yields compositionality of the $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$ semantics. This generalizes Vogler's results and solves the open problem mentioned earlier.

This thesis then presents a class of *Refinable Well-Terminating (RWT) Nets*, which form a large subclass of WT Nets that is closed under almost all of the WT operations, together with a definition of action refinement that allows *any* RWT net to be used as a target or operator of action refinement. The second main result of this thesis is that in contrast to Vogler's semantics, all of our semantics are compositional for RWT Nets as *targets and operators* of action refinement, with the $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$ and $[\![\cdot]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$ semantics remaining fully abstract for MAY- and

MUST-equivalence.

1.2 The Semantic Domains and Recursion

In order to ensure that our semantic theories support recursively-defined concurrent processes, we present an abstract characterization of all our semantics. Our semantic domains form algebraic complete partial orders in which all compact elements are definable as the meanings of WT Nets, and all of our operators are continuous functions.

1.3 Deciding True Concurrency Equivalences

The decision problem for finite-state concurrent processes under a variety of interleaving semantics has been widely studied in the literature, and the computation complexity has been tightly characterized [4, 26, 29, 34, 36]. In contrast, there have been essentially no results on the complexity of the decision problems for *true concurrency* equivalences on finite-state concurrent processes, and little is even known about the decidability of these equivalences. For example, decidability of such a basic true concurrency property as *pomset-trace* equivalence appears not to have been known.

One of the main results of this thesis is that pomset-trace equivalence is decidable for finite 1-safe Petri Nets, and is, in fact, complete for EXPSPACE. Furthermore, we show that the decision problem for history-preserving bisimulation [5, 35, 39, 44, 46] on finite 1-safe Petri nets is complete for DEXPTIME. History-preserving bisimulation had earlier been shown by Vogler [46] to be decidable; however, he left open its complexity.

In contrast to interleaving equivalences, the decidability of pomset-trace equivalence for finite nets does not obviously reduce to equivalence of finite automata. The difficulty is that the causal ordering in a pomset-trace depends *a priori* on the entire pomset-trace, which may be unboundedly large. Inspired by Vogler's decision procedure for history-preserving bisimulation, we show that there is in fact a bound on the required information. This idea leads to our decision procedure for pomset-trace equivalence, and a simple analysis of this procedure yields an EXPSPACE upper bound. The same approach also gives a DEXPTIME decision procedure for history-preserving bisimulation. Our lower bounds for these true concurrency equivalences follow easily by reductions from the corresponding interleaving equivalences [29, 34, 36].

Our methods also yield tight complexity bounds for about a dozen other true concurrency equivalences, several of which resolve open problems in the literature.

1.4 Outline of the Thesis

Chapter 2 presents our class of Well-Terminating Nets together with split refinements, choice refinements, and our CCS/CSP operators. A brief introduction to Hennessy's experiments, Testing-equivalence, partial trace semantics and failures semantics is given in Chapter 3. We then develop our true concurrency semantics for Well-Terminating Nets, prove that they are compositional for all our Net operators and adequate for MAY- and MUST-equivalence, and show that their "interval" restrictions are fully abstract. The corresponding semantic domains are developed in Chapter 4.

Chapter 5 presents our action refinement operator, and shows that our semantics are compositional and our semantic domains are closed under this operator. Our decidability results for true concurrency equivalences appear in Chapter 6. This chapter is self-contained, and hence repeats some earlier definitions. Chapter 7 concludes with a discussion of some further results, open problems, and future work.

Chapter 2

Well-Terminating Nets and Operations

2.1 Well-Terminating Nets

Throughout this thesis, we use the standard definitions (*cf.* [46]) of Petri Nets and their operational behavior. In order to keep this thesis relatively self-contained, we repeat them here:

Definition 2.1.1 A labeled Petri Net, N, is a triple $\langle S_N, T_N, Start_N \rangle$, where S_N is the set of places, T_N is the set of transitions, and $Start_N$ is the set of initially marked places (which contain "tokens"). Every transition, t, in T_N has a label, $l_N(t)$, a preset, $pre_N(t)$, and a post-set, $post_N(t)$. We refer to the label τ as the "hidden action", and refer to all labels other than τ as "visible actions". A transition is *visible* (hidden) iff its label is visible (hidden). For every place $s \in S_N$, we write $pre_N(s)$ and $post_N(s)$ to refer to its preset and post-set. We assume for expository simplicity that all transitions have non-empty presets, and that the initial marking is non-empty.

Transitions are represented graphically as horizontal bars, places are represented as circles, and tokens are represented as dots in these circles. The preset of a transition is the set of places from which there is an arrow to the transition; the post-set of a transition is the set of places to which there is an arrow from the transition. Dually, the preset (post-set) of a place is the set of transitions from (to) which there is an arrow to (from) the place.

A marking of a net is an assignment of a non-negative number of "tokens" to each place in the net. A transition, t, is *enabled* under a marking iff every place in the preset of t contains at least one token. If a transition t is enabled in a marking, then t can *fire* by removing a token from each place in its preset and placing a token into each place in its post-set.

A *firing sequence* of a net, N, is a possibly empty sequence, $t_1 \ldots t_k$, of transitions of N such that t_1 is enabled under the initial marking of N, and each t_i is successively enabled in the marking resulting from firing $t_1 \ldots t_{i-1}$. A *run* is a finite firing sequence. The *reachable markings* of a net are exactly those markings that result from firing some run. A net is *1-safe* iff every place contains at most one token under any reachable marking. Rather than being represented as a function from places to non-negative integers, a marking of a 1-safe net can be written as the set of places that contain a token.

A pair of transitions, t and t', can *fire concurrently* in a 1-safe net iff the union of the preset and post-set of t is disjoint from that of t' and there is a reachable marking in which both t and t' are enabled.

Our class of "Well-Terminating" Nets is related to the class of CSP processes that signal successful termination by performing a distinguished action, $\surd$. In a similar manner, our well-terminating nets signal successful termination by firing any transition labeled with $\surd$. In order to ensure that the net has actually terminated, we require that all places in the net be unmarked after any $\surd$-labeled transition fires. We wish to restrict our attention to labeled, 1-safe nets with "computable behavior," and we thus impose some syntactic and behavioral restrictions that guarantee finite-markings and finite-branching of the underlying transition system.

Definition 2.1.2 The class of *Well-Terminating (WT) Nets* consists of pairs $\langle N, Act\rangle$ such that Act is a finite set of visible labels containing $\surd$ and N is a 1-safe, possibly infinite Petri net, all of whose transitions are labeled with actions in $Act \cup \{\tau\}$. Furthermore, N must satisfy the following properties:

- The initial marking is finite.
- The preset and post-set of every transition is finite.
- Only a finite number of transitions are enabled under any reachable marking.
- All places are unmarked immediately after any $\surd$-labeled transition fires. This condition must be satisfied in every reachable marking.

We note that these conditions together imply that all reachable markings are finite and that nets have only finite concurrency. The condition on the $\surd$-transitions ensures that no transition (not even a $\surd$-transition) can be fired concurrently with, or following, a $\surd$-transition.

Our $\surd$-labeled transitions serve to distinguish deadlock from successful termination. We say that a net *successfully terminates* when a $\surd$-labeled transition fires, while a net is *deadlocked* exactly when no transition is enabled. The $\surd$-action plays a distinguished role in our theory, and our net operators are defined in a way that respects this distinguished role.

WT Nets form natural isomorphism classes:

Definition 2.1.3 Let $\langle N_1, Act\rangle$ and $\langle N_2, Act\rangle$ be WT Nets over a common alphabet, Act. Then $\langle N_1, Act\rangle$ and $\langle N_2, Act\rangle$ are *isomorphic* iff there is a bijection f from S_{N_1} to S_{N_2} and a bijection g from T_{N_1} to T_{N_2} such that $Start_{N_2} = f(Start_{N_1})$, and $l_{N_2}(g(t)) = l_{N_1}(t)$, $pre_{N_2}(g(t)) = f(pre_{N_1}(t))$, and $post_{N_2}(g(t)) = f(post_{N_1}(t))$ for every $t \in T_{N_1}$.

In order to view WT Nets as an operational model for CCS and CSP, we will find it useful to represent the behavior of nets as labeled transition systems. The following definition is standard and is essentially taken verbatim from [19].

Definition 2.1.4 A *labeled transition system (lts)* is a triple $\langle S, Act, \longrightarrow, s_{\text{init}}\rangle$, where

- S is a set of states containing s_{init}.
- Act is a set of labels.

- $\longrightarrow$ is a relation in $S \times Act \times S$.
- s_{init} is designated as the "initial state" in S.

We write $s \xrightarrow{a} s'$ in place of $(s, a, s') \in \longrightarrow$. The relations $\xrightarrow{a}$ are extended to relations $\xrightarrow{v}$, for every $v \in Act^*$, in the obvious way:

1. $s \xrightarrow{\varepsilon} s'$ iff s' is s
2. $s \xrightarrow{av} s'$ iff $s \xrightarrow{a} s''$ for some s'' such that $s'' \xrightarrow{v} s'$.

This means $s \xrightarrow{v} s'$ if s can evolve to s' by performing the sequence of actions v. We also write $s \xrightarrow{v}$ to mean that there exists a s' such that $s \xrightarrow{v} s'$. We say that an action, a, is *enabled* at a state, s, iff $s \xrightarrow{a}$.

If Act contains the label τ, these relations are generalized as follows, for every $v \in Act^*$:

1. $s \stackrel{a}{\Longrightarrow} s'$ iff $s \xrightarrow{\tau^i} s_1 \xrightarrow{a} s_2 \xrightarrow{\tau^j} s'$ for some states s_1, s_2 and some $i, j \geq 0$
2. $s \stackrel{\varepsilon}{\Longrightarrow} s'$ iff $s \xrightarrow{\tau^k} s'$ for some $k \geq 0$
3. $s \stackrel{av}{\Longrightarrow} s'$ iff $s \stackrel{a}{\Longrightarrow} s''$ for some s'' such that $s'' \stackrel{v}{\Longrightarrow} s'$.

This means $s \stackrel{v}{\Longrightarrow} s'$ if s can evolve to s' by performing the sequence of actions v, possibly interspersed with τ-actions. We also write $s \stackrel{v}{\Longrightarrow}$ to mean that there exists a s' such that $s \stackrel{v}{\Longrightarrow} s'$.

The following definition is essentially standard (*cf.* [33]):

Definition 2.1.5 The *labeled transition system* of a WT Net $\langle N, Act\rangle$, written $lts(\langle N, Act\rangle)$, is the labeled transition system over $Act \cup \{\tau\}$ whose states are the reachable markings of N and whose labeled transitions correspond to firings of single transitions of N. In particular, state M goes to state M' via an a-labeled transition in $lts(\langle N, Act\rangle)$ iff marking M' of N can be reached from marking M by firing exactly some a-labeled transition of N. The initial state of $lts(\langle N, Act\rangle)$ is defined to be the initial marking of N.

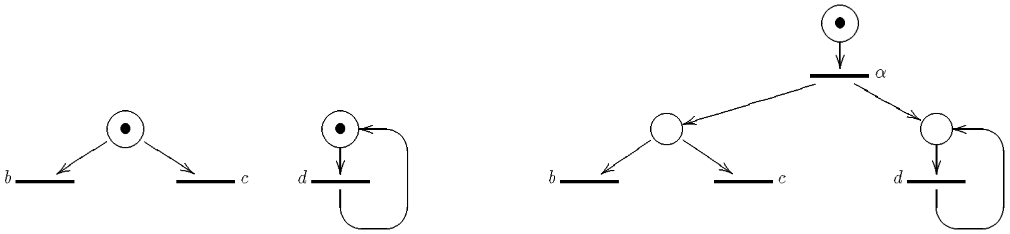
We note that Definition 2.1.2 ensures that the labeled transition system of every WT Net is finitely-branching.

2.2 Operations on Well-Terminating Nets

This section defines WT Net operators corresponding to the familiar CCS/CSP operations of prefixing ($\alpha.$), restriction ($\backslash a$), hiding ($-a$), renaming ($[f]$), CSP-style sequencing ($;$), non-communicating parallel composition ($||$), CSP-style parallel-composition-with-synchronization ($\|_L$), CCS-style parallel-composition-with-hiding ($|$), internal choice ($\oplus$), and CCS-style choice ($+_M$). We also define $split_{(a,a_+,a_-)}$ and $choice_{(a,a_L,a_R)}$ refinement operators on WT Nets.

We begin by defining operators that grow or shrink the alphabet of nets:

Definition 2.2.1 Let $\langle N, Act\rangle$ be a WT Net, and let Act' be a finite set of visible labels. Then $\langle N, Act\rangle$ *grow* $Act' \stackrel{\text{def}}{=} \langle N, Act \cup Act'\rangle$.



$\langle N, Act \rangle$ $\alpha.\langle N, Act \rangle$

Figure 2-1: An Example of Prefixing

Definition 2.2.2 Let $\langle N, Act \rangle$ be a WT Net, and let $Act' \subseteq Act$ be a finite set of visible labels containing $\surd$. Then $\langle N, Act \rangle$ *shrink* $Act' = \langle P, Act' \rangle$, where P is identical to N except that all (visible) transitions with labels from $Act - Act'$ are removed. In particular, $T_P = \{t \in T_N : l_N(t) \in Act' \cup \{\tau\}\}$.

The hiding and renaming operators simply relabel transitions:

Definition 2.2.3 Let $\langle N, Act \rangle$ be a WT Net, and let a be a label in $Act - \{\surd\}$. Then $\langle N, Act \rangle - a = \langle P, Act \rangle$, where P is identical to N except that all a-labeled transitions are relabeled with τ.

Definition 2.2.4 Let $\langle N, Act \rangle$ be a WT Net, and let f be function from Act to Act such that for all $\beta \in Act$, $f(\beta) = \surd$ iff $\beta = \surd$. Then $\langle N, Act \rangle[f] = \langle P, Act \rangle$, where P is identical to N except that $l_P = f \circ l_N$.

The restriction operator simply removes transitions:

Definition 2.2.5 Let $\langle N, Act \rangle$ be a WT Net, and let a be a label in $Act - \{\surd\}$. Then $\langle N, Act \rangle \backslash a = \langle P, Act \rangle$, where P is identical to N except that all a-labeled transitions are removed. In particular, $T_P = \{t \in T_N : l_N(t) \neq a\}$.

The prefixing operator ($\alpha.$), illustrated in Figure 2-1, simply attaches a new place and a new α-labeled transition to the "start" of a net:

Definition 2.2.6 Let $\langle N, Act \rangle$ be a WT Net, and let α be a label in $(Act \cup \{\tau\}) - \{\surd\}$. Then $\langle P, Act \rangle = \alpha.\langle N, Act \rangle$ is defined as:

$$S_P = S_N \uplus \{s_\alpha\}$$

$$T_P = T_N \uplus \{t_\alpha\}$$

$$pre_P(t) = \begin{cases} \{s_\alpha\} & \text{if } t = t_\alpha \\ pre_N(t) & \text{otherwise} \end{cases}$$

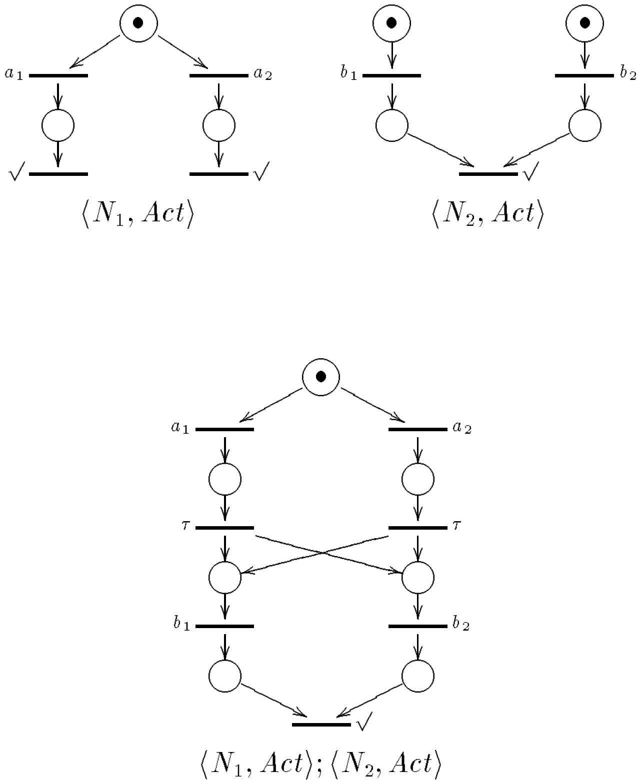


Figure 2-2: An Example of Sequencing

$$post_P(t) = \begin{cases} Start_N & \text{if } t = t_\alpha \\ post_N(t) & \text{otherwise} \end{cases}$$

$$l_P(t) = \begin{cases} \alpha & \text{if } t = t_\alpha \\ l_N(t) & \text{otherwise} \end{cases}$$

$$Start_P = \{s_\alpha\}$$

Our sequencing operator $\langle N_1, Act\rangle; \langle N_2, Act\rangle$ makes critical use of the $\surd$-transitions of N_1 by relabeling them with τ and using them as a hidden (τ-labeled) signal to transfer control to N_2. We illustrate the definition of "sequencing" through the following simple example. Suppose that we are given the WT nets $\langle N_1, Act\rangle$ and $\langle N_2, Act\rangle$ of Figure 2-2, and we want to define $\langle N_1, Act\rangle; \langle N_2, Act\rangle$. We want the firing of either of the $\surd$-transitions of N_1 to be a hidden signal that enables both b_1 and b_2 to fire concurrently. Therefore, we relabel the $\surd$-transitions of N_1 to τ, and then have both of these τ-transitions feed into both of the start places of N_2. The resulting net $\langle N_1, Act\rangle; \langle N_2, Act\rangle$ is given in Figure 2-2. The formal definition appears below.

Definition 2.2.7 Let $\langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets with disjoint sets of places and transitions and with a common alphabet, Act. Then $\langle P, Act\rangle = \langle N_1, Act\rangle; \langle N_2, Act\rangle$ is defined as:

$$S_P = S_{N_1} \cup S_{N_2}$$

$$T_P = T_{N_1} \cup T_{N_2}$$

$$pre_P(t) = \begin{cases} pre_{N_1}(t) & \text{if } t \in T_{N_1} \\ pre_{N_2}(t) & \text{if } t \in T_{N_2} \end{cases}$$

$$post_P(t) = \begin{cases} post_{N_1}(t) & \text{if } t \in T_{N_1} \text{ and } l_{N_1}(t) \neq \surd \\ Start_{N_2} & \text{if } t \in T_{N_1} \text{ and } l_{N_1}(t) = \surd \\ post_{N_2}(t) & \text{if } t \in T_{N_2} \end{cases}$$

$$l_P(t) = \begin{cases} l_{N_1}(t) & \text{if } t \in T_{N_1} \text{ and } l_{N_1}(t) \neq \surd \\ \tau & \text{if } t \in T_{N_1} \text{ and } l_{N_1}(t) = \surd \\ l_{N_2}(t) & \text{if } t \in T_{N_2} \end{cases}$$

$$Start_P = Start_{N_1}$$

Our non-communicating parallel composition operator $||$, places two nets in parallel. In order to preserve the well-terminating property associated with $\surd$-transitions, the nets are required synchronize on the $\surd$-action. Our definition is illustrated in Figure 2-3.

Definition 2.2.8 Let $\langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets with disjoint sets of places and transitions and with a common alphabet, Act.

Then $\langle P, Act\rangle = \langle N_1, Act\rangle \mid\mid \langle N_2, Act\rangle$ is defined as:

$$S_P = S_{N_1} \cup S_{N_2}$$

$$\begin{aligned} T_P = {} & \{(t_1, t_2) \in T_{N_1} \times T_{N_2} : l_{N_1}(t_1) = l_{N_2}(t_2) = \surd\} \uplus \\ & \{(t, *) \in T_{N_1} \times \{*\} : l_{N_1}(t) \neq \surd\} \uplus \{(*, t) \in \{*\} \times T_{N_2} : l_{N_2}(t) \neq \surd\} \end{aligned}$$

$$\begin{aligned} pre_P((t_1, t_2)) &= pre_{N_1}(t_1) \cup pre_{N_2}(t_2) \\ pre_P((t, *)) &= pre_{N_1}(t) \\ pre_P((*, t)) &= pre_{N_2}(t) \end{aligned}$$

$$\begin{aligned} post_P((t_1, t_2)) &= post_{N_1}(t_1) \cup post_{N_2}(t_2) \\ post_P((t, *)) &= post_{N_1}(t) \\ post_P((*, t)) &= post_{N_2}(t) \end{aligned}$$

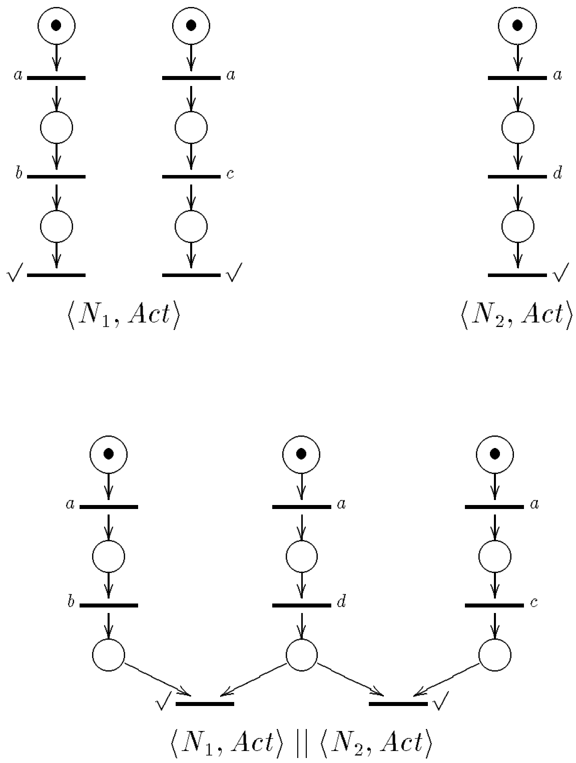


Figure 2-3: An Example of Non-communicating Parallel Composition

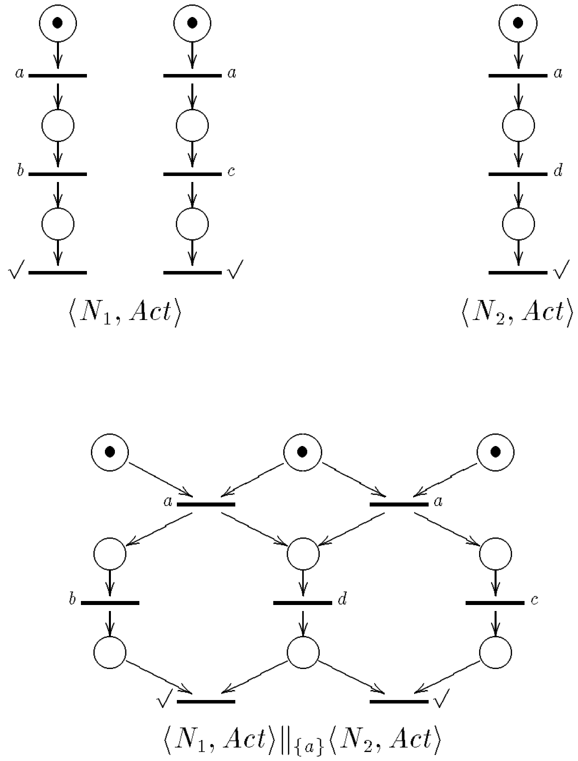


Figure 2-4: An Example of CSP-style Parallel Composition

$$
\begin{aligned}
l_P((t_1, t_2)) &= \surd \\
l_P((t, *)) &= l_{N_1}(t) \\
l_P((*, t)) &= l_{N_2}(t)
\end{aligned}
$$

$$Start_P = Start_{N_1} \cup Start_{N_2}$$

We also have a family of CSP-style parallel composition operators $\|_L$, where L is a set of visible labels. This operator places two nets in parallel and requires them to synchronize on all actions in the set $L \cup \{\surd\}$. In particular, the non-communicating parallel composition operator is definable as $\|_\emptyset$.

Our definition is essentially the same as [47], and is illustrated in Figure 2-4.

Definition 2.2.9 Let $\langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets with disjoint sets of places and transitions and with a common alphabet, Act. Let $L \subseteq Act$, and let $L_\surd = L \cup \{\surd\}$.

Then $\langle P, Act\rangle = \langle N_1, Act\rangle \|_L \langle N_2, Act\rangle$ is defined as:

$$
\begin{aligned}
S_P &= S_{N_1} \cup S_{N_2} \\[1em]
T_P &= \{(t_1, t_2) \in T_{N_1} \times T_{N_2} : l_{N_1}(t_1) = l_{N_2}(t_2) \text{ and } l_{N_1}(t_1) \in L_{\surd}\} \uplus \\
&\quad\; \{(t, *) \in T_{N_1} \times \{*\} : l_{N_1}(t) \notin L_{\surd}\} \uplus \{(*, t) \in \{*\} \times T_{N_2} : l_{N_2}(t) \notin L_{\surd}\} \\[1em]
pre_P((t_1, t_2)) &= pre_{N_1}(t_1) \cup pre_{N_2}(t_2) \\
pre_P((t, *)) &= pre_{N_1}(t) \\
pre_P((*, t)) &= pre_{N_2}(t) \\[1em]
post_P((t_1, t_2)) &= post_{N_1}(t_1) \cup post_{N_2}(t_2) \\
post_P((t, *)) &= post_{N_1}(t) \\
post_P((*, t)) &= post_{N_2}(t) \\[1em]
l_P((t_1, t_2)) &= l_{N_1}(t_1) \\
l_P((t, *)) &= l_{N_1}(t) \\
l_P((*, t)) &= l_{N_2}(t) \\[1em]
Start_P &= Start_{N_1} \cup Start_{N_2}
\end{aligned}
$$

Similar to [15], we also have a CCS-style parallel composition operator $|$, where two nets are placed in parallel and are allowed to perform hidden synchronizations on all complementary actions $a, \overline{a}$; however, they must (visibly) synchronize on the $\surd$ action. Our definition is illustrated in Figure 2-5.

We say that an alphabet Act is *closed under complementation* iff for all labels, $a \in Act$ implies that $\overline{a} \in Act$, where $\overline{\overline{a}} \stackrel{\text{def}}{=} a$.

Definition 2.2.10 Let $\langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets with disjoint sets of places and transitions and with a common alphabet, Act, such that $Act - \{\surd\}$ is closed under complementation. Then $\langle P, Act\rangle = \langle N_1, Act\rangle \mid \langle N_2, Act\rangle$ is defined as:

$$
\begin{aligned}
S_P &= S_{N_1} \cup S_{N_2} \\[1em]
T_P &= \{t \in T_{N_1} : l_{N_1}(t) \neq \surd\} \uplus \{t \in T_{N_2} : l_{N_2}(t) \neq \surd\} \uplus \\
&\quad\; \{(t_1, t_2) \in T_{N_1} \times T_{N_2} : l_{N_1}(t_1) = \overline{l_N(t_2)} \text{ or } l_{N_1}(t_1) = l_{N_2}(t_2) = \surd\} \\[1em]
pre_P((t_1, t_2)) &= pre_{N_1}(t_1) \cup pre_{N_2}(t_2) \\
pre_P((t, *)) &= pre_{N_1}(t)
\end{aligned}
$$

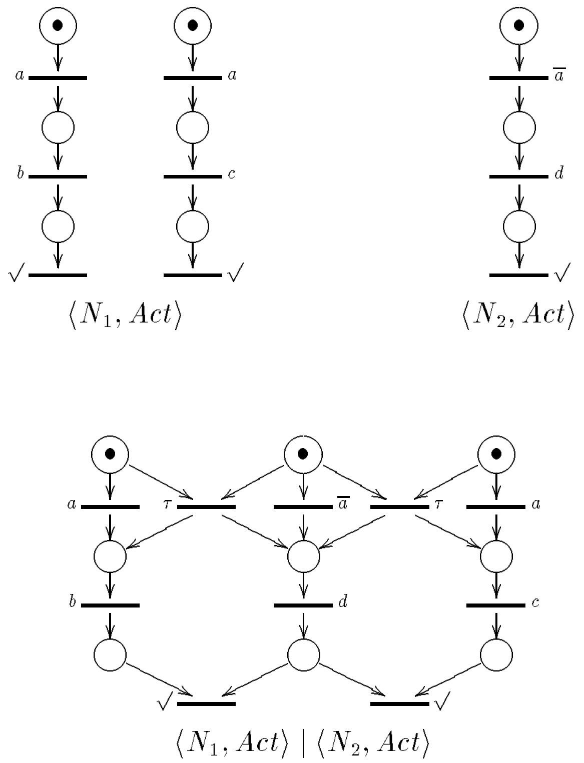


Figure 2-5: An Example of CCS-style Parallel Composition

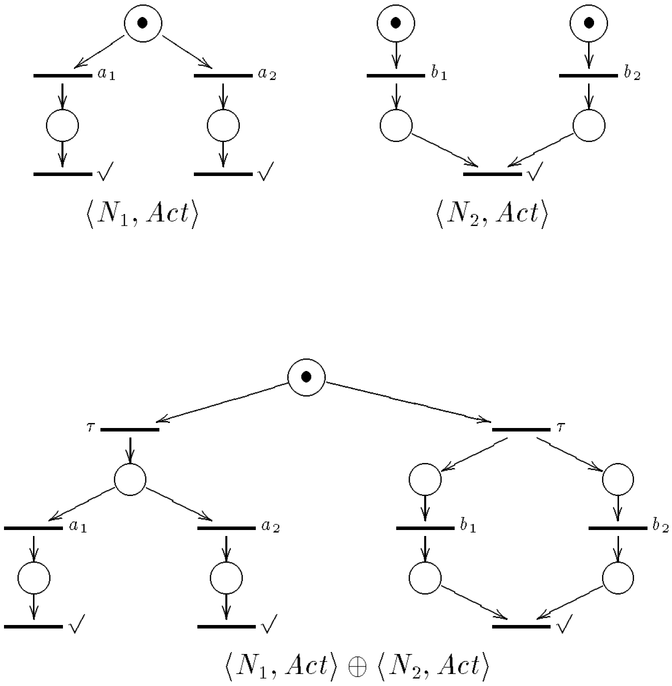


Figure 2-6: An Example of Internal Choice

$$pre_P((*, t)) = pre_{N_2}(t)$$

$$\begin{aligned}
post_P((t_1, t_2)) &= post_{N_1}(t_1) \cup post_{N_2}(t_2) \\
post_P((t, *)) &= post_{N_1}(t) \\
post_P((*, t)) &= post_{N_2}(t)
\end{aligned}$$

$$\begin{aligned}
l_P((t_1, t_2)) &= \begin{cases} \surd & \text{if } l_{N_1}(t_1) = l_{N_2}(t_2) = \surd \\ \tau & \text{otherwise} \end{cases} \\
l_P((t, *)) &= l_{N_1}(t) \\
l_P((*, t)) &= l_{N_2}(t)
\end{aligned}$$

$$Start_P = Start_{N_1} \cup Start_{N_2}$$

We now define the internal choice operator, illustrated in Figure 2-6, which corresponds to prefixing each net with τ, and then "merging" the resulting (necessarily exactly two) initially marked places:

Definition 2.2.11 Let $\langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets with disjoint sets of places and transitions and with a common alphabet, Act. Then $\langle P, Act\rangle = \langle N_1, Act\rangle \oplus \langle N_2, Act\rangle$ is defined as:

$$S_P = S_{N_1} \uplus S_{N_2} \uplus \{s\}$$

$$T_P = T_{N_1} \uplus T_{N_2} \uplus \{t_1, t_2\}, \text{ where } t_1, t_2 \text{ are distinct}$$

$$pre_P(t) = \begin{cases} \{s\} & \text{if } t = t_1 \text{ or } t = t_2 \\ pre_{N_1}(t) & \text{if } t \in T_{N_1} \\ pre_{N_2}(t) & \text{if } t \in T_{N_2} \end{cases}$$

$$post_P(t) = \begin{cases} Start_{N_1} & \text{if } t = t_1 \\ Start_{N_2} & \text{if } t = t_2 \\ post_{N_1}(t) & \text{if } t \in T_{N_1} \\ post_{N_2}(t) & \text{if } t \in T_{N_2} \end{cases}$$

$$l_P(t) = \begin{cases} \tau & \text{if } t = t_1 \text{ or } t = t_2 \\ l_{N_1}(t) & \text{if } t \in T_{N_1} \\ l_{N_2}(t) & \text{if } t \in T_{N_2} \end{cases}$$

$$Start_P = \{s\}$$

We also wish to define the Milner's CCS choice operator, $+_M$, which allows *non-deterministic choice* between two nets. We illustrate the definition of $+_M$ through the following simple example. Suppose that we are given the WT nets $\langle N_1, Act\rangle$ and $\langle N_2, Act\rangle$ of Figure 2-7, and we want to define $\langle N_1, Act\rangle +_M \langle N_2, Act\rangle$. Clearly, we want to introduce conflicts between the a_i and the b_j but preserve the concurrency within the b_j, and so we do a simple cross product construction on the start places of both nets. We note that this causes all the $\surd$-labeled transitions to be in conflict, as desired. The resulting net is given in Figure 2-7.

As discussed in [42], one technical complication arises due to initially marked places that have incoming transitions, and in general, we apply a *start-unwinding* operator on nets before doing the above construction. Our start-unwinding operator, illustrated in Figure 2-8, is essentially the same as that of [15, 42] and produces a net that is "essentially the same"[1] as the original net, except that all initially marked places have empty presets. The "start-unwound" net is identical to the original net whenever all initially marked places of the original net have empty presets.

Definition 2.2.12 Let $\langle N, Act\rangle$ be a WT net, and let $Start\text{-}cyclic_N$ be the initially marked places of N that have non-empty presets, *i.e.*, $Start\text{-}cyclic_N = \{s \in Start_N : pre_N(s) \neq \emptyset\}$. Then $\langle P, Act\rangle = start\text{-}unwind(\langle N, Act\rangle)$ is defined as:

$$S_P = S_N \uplus \{(*, s) : s \in Start\text{-}cyclic_N\}$$

[1]The resulting net is strongly history-preserving bisimilar [39] to the original net.

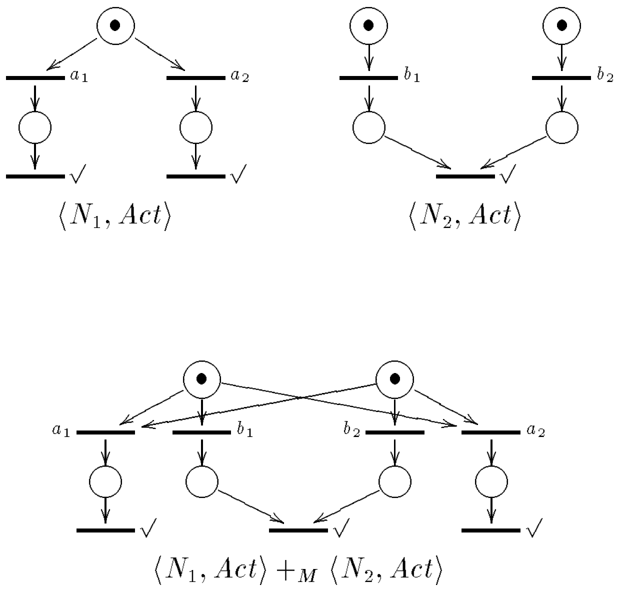


Figure 2-7: An Example of CCS-style Choice

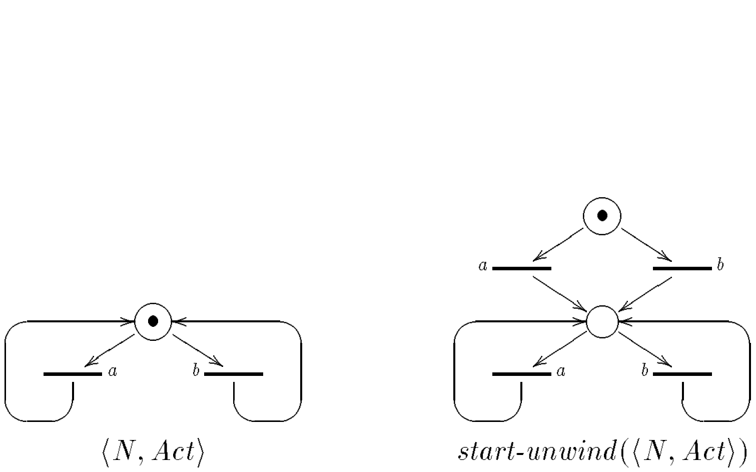


Figure 2-8: An Example of Start-Unwinding

$$T_P = T_N \uplus \{\langle U, t\rangle : t \in T_N, U \neq \emptyset, \text{ and } U \subseteq \mathit{Start\text{-}cyclic}_N \cap \mathit{pre}_N(t)\}$$

$$\begin{aligned} \mathit{pre}_P(t) &= \mathit{pre}_N(t) \\ \mathit{pre}_P(\langle U, t\rangle) &= (\mathit{pre}_N(t) - U) \cup \{(*, s) : s \in U\} \end{aligned}$$

$$\begin{aligned} \mathit{post}_P(t) &= \mathit{post}_N(t) \\ \mathit{post}_P(\langle U, t\rangle) &= \mathit{post}_N(t) \end{aligned}$$

$$\begin{aligned} l_P(t) &= l_N(t) \\ l_P(\langle U, t\rangle) &= l_N(t) \end{aligned}$$

$$\mathit{Start}_P = (\mathit{Start}_N - \mathit{Start\text{-}cyclic}_N) \cup \{(*, s) : s \in \mathit{Start\text{-}cyclic}_N\}$$

Using this start-unwinding operator, we now define the $+_M$ operator on nets.

Definition 2.2.13 Let $\langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets with disjoint sets of places and transitions and with a common alphabet, Act. Let $\langle N_1', Act\rangle$ and $\langle N_2', Act\rangle$ be $\mathit{start\text{-}unwind}(\langle N_1, Act\rangle)$ and $\mathit{start\text{-}unwind}(\langle N_2, Act\rangle)$, respectively. Then $\langle P, Act\rangle = \langle N_1, Act\rangle +_M \langle N_2, Act\rangle$ is defined as:

$$S_P = (S_{N_1'} - \mathit{Start}_{N_1'}) \uplus (S_{N_2'} - \mathit{Start}_{N_2'}) \uplus (\mathit{Start}_{N_1'} \times \mathit{Start}_{N_2'})$$

$$T_P = T_{N_1'} \cup T_{N_2'}$$

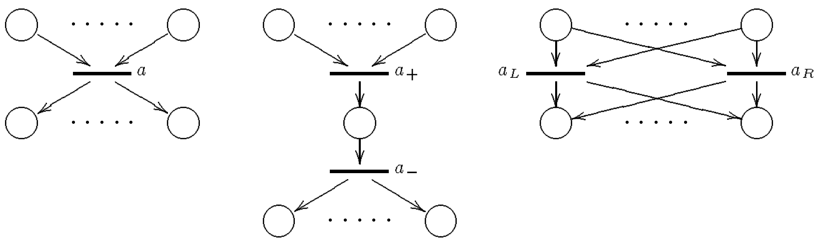
$$\mathit{pre}_P(t) = \begin{cases} (\mathit{pre}_{N_1'}(t) - \mathit{Start}_{N_1'}) \cup \{(s_1, s_2) \in S_P : s_1 \in \mathit{pre}_{N_1'}(t)\} & \text{if } t \in T_{N_1'} \\ (\mathit{pre}_{N_2'}(t) - \mathit{Start}_{N_2'}) \cup \{(s_1, s_2) \in S_P : s_2 \in \mathit{pre}_{N_2'}(t)\} & \text{if } t \in T_{N_2'} \end{cases}$$

$$\mathit{post}_P(t) = \begin{cases} \mathit{post}_{N_1'}(t) & \text{if } t \in T_{N_1'} \\ \mathit{post}_{N_2'}(t) & \text{if } t \in T_{N_2'} \end{cases}$$

$$l_P(t) = \begin{cases} l_{N_1'}(t) & \text{if } t \in T_{N_1'} \\ l_{N_2'}(t) & \text{if } t \in T_{N_2'} \end{cases}$$

$$\mathit{Start}_P = \mathit{Start}_{N_1'} \times \mathit{Start}_{N_2'}$$

Two other simple WT net operators play a significant role in our technical development. Namely, *split refinements* ($\mathit{split}_{(a,a_+,a_-)}$) replace every a-labeled transition by two consecutive transitions labeled a_+ and a_-, and *choice refinements* ($\mathit{choice}_{(a,a_L,a_R)}$) replace every a-labeled transition by two conflicting transitions labeled a_L and a_R. Figure 2-9 gives examples of these kinds of refinements.



$\langle N, Act\rangle$ $\quad$ $split_{(a,a_+,a_-)}(\langle N, Act\rangle)$ $\quad$ $choice_{(a,a_L,a_R)}(\langle N, Act\rangle)$

Figure 2-9: Split Refinements and Choice Refinements

Definition 2.2.14 Let $\langle N, Act\rangle$ be a WT Net, and let $a, a_+, a_- \in Act - \{\surd\}$. Then $\langle P, Act\rangle = split_{(a,a_+,a_-)}(\langle N, Act\rangle)$ is defined as:

$$S_P = S_N \uplus \{(*, t) : t \in T_N \text{ and } l_N(t) = a\}$$

$$T_P = \{t \in T_N : l_N(t) \neq a\} \uplus \{(t, +), (t, -) : t \in T_N \text{ and } l_N(t) = a\}$$

$$\begin{aligned} pre_P(t) &= pre_N(t) \\ pre_P((t,+)) &= pre_N(t) \\ pre_P((t,-)) &= \{(*, t)\} \end{aligned}$$

$$\begin{aligned} post_P(t) &= post_N(t) \\ post_P((t,+)) &= \{(*, t)\} \\ post_P((t,-)) &= post_N(t) \end{aligned}$$

$$\begin{aligned} l_P(t) &= l_N(t) \\ l_P((t,+)) &= a_+ \\ l_P((t,-)) &= a_- \end{aligned}$$

$$Start_P = Start_N$$

Definition 2.2.15 Let $\langle N, Act\rangle$ be a WT Net, and let $a, a_L, a_R \in Act - \{\surd\}$. Then $\langle P, Act\rangle = choice_{(a,a_L,a_R)}(\langle N, Act\rangle)$ is defined as:

$$S_P = S_N$$

$$T_P = \{t \in T_N : l_N(t) \neq a\} \uplus \{(t, L), (t, R) : t \in T_N \text{ and } l_N(t) = a\}$$

$$
\begin{aligned}
pre_P(t) &= pre_N(t) \\
pre_P((t, L)) &= pre_N(t) \\
pre_P((t, R)) &= pre_N(t) \\
\\
post_P(t) &= post_N(t) \\
post_P((t, L)) &= post_N(t) \\
post_P((t, R)) &= post_N(t) \\
\\
l_P(t) &= l_N(t) \\
l_P((t, L)) &= a_L \\
l_P((t, R)) &= a_R \\
\\
Start_P &= Start_N
\end{aligned}
$$

The following theorems show that the class of WT Nets is closed under all of the operators, and that we are justified in referring to the WT Net operators as "CCS/CSP-style" operators.

Theorem 2.2.16 The class of WT Nets is closed under prefixing ($\alpha.$), restriction ($\backslash a$), hiding ($-a$), renaming ($[f]$), CSP-style sequencing ($;$), non-communicating parallel-composition ($||$), CSP-style parallel-composition-with-synchronization ($\|_L$), CCS-style parallel-composition-with-hiding ($|$), internal choice ($\oplus$), start-unwinding, CCS-style choice ($+_M$), *split*, and *choice*.

Proof. The proof is very straightforward but tedious. As an illustration, we prove the case for start-unwinding; the remaining cases are left to the reader.

Let $\langle N, Act\rangle$ be a WT Net and let $\langle P, Act\rangle = \textit{start-unwind}(\langle N, Act\rangle)$. It is easy to see that all transitions in P have labels from $Act \cup \{\tau\}$, $Start_P$ is finite, and every transition in P has finite in-degree and out-degree.

A straightforward inductive argument shows that if $t'_1 \dots t'_k$ is a run of P resulting in the marking M' of P then:

- $t_1 \dots t_k$ is a run of N, where $t'_i = t_i$ if $t'_i \in T_N$, and $t'_i = \langle U, t_i\rangle$ for some U otherwise.
- The marking reached firing after $t_1 \dots t_k$ in N is given by the function M, where $M(s) = M'(s)$ for every $s \in S_N - \textit{Start-cyclic}_N$ and $M(s) = M'(s) + M'((*, s))$ for every $s \in \textit{Start-cyclic}$.

It is then easy to see from the definition of *start-unwind* that P is 1-safe, only a finite number of transitions are enabled under any reachable marking of P, and that all places in P are unmarked immediately after any $\surd$-labeled transition fires; hence $\langle P, Act\rangle$ is a WT Net. ■

Except for the parallel composition operators, all of our net operations are closely related to the corresponding CCS/CSP operators on labeled transition systems (*lts*'s), *cf.* [7, 30]. In particular:

Theorem 2.2.17 For all the CCS/CSP WT net-operators other than $||$, $||_L$, and $|$, the *lts* of the constructed net is strongly bisimilar to the *lts* obtained by applying the corresponding CCS/CSP *lts*-operator to the *lts*'s of the component nets. Also, $lts(\langle N_1, Act\rangle ||_L \langle N_2, Act\rangle)$ is strongly bisimilar to $lts(\langle N_1, Act\rangle)||_{L\cup\{\surd\}} lts(\langle N_2, Act\rangle)$ and $lts(\langle N_1, Act\rangle \; || \; \langle N_2, Act\rangle)$ is strongly bisimilar to $lts(\langle N_1, Act\rangle)||_{\{\surd\}} lts(\langle N_2, Act\rangle)$. Lastly, $lts(\langle N_1, Act\rangle \mid \langle N_2, Act\rangle)$ is strongly bisimilar to $lts(\langle N_1, Act\rangle) \mid lts(\langle N_2, Act\rangle)$, except that visible synchronization is required on $\surd$-actions.

Proof. As an illustration, we prove the case for $+_M$. The remaining cases are straightforward but tedious and are left to the reader.

We first prove that for all WT Nets N, the *lts* of $\langle N, Act\rangle$ is strongly bisimilar to the *lts* of *start-unwind*$(\langle N, Act\rangle)$. Let

$$\begin{aligned}\mathcal{B} = \{(M, M')\colon\ & M' \text{ is a reachable marking of } \textit{start-unwind}(\langle N, Act\rangle),\\ & M(s) = M'(s) \text{ for all } s \in S_N - \textit{Start-cyclic}_N,\\ & \text{and } M(s) = M'(s) + M'((*, s)) \text{ for all } s \in \textit{Start-cyclic}_N\}\end{aligned}$$

Using an argument similar to that in the proof of Theorem 2.2.16, it is straightforward to show that $\mathcal{B}$ is a bisimulation between the *lts* of $\langle N, Act\rangle$ and the *lts* of *start-unwind*$(\langle N, Act\rangle)$.

Let $\langle N_1', Act\rangle$ and $\langle N_2', Act\rangle$ be *start-unwind*$(\langle N_1, Act\rangle)$ and *start-unwind*$(\langle N_2, Act\rangle)$, respectively, and let

$$\begin{aligned}\mathcal{C} = \{(M, M')\colon\ & M' \text{ is a reachable marking of } \langle N_1, Act\rangle +_M \langle N_2, Act\rangle \text{ and}\\ & M = \{s_1 \in \textit{Start}_{N_1'} : (s_1, s_2) \in M' \text{ for all } s_2 \in \textit{Start}_{N_2'}\}\\ & \quad \cup \{s_2 \in \textit{Start}_{N_2'} : (s_1, s_2) \in M' \text{ for all } s_1 \in \textit{Start}_{N_1'}\}\\ & \quad \cup (M' \cap ((S_{N_1'} - \textit{Start}_{N_1'}) \cup (S_{N_2'} - \textit{Start}_{N_2'})))\}\end{aligned}$$

We observe that since $\textit{Start}_{N_1'}$ and $\textit{Start}_{N_2'}$ have empty presets, the definition of $+_M$ ensures that firing any initial transition of N_1' in $\langle N_1, Act\rangle +_M lts(\langle N_2, Act\rangle)$ will disable all transitions of N_2', and vice-versa. We further observe that for any reachable marking of $\langle N_1, Act\rangle +_M \langle N_2, Act\rangle$ and any $s_1 \in \textit{Start}_{N_1'}$, if some place (s_1, s_2) is empty while some place (s_1, s_2') contains a token, then a transition of N_2' must have fired, and vice-versa. It is then straightforward to show from the definition of $+_M$ on nets and labeled transition systems that $\mathcal{C}$ is a strong bisimulation between $lts(\langle N_1', Act\rangle) +_M lts(\langle N_2', Act\rangle)$ and the *lts* of $\langle N_1, Act\rangle +_M lts(\langle N_2, Act\rangle)$. The details are left to the reader.

Since strong bisimulation is a congruence with respect to $+_M$ (*cf.* [30]), the presence of $\mathcal{C}$ together with the above fact about start-unwinding immediately implies that $lts(\langle N_1, Act\rangle) +_M lts(\langle N_2, Act\rangle)$ is strongly bisimilar to the *lts* of $\langle N_1, Act\rangle +_M \langle N_2, Act\rangle$. ∎

The following propositions show that internal choice and CCS-style parallel composition can be "programmed" from the other operators. These propositions will be helpful in proving properties about the WT Net operators.

Proposition 2.2.18 Let $\langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets with disjoint sets of places and transitions and with a common alphabet, Act. Then there is a net context $C[\cdot,\cdot]$ built from prefixing and CCS choice such that $C[\langle N_1, Act\rangle, \langle N_2, Act\rangle]$ is isomorphic to $\langle N_1, Act\rangle \oplus \langle N_2, Act\rangle$.

Proof. It is easy to show that $\langle N_1, Act\rangle \oplus \langle N_2, Act\rangle$ is isomorphic to the net

$$(\tau.\langle N_1, Act\rangle) +_M (\tau.\langle N_2, Act\rangle).$$

The details are trivial and are left to the reader. ∎

Proposition 2.2.19 Let $\langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets with disjoint sets of places and transitions and with a common alphabet, Act, such that $Act - \{\surd\}$ is closed under complementation. Then there is a net context $C[\cdot,\cdot]$ built from action expansion and shrinking, CSP-style parallel composition, choice refinements, and hiding such that $C[\langle N_1, Act\rangle, \langle N_2, Act\rangle]$ is isomorphic to $\langle N_1, Act\rangle \mid \langle N_2, Act\rangle$.

Proof. Let $\{a_1, \overline{a_1}, \ldots, a_k, \overline{a_k}\} = Act - \{\surd\}$, and let $Act' = \{a'_1, \overline{a'_1}, \ldots, a'_k, \overline{a'_k}\}$ be distinct symbols not in Act. Let σ and σ' be the sequences of choice refinements

$$\sigma = \mathit{choice}_{(a_1,a_1,a'_1)} \cdot \mathit{choice}_{(\overline{a_1},\overline{a_1},\overline{a'_1})} \ldots \mathit{choice}_{(a_k,a_k,a'_k)} \cdot \mathit{choice}_{(\overline{a_k},\overline{a_k},\overline{a'_k})}$$

$$\sigma' = \mathit{choice}_{(a_1,a_1,\overline{a'_1})} \cdot \mathit{choice}_{(\overline{a_1},\overline{a_1},a'_1)} \ldots \mathit{choice}_{(a_k,a_k,\overline{a'_k})} \cdot \mathit{choice}_{(\overline{a_k},\overline{a_k},a'_k)}$$

Let

$$\langle N'_1, Act''\rangle \stackrel{\text{def}}{=} \sigma(\langle N_1, Act\rangle \ \mathit{grow}\ Act')$$

$$\langle N'_2, Act''\rangle \stackrel{\text{def}}{=} \sigma'(\langle N_2, Act\rangle \ \mathit{grow}\ Act')$$

Then it is straightforward to show that

$$\langle N_1, Act\rangle \mid \langle N_2, Act\rangle = ((\langle N'_1, Act''\rangle \|_{Act' \cup \{\surd\}} \langle N'_2, Act''\rangle) - Act')\ \mathit{shrink}\ \ Act,$$

where equality refers to net isomorphism, and $-Act'$ is shorthand for successively hiding each action in Act'. The details are straightforward and are left to the reader. ∎

Chapter 3

Semantics of Well-Terminating Nets

3.1 Testing Equivalence

This chapter develops some semantics for WT Nets that are compositional for all the WT Net operators presented in Chapter 2 and are respectively adequate for MAY-equivalence, MUST-equivalence, and Testing Equivalence [19]. Some fully abstract versions of these semantics are then presented.

Definition 3.1.1 A semantics, $[\![\cdot]\!]$, assigning to any process, P, a meaning, $[\![P]\!]$, is *compositional* for an operator on processes if semantic equality is a congruence for the operator, *i.e.*, the operator preserves semantic equality. We say that a semantics is *adequate* for an equivalence on processes if semantic equality implies process equivalence. Finally, we say that a semantics is *fully abstract* for a process equivalence *with respect to* a set of operators if the semantics is adequate for the equivalence and semantic equality is the coarsest congruence for those operators.

We presume that the reader is familiar with the experiment-based theory of MAY-equivalence, MUST-equivalence, and Testing equivalence on labeled transition systems developed in [19]. In order to keep this thesis relatively self-contained, we repeat the basic definitions here.

The idea behind experiment-based testing is that experimenters are given the ability to interact with processes in a way that affects both the process and the experimenter. In order to model success of an experiment, a special action ω is chosen to represent success. In this setting, both processes and experimenters are labeled transition systems over a common alphabet, except that in addition, the experimenter is allowed to independently perform the special actions $\mathbf{1}$ and ω. Processes do not have the ability to perform either $\mathbf{1}$ or ω. Both the experimenter and the process must "move together" on visible actions in the common alphabet, but can move independently on the τ action. In general, the behavior of an experimenter on a process is non-deterministic.

An *experiment* is a sequence of possible interactions between an experimenter and a process. Such a sequence is a *computation* iff it is an interaction which cannot be extended, *i.e.*, it is a maximal sequence of interactions. A computation is *successful* iff the experimenter passes through a state in which the ω action is enabled. We say that a process, p, *may satisfy* an experimenter, e, iff *some* interactive computation between e and p is successful. We say that a process, p, *must satisfy* an experimenter, e, iff *every* interactive computation between e and p is successful.

Definition 3.1.2 Let TS_1 and TS_2 be labeled transition systems respectively over alphabets Act_1, Act_2, where Act_1 and Act_2 may contain the τ action but do not contain the $\mathbf{1}$ or ω action. Let E be the set of labeled transition systems over $Act_1 \cup Act_2 \cup \{\mathbf{1}, \omega\}$. Then TS_1 and TS_2 are MAY-*equivalent* iff $Act_1 = Act_2$ and TS_1 and TS_2 *may satisfy* the same set of experimenters in E. Similarly, TS_1 and TS_2 are MUST-*equivalent* iff $Act_1 = Act_2$ and TS_1 and TS_2 *must satisfy* the same set of experimenters in E. TS_1 and TS_2 are Testing-*equivalent* iff they are both MAY-equivalent and MUST-equivalent.

The definitions of these equivalences carry over directly to WT Nets: two WT Nets will be said to be MAY-equivalent, MUST-equivalent, or Testing equivalent iff their labeled transition systems are respectively MAY-equivalent, MUST-equivalent, or Testing equivalent under the above definition. We assume without loss of generality that for any WT Net, $\langle N, Act\rangle$, the special actions $\mathbf{1}$ and ω are not in Act.

For technical simplicity, we will work with an alternate formulation of these equivalences, namely, partial trace equivalence [19, 30] and failures equivalence [7, 8, 9, 21]. In order to keep this thesis relatively self-contained, we repeat the definitions here:

Definition 3.1.3 Let TS be a labeled transition system, $\langle S, Act \cup \{\tau\}, \longrightarrow, s_{\text{init}}\rangle$, where Act is a set of visible actions. A state s is *divergent* iff s can perform an infinite sequence of τ-actions. A *failure set* of a state s is any set of *visible* actions, a, that are not enabled at s, even after further performing any finite sequence of τ-labeled actions; that is, $s \stackrel{a}{\not\Longrightarrow}$. Then:

$$traces(TS) \stackrel{\text{def}}{=} \{v \in Act^* : s_{\text{init}} \stackrel{v}{\Longrightarrow}\}$$

$$\begin{aligned}\mathcal{F}(TS) \stackrel{\text{def}}{=} \{\langle v, F\rangle : v \in Act^*,\ F \subseteq Act, &\text{ and there is some state } s \text{ such that} \\ s_{\text{init}} \stackrel{v}{\Longrightarrow} s &\text{ and } F \text{ is a failure set of } s\} \\ \cup\ \{\langle v, F\rangle : v \in \mathcal{D}(TS) &\text{ and } F \subseteq Act\}\end{aligned}$$

$$\mathcal{D}(TS) \stackrel{\text{def}}{=} \{v \cdot v' : v, v' \in Act^* \text{ and } s_{\text{init}} \stackrel{v}{\Longrightarrow} s \text{ for some divergent state } s\}$$

For any WT Net $\langle N, Act\rangle$, we define $traces(\langle N, Act\rangle) \stackrel{\text{def}}{=} traces(lts(\langle N, Act\rangle))$, $\mathcal{F}(\langle N, Act\rangle) \stackrel{\text{def}}{=} \mathcal{F}(lts(\langle N, Act\rangle))$, and $\mathcal{D}(\langle N, Act\rangle) \stackrel{\text{def}}{=} \mathcal{D}(lts(\langle N, Act\rangle))$.

Proposition 3.1.4 Let TS_1 and TS_2 be labeled transition systems respectively over *finite* alphabets Act_1, Act_2, where Act_1 and Act_2 may contain the τ action but do not contain the $\mathbf{1}$ or ω action. Then

- TS_1 and TS_2 are MAY-equivalent iff $Act_1 = Act_2$ and $traces(TS_1) = traces(TS_2)$.
- TS_1 and TS_2 are MUST-equivalent iff $Act_1 = Act_2$, $\mathcal{F}(TS_1) = \mathcal{F}(TS_2)$ and $\mathcal{D}(TS_1) = \mathcal{D}(TS_2)$.

- TS_1 and TS_2 are Testing-equivalent iff $Act_1 = Act_2$, $traces(TS_1) = traces(TS_2)$, $\mathcal{F}(TS_1) = \mathcal{F}(TS_2)$ and $\mathcal{D}(TS_1) = \mathcal{D}(TS_2)$.

The proof is a straightforward generalization of that in [19] and is left to the reader.

As shown in [19], MAY-equivalence, MUST-equivalence, and Testing-equivalence are compositional for all the standard CCS/CSP operators on labeled transition systems. Furthermore, they are compositional for (the natural definition of) choice refinements on labeled transition systems. Similar properties hold for WT Nets:

Proposition 3.1.5 MAY-equivalence, MUST-equivalence, and Testing-equivalence on WT Nets are compositional for all our CCS/CSP-style operators, choice refinements, and alphabet expansion and shrinking.

The proof is analogous to that of [19] and is omitted.

Since labeled transition systems are inherently sequential, these equivalences are also compositional for (the natural definition of) split refinements on labeled transition systems. A similar result holds for purely *sequential* WT Nets:

Proposition 3.1.6 MAY-equivalence, MUST-equivalence, and Testing-equivalence are compositional for split refinements on *sequential* WT Nets, in which no transitions can fire concurrently in any reachable marking.

Proof. Let $\langle N, Act\rangle$ be a *sequential* WT Net, and let a, a_+, a_- be distinct symbols in Act. For any sequence $v \in Act^*$, we define $split_{(a,a_+,a_-)}(v)$ to be the sequence $\alpha_1 \ldots \alpha_{|v|}$, where each $\alpha_i = a_+.a_-$ if $v[i] = a$, and $\alpha_i = v[i]$ otherwise.

Firing any newly-created a_+-labeled transition in $split_{(a,a_+,a_-)}(\langle N, Act\rangle)$ has the effect of "half-firing" the corresponding a-labeled transition of N, *i.e.*, removing all the tokens from the preset of the a-labeled transition but not placing any tokens in its post-set. Since $\langle N, Act\rangle$ is a sequential net, a_- is thus the one and only action enabled in $split_{(a,a_+,a_-)}(\langle N, Act\rangle)$ after performing any sequence of transitions that ends with an occurrence of a newly-created a_+-labeled transition.

It is then straightforward to show that

$$\begin{aligned}
&traces(split_{(a,a_+,a_-)}(\langle N, Act\rangle)) = \\
&\qquad \{split_{(a,a_+,a_-)}(v) \;:\; v \in traces(\langle N, Act\rangle)\} \cup \{split_{(a,a_+,a_-)}(v)\cdot a_+ \;:\; v\cdot a \in traces(\langle N, Act\rangle)\}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{F}(split_{(a,a_+,a_-)}(\langle N, Act\rangle)) = \\
&\qquad \{\langle split_{(a,a_+,a_-)}(v), F'\rangle \;:\; \text{there is some } F \text{ with } \langle v, F\rangle \in \mathcal{F}(\langle N, Act\rangle) \text{ such that} \\
&\qquad\qquad\qquad F' \subseteq F \cup \{a\}, \text{ and if } a_+ \in F' \text{ then } a \in F\} \\
&\qquad \cup \{\langle split_{(a,a_+,a_-)}(v)\cdot a_+,\; F'\rangle \;:\; \langle v\cdot a,\; \emptyset\rangle \in \mathcal{F}(\langle N, Act\rangle) \text{ and } F' \subseteq Act - \{a_-\}\} \\
&\qquad \cup \{\langle v, F\rangle \;:\; v \in \mathcal{D}(split_{(a,a_+,a_-)}(\langle N, Act\rangle)) \text{ and } F \subseteq Act\}
\end{aligned}$$

$$\mathcal{D}(split_{(a,a_+,a_-)}(\langle N, Act\rangle)) = \{split_{(a,a_+,a_-)}(v)\cdot v' \;:\; v \in \mathcal{D}(\langle N, Act\rangle) \text{ and } v' \in Act^*\}$$

The proposition is then a simple consequence of Proposition 3.1.4. ∎

However, as is well-known, neither MAY-equivalence, MUST-equivalence, nor Testing equivalence on arbitrary WT Nets is compositional for split refinements:

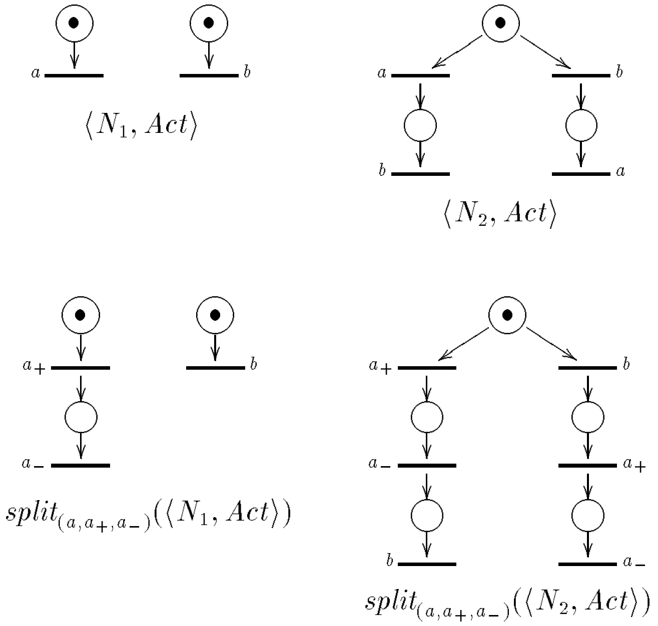


Figure 3-1: Standard Example

Proposition 3.1.7 ([10]) MAY-equivalence, MUST-equivalence, and Testing equivalence are not compositional for split refinements on arbitrary WT Nets.

Proof. It follows easily from Definition 3.1.3 and Proposition 3.1.4 that if any two divergence-free WT Nets are trace *inequivalent* then they are MAY-*inequivalent*, MUST-*inequivalent*, and Testing-*inequivalent*. To prove the proposition, we repeat the example given in [10], and illustrated in Figure 3-1. It is easy to show that $\langle N_1, Act\rangle$ and $\langle N_2, Act\rangle$ of Figure 3-1 are Testing-equivalent. However, $split_{(a,a_+,a_-)}(\langle N_1, Act\rangle)$ and $split_{(a,a_+,a_-)}(\langle N_2, Act\rangle)$ are trace-inequivalent, since a_+ba_- is a trace of $split_{(a,a_+,a_-)}(\langle N_1, Act\rangle)$ but not of $split_{(a,a_+,a_-)}(\langle N_2, Act\rangle)$. We note that $\langle N_1, Act\rangle$ is *not* a sequential net, since the a-labeled and b-labeled transitions can fire concurrently. ∎

It is well-known that trace-inequivalent *lts*'s cannot be strongly bisimilar (*cf.* [30]). Since the labeled transitions systems of $\langle N_1, Act\rangle$ and $\langle N_2, Act\rangle$ of Figure 3-1 are strongly bisimilar, the same example shows that *no interleaving semantics* (that lies in between trace equivalence and strong bisimulation) can be compositional for split refinements on arbitrary WT Nets. As is discussed in [39, 49], it is necessary keep track of "pomsets", which generalize linear sequences of actions to multi-sets of actions partially ordered to reflect causality and concurrency.

3.2 Some Compositional Semantics for WT Nets and Operators

We begin with the standard notions of pomsets.

Definition 3.2.1 A *pomset* is a labeled partial order. Formally, a pomset, p, consists of a set Events_p whose elements are called *events*, a set Labels_p whose elements are called *labels*, a function $\text{label}_p\colon \text{Events}_p \rightarrow \text{Labels}_p$, and a partial order relation $\leq_p$ on Events_p. We say that p is a pomset over an alphabet Act iff Act contains all the labels of p.

If p is a pomset with an empty carrier, we often simply write $\emptyset$ to denote p. If p is a pomset with a single event, labeled a, we often simply write a to denote p.

We say that event e *causes* event e' in a pomset p iff $e <_p e'$. The *downward-closure*, $down_p(e)$, of event e in a pomset p is $\{e' \in \text{Events}_p : e' <_p e\}$. The downward-closure, $down_p(E)$, of a subset E of Events_p is $E \cup \bigcup\{down_p(e) : e \in E\}$; E is *downward-closed* iff $down_p(E) = E$. We write $min(p)$ to denote the set of events in p that are minimal with respect to $<_p$, *i.e.*, events that do not have any causes in p. We write $max(p)$ to denote the set of events in p that are maximal with respect to $<_p$, *i.e.*, events that do not cause any event in p. We say that event e is a *maximal cause* of an event e' in pomset p iff $e <_p e'$ and there is no event $e'' \in \text{Events}_p$ such that $e <_p e'' <_p e'$.

The *size* of a pomset p, written $|p|$, is the size of the set Events_p. A *chain* in p is a sequence of events $x_1 x_2 \ldots x_k$ of p such that $x_1 <_p x_2 <_p \ldots <_p x_k$. The *depth* of an event x in p, written $depth_p(x)$, is the maximum length of any chain in p of the form $x_1 <_p x_2 <_p \ldots <_p x_k <_p x$, for any events $x_1, \ldots, x_k$. The *depth* of p, written $depth(p)$, is the maximum length of any chain in p.

A *cut* of p is any subset C of Events_p such that no two distinct events in C are causally related by $<_p$. The *width* of p is the maximum size of any cut of p.

A pomset p is a *prefix* of a pomset q iff p is a restriction of q to a downward-closed subset of Events_q.

A function f is an *isomorphism* between pomset p and pomset q iff it is a label-preserving order-isomorphism, namely,

- $f\colon \text{Events}_p \rightarrow \text{Events}_q$ is a bijection,
- $\text{label}_p = \text{label}_q \circ f$,
- $e \leq_p e'$ iff $f(e) \leq_q f(e')$ for all $e, e' \in \text{Events}_p$.

A pomset p' is a *linearization* of a pomset p iff it has the same events and labels as p and $<_{p'}$ is a total ordering that contains $<_p$. For any pomset q such that $<_q$ is a total ordering and any $1 \leq i \leq |\text{Events}_q|$, the i^{th} *largest event* of q is the (necessarily unique) event $e \in \text{Events}_q$ such that the longest chain $e_1 <_q \ldots <_q e_k <_q e$ in q is of length i.

We now define the pomsets arising from WT Nets:

Definition 3.2.2 The *places* of a transition t of a net N are the places directly connected to it, *i.e.*, the union of the preset and postset of t. Let t_1, t_2 be transitions of a net N. We say that t_1 and t_2 are *statically concurrent* in N iff the places of t_1 are disjoint from the places of t_2.

A *transition-sequence* is a sequence of transitions of a net N. For transition-sequence $r = t_1 \ldots t_n$ and $1 \leq i \leq n$, we write $r[i]$ to denote the i^{th} element, t_i, of r. The *transition-pomset* of $r = t_1 \ldots t_n$ has as events the integers from 1 to n, where the label of event i is t_i and the partial ordering is the transitive closure of the following "proximate cause" relation: event

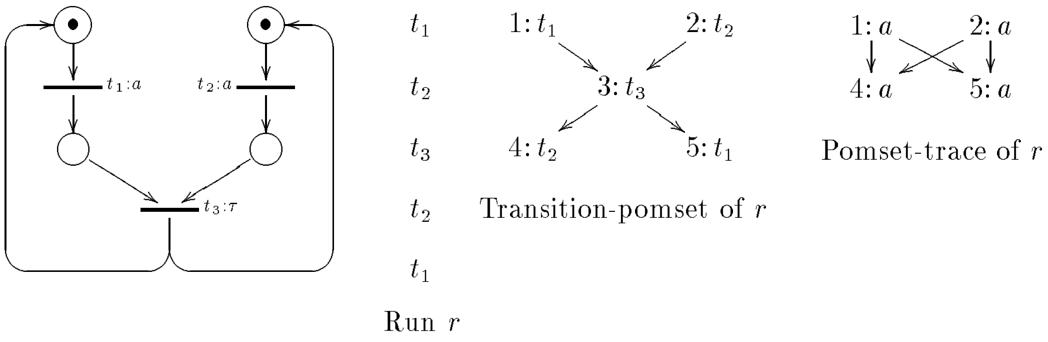


Figure 3-2: An Example of a Transition-pomset and Pomset-trace

i proximately causes event j iff $i < j$ and t_i and t_j are *not* statically concurrent in N, *cf.* Figure 3-2. The *pomset-runs* of a WT Net $\langle N, Act\rangle$ are the transition-pomsets of *runs* of N (*cf.* Definition 2.1.1).

If q is a transition-pomset of N, then $\mathit{visible}(q)$ is the restriction of q to its events with visible-transition labels (*cf.* Definition 2.1.1); furthermore, the label of each event i is the *label* of transition $l_q(i)$ (rather than transition $l_q(i)$ itself). The *pomset-traces* of a WT Net $\langle N, Act\rangle$ are the set of $\mathit{visible}(q)$ such that q is a finite pomset run of N, *cf.* Figure 3-2.

It is well known (*cf.* [42]) that there is a uniquely determined final marking associated with each finite pomset run of a net; this is the marking reached after sequentially firing the events of the run in any order that is consistent with its partial order.

Proposition 3.2.3 Let r be a run of a net N, let p' be a linearization of the transition-pomset of r, and let r' be the transition-sequence corresponding to p', *i.e.*, $r' = t_1 \dots t_{|r|}$, where each t_i is the label of the i^{th} largest event of p'. Then r' is a run of N reaching the same final marking as r.

Proof. Let v be the sequence $e_1 \dots e_{|r|}$, where each e_i is the i^{th} largest event of p'. Then it is easy to see that v is a permutation of the sequence $1 \dots |r|$, and $r'[i] = r[v[i]]$ for all $1 \le i \le |r|$.

We prove the proposition by induction on the number, n, of pairs (i, j) such that $i < j$ (as integers) but $v[i] > v[j]$ (as integers). The base case of $n = 0$ is trivial.

For the induction step, let $n \ge 1$. Then there is some k such that $v[k] > v[k+1]$. Let w be v with the k^{th} and $k+1^{\text{th}}$ elements "swapped"; that is, $w[k] = v[k+1]$, $w[k+1] = v[k]$, and w and v agree on all other indices. Clearly, the number of pairs (i, j) such that $i < j$ but $w[i] > w[j]$ is strictly less than n. Let p'' be the (totally-ordered) transition-pomset with the same labels and events as p' and such that for all events $e, e' \in \text{Events}_{p''}$, $e <_{p''} e'$ iff e occurs before e' in the sequence w. It is easy to show that p'' is a linearization of the transition-pomset of r. Thus, by induction, the transition-sequence r'' corresponding to p'' is a run of N reaching the same final marking as r. Furthermore, it is easy to see that $r''[i] = r[w[i]]$ for every $1 \le i \le |r|$.

Since p' is a linearization of the transition-pomset of r, clearly, event $v[k+1](= w[k])$ must not cause event $v[k](= w[k+1])$ in the transition-pomset of r, and so by Definition 6.2.1, transition $r[w[k]]$ and transition $r[w[k+1]]$ are statically concurrent in N. Furthermore, since r''

is a run of N, all places in the preset of transition $r[w[k+1]]$ must be marked after the run $r[w[1]] \ldots r[w[k]]$. The definition of static concurrency implies that no firing of transition $r[w[k]]$ can add any tokens to the preset of transition $r[w[k+1]]$; thus, transition $r[w[k+1]]$ must be enabled after the run $r[w[1]] \ldots r[w[k-1]]$ as well. Conversely, all places in the preset of transition $r[w[k]]$ must be marked after the run $r[w[1]] \ldots r[w[k-1]]$. The definition of static concurrency implies that no firing of transition $r[w[k+1]]$ can remove any tokens from the preset of $r[w[k]]$; thus, $r[w[k]]$ must be enabled after the run $r[w[1]] \ldots r[w[k-1]]r[w[k+1]]$ as well. It then follows easily that $r[w[1]] \ldots r[w[k-1]]r[w[k+1]]r[w[k]]$ is a run of N reaching the same final marking as the run $r[w[1]] \ldots r[w[k-1]]r[w[k]]r[w[k+1]]$, from which the lemma follows easily. ■

Our definition of "pomset-failures" is a natural generalization of (sequential) "failures" in that it associates "failure sets" to finite pomsets.

Definition 3.2.4 A *pomset-failure* is a pair $\langle p, F \rangle$, where p is a finite pomset, and F is a finite set of labels. We say that $\langle p, F \rangle$ is a pomset-failure over an alphabet Act iff Act contains all the labels of p and $F \subseteq Act$.

We define a "failure set" of a marking as any set of *visible* actions that are not enabled under that final marking, even after further firing any finite sequence of τ-labeled transitions. This is exactly the standard definition of "failure sets" of states of the labeled transition system of the net (*cf.* Definition 3.1.3). Using Proposition 3.2.3, we can unambiguously refer to the marking of a net reached after a pomset-run.

Definition 3.2.5 The *pomset-failures* of a WT Net $\langle N, Act \rangle$ are the pairs $\langle visible(q), F \rangle$ such that q is a finite pomset run of N and $F \subseteq Act$ is a failure set of the marking after q.

We also wish to define a notion of "pomset-divergences" that is a natural generalization of (sequential) "divergences."

Definition 3.2.6 A *pomset-divergence* is a pair $\langle p, D \rangle$, where p is a finite pomset and D is a *non-empty* set of downward-closed subsets of Events_p. We say that $\langle p, D \rangle$ is a pomset-divergence over an alphabet Act iff Act contains all the labels of p.

Given any pomset run with only a finite number of visible events, it is easy to see that any infinite chain of τ-labeled-events indicates a divergence of the net. We wish to define pomset-divergences of nets in such a way that we keep track of all the *concurrent divergences* within a pomset run while abstracting away from the τ-labeled events.

Definition 3.2.7 Let q be an infinite pomset run of a WT Net $\langle N, Act \rangle$ with a finite number of visible events. Let $\mathcal{D}$ be the family of sets of the form (events of) $visible(down_q(C))$ such that C is an infinite chain of τ-labeled events of q. Then $\langle visible(q), \mathcal{D} \rangle$ is a *pomset-divergence* of $\langle N, Act \rangle$, *cf.* Figure 3-3.

It turns out that the semantics defined by simply taking these pomset-failures and pomset-divergences makes too many distinctions between nets, and we need to "blur" certain kinds of information from our runs. This we accomplish through various closure operations. The first such closure involves taking "augmentations" of our pomset-failures and pomset-divergences. We first restate the standard definition for pomsets, where an augmentation is simply an increase in the partial ordering.

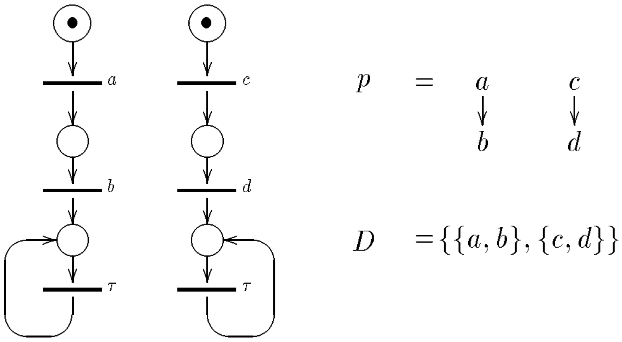


Figure 3-3: An Example of a Pomset Divergence $\langle p, D\rangle$

Definition 3.2.8 Pomset p' is an *augmentation* of pomset p iff p and p' have the same set of events with the same labels, and the partial ordering of p' contains the partial ordering of p. Let $augment(p)$ be the set of augmentations of p. The augmentations, $augment(\langle p, F\rangle)$, of a pomset-failure $\langle p, F\rangle$ is the set $\{\langle p', F\rangle \mid p' \in augment(p)\}$. For pomset-divergences, let $augment(\langle p, \mathcal{D}\rangle)$ be

$$\{\langle p', \mathcal{D}'\rangle : p' \in augment(p) \text{ and } \mathcal{D}' = \{down_{p'}(d) : d \in D\}\}$$

We write $p' \succeq p$ iff p' is an augmentation of p, and write $\langle p', D'\rangle \succeq \langle p, D\rangle$ iff $\langle p', D'\rangle$ is an augmentation of $\langle p, D\rangle$.

Our other closure operation arises from the fact that MUST-experiments fail to yield information about the behavior of a net after a divergence. To get around this difficulty, we define below the notion of an *extension* of a pomset-divergence; the idea is that the extension is another pomset-divergence which may contain more information concerning events and divergences which "happen after" one or more divergences in the original pomset-divergence. All the information about a process after a pomset-divergence is blurred by throwing in all possible pomset-failures and pomset-divergences which extend the original pomset-divergence.

Definition 3.2.9 Pomset-divergence $\langle p', \mathcal{D}'\rangle$ *extends* pomset-divergence $\langle p, \mathcal{D}\rangle$, written $\langle p, \mathcal{D}\rangle \sqsubseteq \langle p', \mathcal{D}'\rangle$, iff

p is a prefix of p'
for all $e \in p' - p$, there is some $d \in \mathcal{D}$ with $d \subseteq down_{p'}(e)$; and
for all $d' \in \mathcal{D}'$, there is some $d \in \mathcal{D}$ with $d \subseteq d'$.

For any alphabet Act which contains all the labels in pomset p, let $extend_{Act}(\langle p, \mathcal{D}\rangle)$ be the set of pomset-divergences over Act which extend $\langle p, D\rangle$. Finally, let

$$implied\text{-}failures_{Act}(\langle p, \mathcal{D}\rangle) \stackrel{\text{def}}{=} \{\langle p, F\rangle : F \subseteq Act\}.$$

We lift these operations on individual pomsets, failures, *etc.*, to sets of individuals by pointwise union. For example,

$$augment(X) \stackrel{\mathrm{def}}{=} \bigcup_{x \in X} augment(x).$$

We are now ready to define the pomset versions of the MAY-, MUST-, and Testing-semantics.

Definition 3.2.10 For any WT Net $\langle N, Act\rangle$,

$$\begin{aligned}
&[\![\langle N, Act\rangle]\!]^{\mathrm{MAY}} \stackrel{\mathrm{def}}{=} \langle augment(pomset\text{-}traces(\langle N, Act\rangle)), Act\rangle\\
&Div(\langle N, Act\rangle) \stackrel{\mathrm{def}}{=} augment(extend_{Act}(pomset\text{-}divergences(\langle N, Act\rangle))),\\
&Fail(\langle N, Act\rangle) \stackrel{\mathrm{def}}{=} augment(pomset\text{-}failures(\langle N, Act\rangle)) \cup implied\text{-}failures_{Act}(Div(\langle N, Act\rangle)),\\
&[\![\langle N, Act\rangle]\!]^{\mathrm{MUST}} \stackrel{\mathrm{def}}{=} \langle Fail(\langle N, Act\rangle), Div(\langle N, Act\rangle), Act\rangle,\\
&[\![\langle N, Act\rangle]\!]^{\mathrm{TEST}} \stackrel{\mathrm{def}}{=} \langle [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}, [\![\langle N, Act\rangle]\!]^{\mathrm{MUST}}\rangle.
\end{aligned}$$

Our definition of semantical equality implicitly equates label-preserving order-isomorphic pomsets.

We observe that:

Proposition 3.2.11 For any WT Net $\langle N, Act\rangle$, if $[\![\langle N, Act\rangle]\!]^{\mathrm{MAY}} = \langle PT, Act\rangle$ and $[\![\langle N, Act\rangle]\!]^{\mathrm{MUST}} = \langle PF, PD, Act\rangle$, then PT is a set of pomset-traces over Act, PF is a set of pomset-failures over Act, and PD is a set of pomset-divergences over Act.

The proof is trivial and is left to the reader.

Theorem 3.2.12 $[\![\cdot]\!]^{\mathrm{MAY}}$, $[\![\cdot]\!]^{\mathrm{MUST}}$, and $[\![\cdot]\!]^{\mathrm{TEST}}$ on WT Nets are respectively adequate for MAY-equivalence, MUST-equivalence, and Testing-equivalence.

Proof. There is an obvious correspondence between sequences of actions and linearly-ordered pomsets, which we implicitly use in the equalities below. Since the $[\![\cdot]\!]^{\mathrm{MAY}}$ and $[\![\cdot]\!]^{\mathrm{MUST}}$ semantics are augmentation-closed, it is straightforward to show that for any WT Net $\langle N, Act\rangle$,

$$traces(\langle N, Act\rangle) = \{v \in fst([\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}) : v \text{ is linearly ordered}\}$$

$$\mathcal{F}(\langle N, Act\rangle) = \{\langle v, F\rangle : v \text{ is linearly ordered and } \langle v, F\rangle \in fst([\![\langle N, Act\rangle]\!]^{\mathrm{MUST}})\}$$

$$\mathcal{D}(\langle N, Act\rangle) = \{v : v \text{ is linearly ordered and } \langle v, D\rangle \in snd([\![\langle N, Act\rangle]\!]^{\mathrm{MUST}}) \text{ for some } D\}$$

$$Act = snd([\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}) \quad = \quad third([\![\langle N, Act\rangle]\!]^{\mathrm{MUST}})$$

from which the theorem follows directly. ∎

The following closure properties of the semantics will be useful in proving compositionality. We extend the definition of prefixes to pomset-divergences:

Definition 3.2.13 Let $\langle p, D_p\rangle, \langle q, D_q\rangle$ be pomset-divergences. Then $\langle p, D_p\rangle$ is a *prefix* of $\langle q, D_q\rangle$ iff p is a prefix of q and $D_p \subseteq D_q$.

Proposition 3.2.14 Let $\langle N, Act\rangle$ be a WT Net. Then *pomset-traces*$(\langle N, Act\rangle)$ is a prefix-closed set of pomset-traces and and *pomset-divergences*$(\langle N, Act\rangle)$ is a prefix-closed set of pomset-divergences. Furthermore, for any pomset-failure $\langle p, F\rangle$ and prefix q of p, if $\langle p, F\rangle$ is a pomset-failure of N, then so is $\langle q, \emptyset\rangle$.

Proof. The proposition is easily proved from the definitions of pomset-traces, pomset-failures, and pomset-divergences of nets. ∎

Proposition 3.2.15 Let $\langle p, D_p\rangle$, $\langle q, D_q\rangle$, and $\langle r, D_r\rangle$ be pomset-divergences with $\langle p, D_p\rangle \sqsubseteq \langle q, D_q\rangle \preceq \langle r, D_r\rangle$. Then there is some $\langle p', D_{p'}\rangle \sqsubseteq \langle r, D_r\rangle$ such that $\langle p', D_{p'}\rangle$ is an augmentation of a prefix of $\langle p, D_p\rangle$.

Proof. Let p' be the restriction of r to the set $\{e \in \mathrm{Events}_p \colon down_r(e) \subseteq p\}$, and let $D_{p'} = \{down_{p'}(d) \colon d \in D_p \text{ and } d \subseteq p'\}$. The remainder of the proof is straightforward and is left to the reader. ∎

Proposition 3.2.16 Let $\langle p, D_p\rangle$, $\langle q, D_q\rangle$, and $\langle r, D_r\rangle$ be pomset-divergences with $\langle p, D_p\rangle \preceq \langle q, D_q\rangle \sqsubseteq \langle r, D_r\rangle$. Then there is some $\langle q', D_{q'}\rangle \preceq \langle r, D_r\rangle$ such that $\langle q', D_{q'}\rangle$ extends $\langle p, D_p\rangle$.

Proof. Let q' have the same events and same labels as r, and let $D_{q'} = D_r$. Furthermore, define $\leq_{q'}$ as $x <_{q'} y$ iff $x \in \mathrm{Events}_p$ and $x <_r y$. The remainder of the proof is straightforward and is left to the reader. ∎

Proposition 3.2.17 Let p, q, q' be pomsets such that $p \preceq q$ and q' is a prefix of q. Then there is some $p' \preceq q'$ such that p' is a prefix of p.

Let $\langle p, D_p\rangle, \langle q, D_q\rangle, \langle q', D_{q'}\rangle$ be pomset-divergences such that $\langle p, D_p\rangle \preceq \langle q, D_q\rangle$ and $\langle q', D_{q'}\rangle$ is a prefix of $\langle q, D_q\rangle$. Then there is some $\langle p', D_{p'}\rangle \preceq \langle q', D_{q'}\rangle$ such that $\langle p', D_{p'}\rangle$ is a prefix of $\langle p, D_p\rangle$.

Let $\langle p, D_p\rangle, \langle q, D_q\rangle, \langle q', D_{q'}\rangle$ be pomset-divergences such that $\langle p, D_p\rangle \sqsubseteq \langle q, D_q\rangle$ and $\langle q', D_{q'}\rangle$ is a prefix of $\langle q, D_q\rangle$. Then there is some $\langle p', D_{p'}\rangle \sqsubseteq \langle q', D_{q'}\rangle$ such that $\langle p', D_{p'}\rangle$ is a prefix of $\langle p, D_p\rangle$.

Proof. Let p' be the restriction of p to $\mathrm{Events}_{q'}$ and, for the second and third parts, let $D_{p'} = \{d \in D_p : d \subseteq \mathrm{Events}_{q'}\}$. The remainder of the proof is straightforward and is left to the reader. ∎

Proposition 3.2.18 Let $\langle N, Act\rangle$ be a WT Net. Then *Fail*$(\langle N, Act\rangle)$ is an augmentation-closed set of pomset-failures and *Div*$(\langle N, Act\rangle)$ is an augmentation-closed and extension-closed set of pomset-divergences.

Proof. It is easy to see from Definition 3.2.10 and the definition of *implied-failures* that that both sets are augmentation-closed. The extension-closure of $Div(\langle N, Act\rangle)$ is a simple consequence of Proposition 3.2.16. ∎

The following operations on pomsets and pomset-divergences correspond to our operators on WT Nets and will be useful in proving compositionality of our semantics.

Definition 3.2.19 Let p be a pomset and a a label. We define p *with* a to be the set of pomsets p' such that p is a prefix of p', there is exactly one event in $p' - p$, and this event is a-labeled.

Definition 3.2.20 Let p be a pomset and let X be a set of maximal events in p. Then $p - X$ is p restricted to $\mathrm{Events}_p - X$.

Definition 3.2.21 Let p be a pomset, $\langle p, D\rangle$ a pomset-divergence, and a a label. We define $a.p$ to be the pomset with $\mathrm{Events}_{a.p} = \mathrm{Events}_p \cup \{e_a\}$ for some $e_a \notin \mathrm{Events}_p$, $l_{a.p}(e_a) = a$ and $l_{a.p}$ agrees with l_p on Events_p, and $\leq_{a.p} = \leq_p \cup (\{e_a\} \times \mathrm{Events}_p)$. Furthermore, $a.\langle p, D\rangle \stackrel{\mathrm{def}}{=} \langle a.p, \{d \cup \{e_a\} : d \in D\}\rangle$.

Definition 3.2.22 Let p be a pomset, $\langle p, D\rangle$ a pomset-divergence, and a a label. We define $p\!-a$ to be p restricted to its events that are not a-labeled, and
$\langle p, D\rangle - a \stackrel{\mathrm{def}}{=} \langle p\!-a, \{d \cap \mathrm{Events}_{p-a} : d \in D\}\rangle$.

Definition 3.2.23 Let p be a pomset, $\langle p, D\rangle$ a pomset-divergence, and f a function from labels to labels whose domain contains all the labels in p. Then $p[f]$ has the same events as p with the same ordering, but $l_{p[f]} = f \circ l_p$. Furthermore, $\langle p, D\rangle[f] \stackrel{\mathrm{def}}{=} \langle p[f], D\rangle$.

Definition 3.2.24 Let p and q be pomsets with disjoint sets of events. We define $p;q$ to be the pomset such that $\mathrm{Events}_{p;q} = \mathrm{Events}_p \cup \mathrm{Events}_q$, $l_{p;q}$ agrees with l_p on Events_p and agrees with l_q on Events_q, and $\leq_{p;q} = \leq_p \cup \leq_q \cup (\mathrm{Events}_p \times \mathrm{Events}_q)$.

Definition 3.2.25 Let p be a pomset, $\langle p, D\rangle$ a pomset-divergence, and a, a_L, a_R labels. Then $choice_{(a,a_L,a_R)}(p)$ is the set of pomsets q with the same events and same ordering as p and such that l_q agrees with l_p on all non-a-labeled events of p, and $l_q(x) = a_L$ or $l_q(x) = a_R$ for all a-labeled events x of p. Furthermore,

$$choice_{(a,a_L,a_R)}(\langle p, D\rangle) \stackrel{\mathrm{def}}{=} \{\langle q, D\rangle : q \in choice_{(a,a_L,a_R)}(p)\}.$$

Definition 3.2.26 Let p be a pomset, a, a_+, a_- be labels, and $H \subseteq \{x \in max(p) \colon l_p(x) = a\}$. Then $split_{(a,a_+,a_-,H)}(p)$ is defined to be the pomset q with all a-labeled events in H "half-split" and all other a-labeled events "fully split," *i.e.*,

- $\mathrm{Events}_q = \{(y,0) \in \mathrm{Events}_p : l_p(y) \neq a\} \cup \{(y,1) \colon y \in H\}$
  $\cup \{(y,1),(y,2) \colon y \in \mathrm{Events}_p - H \text{ and } l_p(y) = a\}$.
- For all $(y,i) \in \mathrm{Events}_q$, $l_q((y,1)) = a_+$, $l_q((y,2)) = a_-$, and $l_q((y,0)) = l_p(y)$.
- For all $(x,i),(y,j) \in \mathrm{Events}_q$, $(x,i) <_q (y,j)$ iff either $x <_q y$ or ($x =_p y$ and $i < j$).

We then define:

$$split_{(a,a_+,a_-)}(p) \stackrel{\text{def}}{=} \{split_{(a,a_+,a_-,H)}(p) : H \subseteq \{x \in max(p) : l_p(x) = a\}\}$$

Our definition of $\|_A$ on pomsets generalizes that of [8, 21] on sequences of actions. In particular, we are careful to prohibit synchronizations between pairs of actions that introduce too many ordering constraints and hence violate anti-symmetry of the partial orders.

Definition 3.2.27 Let A be a finite alphabet, let p and q be pomsets with disjoint sets of events, and let D_p and D_q be downward-closed subsets of Events_p and Events_q, respectively, such that $D_p \cup D_q \neq \emptyset$. For any bijection f from $\{e \in \text{Events}_p : l_p(e) \in A\}$ to $\{e' \in \text{Events}_q : l_q(e') \in A\}$ such that

- f is *label-preserving*, *i.e.*, $l_p(e) = l_q(f(e))$ for all $e \in \text{Events}_p$ with $l_p(e) \in A$, and
- f is *order-non-contradicting*, *i.e.*,
  - The transitive-closure of $\leq_p \cup \{(e, e') \in \text{Events}_p \times \text{Events}_p : f(e) <_q f(e')\}$ is a partial ordering; in particular, it is anti-symmetric.
  - The transitive-closure of $\leq_q \cup \{(f(e), f(e')) \in \text{Events}_q \times \text{Events}_q : e <_p e'\}$ is a partial ordering; in particular, it is anti-symmetric.

we define $r = p\|_A^f q$ as:

- $\text{Events}_r = \{(e, *) : e \in \text{Events}_p \text{ and } l_p(e) \notin A\} \cup \{(*, e') : e' \in \text{Events}_q \text{ and } l_q(e') \notin A\} \cup \{(e, f(e)) : e \in \text{Events}_p \text{ and } l_p(e) \in A\}$.
- $l_r(e, *) = l_p(e)$, $l_r(*, e') = l_q(e')$, and $l_r(e, f(e)) = l_p(e)$.
- $(x, y) \leq_r (x', y')$ iff either $x \leq_p x'$ or $y \leq_q y'$.

We define $\langle p, D_p\rangle \|_A^f \langle q, D_q\rangle = \langle r, D_r\rangle$, where $r = p\|_A^f q$ and

$$\begin{aligned} D_r = &\{down_r(E) : E = \{z \in \text{Events}_r : fst(z) \in d\} \text{ for some } d \in D_p\} \\ &\cup \{down_r(E') : E' = \{z \in \text{Events}_r : snd(z) \in d'\} \text{ for some } d' \in D_q\} \end{aligned}$$

We define

$$\begin{aligned} p\|_A q \stackrel{\text{def}}{=} \{p\|_A^f q : \ &f \text{ is a label-preserving, ordering-non-contradicting bijection from} \\ &\{e \in \text{Events}_p : l_p(e) \in A\} \text{ to } \{e' \in \text{Events}_q : l_q(e') \in A\}\} \end{aligned}$$

and

$$\begin{aligned} \langle p, D_p\rangle \|_A \langle q, D_q\rangle \stackrel{\text{def}}{=} \{\langle p, D_p\rangle \|_A^f \langle q, D_q\rangle : \ &f \text{ is a label-preserving, ordering-non-contradicting} \\ &\text{bijection from } \{e \in \text{Events}_p : l_p(e) \in A\} \\ &\text{to } \{e' \in \text{Events}_q : l_q(e') \in A\}\} \end{aligned}$$

It is easy to show that pomsets and pomset-divergences are closed under all the above operations; the details are left to the reader.

We now define corresponding operations on *sets* of pomset-traces, pomset-failures, and pomset-divergences. We will use these definitions heavily in proving the compositionality of our semantics with respect to the WT Net operators.

Definition 3.2.28 Let Act be a finite alphabet containing the distinguished symbol $\surd$ and let PT be a set of pomset-traces over Act. Let $a, a_L, a_R, a_+, a_- \in Act - \{\surd\}$, let A be a subset of Act containing $\surd$, let f be a function from Act to Act such that for all $\alpha \in Act$, $f(\alpha) = \surd$ iff $\alpha = \surd$, and let Act' be a finite set of labels containing $\surd$. Then:

$$\langle PT, Act\rangle \ \mathit{grow}\ Act' \stackrel{\text{def}}{=} \langle PT, Act \cup Act'\rangle$$

$$\langle PT, Act\rangle \ \mathit{shrink}\ Act' \stackrel{\text{def}}{=} \langle \{p \in PT : \text{ all events in } p \text{ have labels in } Act'\}, Act'\rangle$$

$$a.\langle PT, Act\rangle \stackrel{\text{def}}{=} \langle \{\emptyset\} \cup \{a.p : p \in PT\}, Act\rangle$$

$$\langle PT, Act\rangle \backslash a \stackrel{\text{def}}{=} \langle \{p \in PT : p \text{ has no } a\text{-labeled event}\}, Act\rangle$$

$$\langle PT, Act\rangle[f] \stackrel{\text{def}}{=} \langle \{p[f] : p \in PT\}, Act\rangle$$

$$\langle PT, Act\rangle - a \stackrel{\text{def}}{=} \langle \{p - a : p \in PT\}, Act\rangle$$

$$\begin{aligned}\langle PT_1, Act\rangle ; \langle PT_2, Act\rangle \stackrel{\text{def}}{=} \ & \langle \{p \in PT_1 : p \text{ does not contain a } \surd\text{-labeled event}\} \\ & \cup \{(p_1; p_2) : (p_1; \surd) \in PT_1 \text{ and } p_2 \in PT_2\}, Act\rangle\end{aligned}$$

$$\langle PT_1, Act\rangle \oplus \langle PT_2, Act\rangle \stackrel{\text{def}}{=} \langle PT_1 \cup PT_2, Act\rangle$$

$$\langle PT_1, Act\rangle +_M \langle PT_2, Act\rangle \stackrel{\text{def}}{=} \langle PT_1 \cup PT_2, Act\rangle$$

$$\langle PT_1, Act\rangle ||_A \langle PT_2, Act\rangle \stackrel{\text{def}}{=} \langle \mathit{augment}(\bigcup \{p_1 ||_A p_2 : p_1 \in PT_1, p_2 \in PT_2\}), Act\rangle$$

$$\langle PT_1, Act\rangle \ || \ \langle PT_2, Act\rangle \stackrel{\text{def}}{=} \langle \mathit{augment}(\bigcup \{p_1 ||_{\{\surd\}} p_2 : p_1 \in PT_1, p_2 \in PT_2\}), Act\rangle$$

$$\mathit{split}_{(a,a_+,a_-)}(\langle PT, Act\rangle) \stackrel{\text{def}}{=} \langle \mathit{augment}(\bigcup \{\mathit{split}_{(a,a_+,a_-)}(p) : p \in PT\}), Act\rangle$$

$$\mathit{choice}_{(a,a_L,a_R)}(\langle PT, Act\rangle) \stackrel{\text{def}}{=} \langle \bigcup \{\mathit{choice}_{(a,a_L,a_R)}(p) : p \in PT\}, Act\rangle$$

$$\begin{aligned}
&\langle PT_1, Act\rangle \mid \langle PT_2, Act\rangle \stackrel{\text{def}}{=} \\
&\quad ((\sigma(\langle PT_1, Act\rangle \ grow \ Act') \|_{Act' \cup \{\surd\}} \sigma'(\langle PT_2, Act\rangle \ grow \ Act')) - Act') \ shrink \ Act
\end{aligned}$$

where $\{a_1, \overline{a_1}, \ldots, a_k, \overline{a_k}\} = Act - \{\surd\}$,
$Act' = \{a_1', \overline{a_1'}, \ldots, a_k', \overline{a_k'}\}$ are distinct symbols not in Act,
σ is the sequence $choice_{(a_1, a_1, a_1')} \cdot choice_{(\overline{a_1}, \overline{a_1}, \overline{a_1'})} \ldots choice_{(a_k, a_k, a_k')} \cdot choice_{(\overline{a_k}, \overline{a_k}, \overline{a_k'})}$,
and σ' is the sequence $choice_{(a_1, a_1, \overline{a_1'})} \cdot choice_{(\overline{a_1}, \overline{a_1}, a_1')} \ldots choice_{(a_k, a_k, \overline{a_k'})} \cdot choice_{(\overline{a_k}, \overline{a_k}, a_k')}$

Definition 3.2.29 Let Act be a finite alphabet containing the distinguished symbol $\surd$, let PF, PF_1, PF_2 be sets of pomset-failures over Act, and let PD, PD_1, PD_2 be sets of pomset-divergences over Act. Let $a, a_L, a_R, a_+, a_- \in Act - \{\surd\}$, let A be a subset of Act containing $\surd$, let f be a function from Act to Act such that for all $\alpha \in Act$, $f(\alpha) = \surd$ iff $\alpha = \surd$, and let Act' be a finite set of labels containing $\surd$.

Then:

$$\begin{aligned}
\langle PF, PD, Act\rangle \ grow \ Act' &\stackrel{\text{def}}{=} \langle PF', PD', Act \cup Act'\rangle \\
&\text{where} \\
PF' &= \{\langle p, F \cup X\rangle : X \subseteq Act' - Act \text{ and } \langle p, F\rangle \in PF\} \\
&\quad \cup \ implied\text{-}failures_{Act \cup Act'}(PD') \\
PD' &= augment(extend_{Act \cup Act'}(PD))
\end{aligned}$$

$$\begin{aligned}
\langle PF, PD, Act\rangle \ shrink \ Act' &\stackrel{\text{def}}{=} \langle PF', PD', Act'\rangle \\
&\text{where} \\
PF' &= \{\langle p, F\rangle \in PF : \text{ all events in } p \text{ have labels in } Act' \text{ and } F \subseteq Act'\} \\
PD' &= \{\langle p, D\rangle \in PD : \text{ all events in } p \text{ have labels in } Act'\}
\end{aligned}$$

$$\begin{aligned}
a.\langle PF, PD, Act\rangle &\stackrel{\text{def}}{=} \langle PF', PD', Act\rangle \\
&\text{where} \\
PF' &= \{\langle \emptyset, F\rangle : F \subseteq Act - \{a\}\} \cup \{\langle a.p, F\rangle : \langle p, F\rangle \in PF\} \\
PD' &= \{a.\langle p, D\rangle : \langle p, D\rangle \in PD\}
\end{aligned}$$

$$\begin{aligned}
\langle PF, PD, Act\rangle \backslash a &\stackrel{\text{def}}{=} \langle PF', PD', Act\rangle \\
&\text{where} \\
PF' &= \{\langle p, F\rangle : p \text{ has no } a\text{-labeled event and } \langle p, F - \{a\}\rangle \in PF\} \\
&\quad \cup \ implied\text{-}failures_{Act}(PD') \\
PD' &= augment(extend_{Act}(\{\langle p, D\rangle \in PD : p \text{ has no } a\text{-labeled event}\}))
\end{aligned}$$

$$\begin{array}{rcl} \langle PF, PD, Act\rangle[f] & \stackrel{\text{def}}{=} & \langle PF', PD', Act\rangle \\ & \text{where} & \\ PF' & = & \{\langle p[f], F\rangle : F \subseteq Act \text{ and } \langle p, \{b \in Act : f(b) = a \text{ for some } a \in F\}\rangle \in PF\} \\ & & \quad \cup\, \textit{implied-failures}_{Act}(PD') \\ PD' & = & \textit{augment}(\textit{extend}_{Act}(\{\langle p, D\rangle[f] : \langle p, D\rangle \in PD\})) \end{array}$$

$$\begin{array}{rcl} \multicolumn{3}{l}{\langle PF, PD, Act\rangle - a \stackrel{\text{def}}{=} \langle PF', PD', Act\rangle} \\ & \text{where} & \\ PF' & = & \{\langle p - a, F\rangle : \langle p, F \cup \{a\}\rangle \in PF\} \cup \textit{implied-failures}_{Act}(PD') \\ PD' & = & \textit{augment}(\textit{extend}_{Act}(\{\langle p, D \cup D_p\rangle - a : \langle p, D\rangle \in PD \cup PF, \\ & & \qquad \langle p, D \cup D_p\rangle \text{ is a pomset-divergence,} \\ & & \qquad \text{and for all } n \geq 0, \text{ there is some } p_n \text{ with } \langle p_n, D\rangle \in PD \cup PF \\ & & \qquad \text{such that} \\ & & \qquad \langle p, D_p\rangle \sqsubseteq \langle p_n, \{\text{Events}_{p_n}\}\rangle, \text{ all events in } p_n - p \text{ are } a\text{-labeled,} \\ & & \qquad \text{and for every } d \in D_p, \\ & & \qquad \text{there is some } n\text{-length chain of } a\text{-labeled events in } p_n - p \\ & & \qquad \text{whose downward closure restricted to } p \text{ is } d\})) \end{array}$$

$$\begin{array}{rcl} \multicolumn{3}{l}{\textit{choice}_{(a,a_L,a_R)}(\langle PF, PD, Act\rangle) \stackrel{\text{def}}{=} \langle PF', PD', Act\rangle} \\ & \text{where} & \\ PF' & = & \{\langle q, F_q\rangle : \text{ there is some } \langle p, F_p\rangle \in PF \text{ such that } q \in \textit{choice}_{(a,a_L,a_R)}(p),\ F_q \subseteq F_p \cup \{a\}, \\ & & \qquad \text{and if } a_L \in F_q \text{ or } a_R \in F_q \text{ then } a \in F_p\} \\ & & \quad \cup\, \textit{implied-failures}_{Act}(PD') \\ \\ PD' & = & \textit{augment}(\textit{extend}_{Act}(\bigcup\{\textit{choice}_{(a,a_L,a_R)}(\langle p, D\rangle) : \langle p, D\rangle \in PD\})) \end{array}$$

$$\langle PF_1, PD_1, Act\rangle \oplus \langle PF_2, PD_2, Act\rangle \stackrel{\text{def}}{=} \langle PF_1 \cup PF_2, PD_1 \cup PD_2, Act\rangle$$

$$\begin{array}{rcl} \multicolumn{3}{l}{\langle PF_1, PD_1, Act\rangle; \langle PF_2, PD_2, Act\rangle \stackrel{\text{def}}{=} \langle PF', PD', Act\rangle} \\ & \text{where} & \\ PF' & = & \{\langle p, F\rangle : \langle p, F \cup \{\surd\}\rangle \in PF_1 \text{ and } p \text{ does not contain a } \surd\text{-labeled event}\} \\ & & \cup\, \{\langle p_1; p_2, F\rangle : \langle p_1; \surd, \emptyset\rangle \in PF_1 \text{ and } \langle p_2, F\rangle \in PF_2\} \cup \textit{implied-failures}_{Act}(PD') \\ PD' & = & PD_1 \cup \{\langle p_1; p_2, \{d \cup \text{Events}_{p_1} : d \in D\rangle : \langle p_1; \surd, \emptyset\rangle \in PF_1 \text{ and } \langle p_2, D\rangle \in PD_2\} \end{array}$$

$$\begin{array}{rcl} \multicolumn{3}{l}{\langle PF_1, PD_1, Act\rangle \|_A \langle PF_2, PD_2, Act\rangle \stackrel{\text{def}}{=} \langle PF', PD', Act\rangle} \\ & \text{where} & \\ PF' & = & \textit{augment}(\{\langle p, F\rangle : \text{ there are some } \langle p_1, F_1\rangle \in PF_1, \langle p_2, F_2\rangle \in PF_2 \text{ such that } p \in p_1 \|_A p_2, \\ & & \qquad F - A \subseteq F_1 \cap F_2 \text{ and } F \cap A \subseteq F_1 \cup F_2\}) \\ & & \quad \cup\, \textit{implied-failures}_{Act}(PD') \\ \\ PD' & = & \textit{augment}(\textit{extend}_{Act}(\bigcup\{\langle p_1, D_1\rangle \|_A \langle p_2, D_2\rangle : \langle p_1, D_1\rangle \in PD_1 \cup PF_1, \langle p_2, D_2\rangle \in PD_2 \cup PF_2, \\ & & \qquad D_1 \text{ and } D_2 \text{ are (possibly empty) downward-closed subsets of} \\ & & \qquad \text{Events}_{p_1} \text{ and Events}_{p_2}, \text{ respectively, and } D_1 \cup D_2 \neq \emptyset\})) \end{array}$$

$$\langle PF_1, PD_1, Act\rangle \mid\mid \langle PF_2, PD_2, Act\rangle \stackrel{\text{def}}{=} \langle PF_1, PD_1, Act\rangle \|_{\{\surd\}} \langle PF_2, PD_2, Act\rangle$$

$\langle PF_1, PD_1, Act\rangle \mid \langle PF_2, PD_2, Act\rangle \stackrel{\text{def}}{=}$
$((\sigma(\langle PF_1, PD_1, Act\rangle \textit{ grow } Act')\|_{Act' \cup \{\surd\}} \sigma'(\langle PF_2, PD_2, Act\rangle \textit{ grow } Act')) - Act') \textit{ shrink } Act$
where $\{a_1, \overline{a_1}, \ldots, a_k, \overline{a_k}\} = Act - \{\surd\}$,
$Act' = \{a'_1, \overline{a'_1}, \ldots, a'_k, \overline{a'_k}\}$ are distinct symbols not in Act,
σ is the sequence $\mathit{choice}_{(a_1, a_1, a'_1)} \cdot \mathit{choice}_{(\overline{a_1}, \overline{a_1}, \overline{a'_1})} \ldots \mathit{choice}_{(a_k, a_k, a'_k)} \cdot \mathit{choice}_{(\overline{a_k}, \overline{a_k}, \overline{a'_k})}$,
and σ' is the sequence $\mathit{choice}_{(a_1, a_1, \overline{a'_1})} \cdot \mathit{choice}_{(\overline{a_1}, \overline{a_1}, a'_1)} \ldots \mathit{choice}_{(a_k, a_k, \overline{a'_k})} \cdot \mathit{choice}_{(\overline{a_k}, \overline{a_k}, a'_k)}$

Theorem 3.2.30 $[\![\cdot]\!]^{\mathrm{MAY}}$ is compositional for split refinements, choice refinements, alphabet expansion and shrinking, and all of our CCS/CSP operators.

Proof. Let Act be a finite alphabet containing $\surd$, let $a, a_L, a_R, a_+, a_- \in Act - \{\surd\}$, let $A \subseteq Act$, let f be a function from Act to Act such that for all $\alpha \in Act$, $f(\alpha) = \surd$ iff $\alpha = \surd$, and let Act' be a finite set of labels containing $\surd$. Furthermore, let $\langle N, Act\rangle, \langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets.

It is straightforward but tedious to show that the following identities hold, where the operations on the right-hand side of the equations are those defined in Definition 3.2.28. As an illustration, we will prove the equality for prefixing; the details of the other cases are left to the reader.

$$[\![\langle N, Act\rangle \textit{ grow } Act']\!]^{\mathrm{MAY}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}} \textit{ grow } Act'$$

$$[\![\langle N, Act\rangle \textit{ shrink } Act']\!]^{\mathrm{MAY}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}} \textit{ shrink } Act'$$

$$[\![a.\langle N, Act\rangle]\!]^{\mathrm{MAY}} = a.[\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}$$

$$[\![\tau.\langle N, Act\rangle]\!]^{\mathrm{MAY}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}$$

$$[\![\langle N, Act\rangle \backslash a]\!]^{\mathrm{MAY}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}} \backslash a$$

$$[\![\langle N, Act\rangle [f]]\!]^{\mathrm{MAY}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}} [f]$$

$$[\![\langle N, Act\rangle - a]\!]^{\mathrm{MAY}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}} - a$$

$$[\![\langle N_1, Act\rangle ; \langle N_2, Act\rangle]\!]^{\mathrm{MAY}} = [\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}} ; [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}$$

$$[\![\langle N_1, Act\rangle \oplus \langle N_2, Act\rangle]\!]^{\mathrm{MAY}} = [\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}} \oplus [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}$$

$$[\![\langle N_1, Act\rangle +_M \langle N_2, Act\rangle]\!]^{\mathrm{MAY}} = [\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}} +_M [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}$$

$$[\![\langle N_1, Act\rangle \mid\mid \langle N_2, Act\rangle]\!]^{\mathrm{MAY}} = [\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}} \mid\mid [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}$$

$$[\![\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle]\!]^{\mathrm{MAY}} = [\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}} \|_{A\cup\{\surd\}} [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}$$

$$[\![\langle N_1, Act\rangle \mid \langle N_2, Act\rangle]\!]^{\mathrm{MAY}} = [\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}} \mid [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}$$

$$[\![\mathit{choice}_{(a,a_L,a_R)}(\langle N, Act\rangle)]\!]^{\mathrm{MAY}} = \mathit{choice}_{(a,a_L,a_R)}([\![\langle N, Act\rangle]\!]^{\mathrm{MAY}})$$

$$[\![\mathit{split}_{(a,a_+,a_-)}(\langle N, Act\rangle)]\!]^{\mathrm{MAY}} = \mathit{split}_{(a,a_+,a_-)}([\![\langle N, Act\rangle]\!]^{\mathrm{MAY}})$$

To prove the equality for prefixing, we first make without proof the easy observation that

$$\mathit{pomset\text{-}traces}(a.\langle N, Act\rangle) = \mathit{fst}(a.\langle \mathit{pomset\text{-}traces}(\langle N, Act\rangle), Act\rangle).$$

For one direction of the desired equality above, let $q \in [\![a.\langle N, Act\rangle]\!]^{\mathrm{MAY}}$; then, from the above fact, the definition of $[\![\cdot]\!]^{\mathrm{MAY}}$, and Definition 3.2.28, it is easy to see that either $q = \emptyset$ or $q \succeq a.p$ for some pomset-trace p of $\langle N, Act\rangle$. It follows from general properties of pomsets that either $q = \emptyset$ or $q = a.q'$ for some $q' \succeq p$, from which it follows immediately that $q \in a.[\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}$. For the other direction, let $r \in a.[\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}$; then either $r = \emptyset$ or $r = a.p$ for some p that is an augmentation of some pomset-trace p' of $\langle N, Act\rangle$. If r is non-empty, it follows from general properties of pomsets that $r \succeq a.p'$. Using the definition of $[\![\cdot]\!]^{\mathrm{MAY}}$ and the highlighted fact above, it is then easy to see that $r \in [\![a.\langle N, Act\rangle]\!]^{\mathrm{MAY}}$, proving this case.

Proposition 2.2.19 and the above equalities for alphabet expansion and shrinking, CSP-style parallel composition, choice refinements, and hiding together immediately imply the compositionality of CCS-style parallel composition. ■

The following proposition will be helpful in our proof of compositionality for the $[\![\cdot]\!]^{\mathrm{MUST}}$ semantics:

Proposition 3.2.31 Let Act be a set of labels, let $\langle q_1, D_{q_1}\rangle \in \mathit{augment}(\mathit{extend}_{Act}(\langle p_1, D_{p_1}\rangle))$, let $\langle q_2, D_{q_2}\rangle \in \mathit{augment}(\mathit{extend}_{Act}(\langle p_2, D_{p_2}\rangle))$, and let $\langle q, D_q\rangle \in \langle q_1, D_{q_1}\rangle \|_A \langle q_2, D_{q_2}\rangle$. Then there are some prefixes $\langle p_1', D_{p_1'}\rangle, \langle p_2', D_{p_2'}\rangle$ of $\langle p_1, D_{p_1}\rangle, \langle p_2, D_{p_2}\rangle$, respectively, such that $\langle q, D_q\rangle \in \mathit{augment}(\mathit{extend}_{Act}(\langle p_1', D_{p_1'}\rangle \|_A \langle p_2', D_{p_2'}\rangle))$.

Proof. By definition, $\langle q, D_q\rangle = \langle q_1, D_{q_1}\rangle \|_A^f \langle q_2, D_{q_2}\rangle$ for some label-preserving, order-non-contradicting bijection f from $\{e \in \mathrm{Events}_{q_1} : l_{q_1}(e) \in A\}$ to $\{e' \in \mathrm{Events}_{q_2} : l_{q_2}(e') \in A\}$. Let f' be f restricted to $\mathrm{Events}_{p_1} \times \mathrm{Events}_{p_2}$. Let p_1' be the prefix of p_1 with carrier $\{x \in \mathrm{Events}_{p_1} :$ for all $y \in \mathrm{Events}_{p_1}$, if $y \leq_{p_1} x$ and $l_{p_1}(y) \in A$ then $f(y) \in \mathrm{Events}_{p_2}\}$, and let $D_{p_1'} = \{d \in D_{p_1} : d \subseteq p_1'\}$. Similarly, let p_2' be the prefix of p_2 with carrier $\{x \in \mathrm{Events}_{p_2} :$ for all $y \in$

Events$_{p_2}$, if $y \leq_{p_2} x$ and $l_{p_2}(y) \in A$ then $f^{-1}(y) \in$ Events$_{p_1}$}, and let $D_{p_2'} = \{d \in D_{p_2} : d \subseteq p_2'\}$. It is straightforward but tedious to show that $\langle q, D_q\rangle \in augment(extend_{Act}(\langle p_1', D_{p_1'}\rangle \|_A^{f'} \langle p_2', D_{p_2'}\rangle))$; the details are left to the reader. ∎

Theorem 3.2.32 $[\![\cdot]\!]^{\text{MUST}}$ and $[\![\cdot]\!]^{\text{TEST}}$ are compositional for all the WT Net operators, *except for* split refinements and the CCS choice operator, $+_M$.

Proof. Let Act be a finite alphabet containing $\surd$, let $a, a_L, a_R, a_+, a_- \in Act$, let $A \subseteq Act$, let f be a function from Act to Act such that for all $\alpha \in Act$, $f(\alpha) = \surd$ iff $\alpha = \surd$, and let Act' be a finite set of labels containing $\surd$. Furthermore, let $\langle N, Act\rangle, \langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets.

It is straightforward but tedious to show that the following identities hold, where the operations on the right-hand side of the equations are those defined in Definition 3.2.29. We prove the equalities for CSP-style parallel composition and hiding; the details of the remaining cases are left to the reader.

$$[\![\langle N, Act\rangle \; grow \; Act']\!]^{\text{MUST}} = [\![\langle N, Act\rangle]\!]^{\text{MUST}} \; grow \; Act'$$

$$[\![\langle N, Act\rangle \; shrink \; Act']\!]^{\text{MUST}} = [\![\langle N, Act\rangle]\!]^{\text{MUST}} \; shrink \; Act'$$

$$[\![a.\langle N, Act\rangle]\!]^{\text{MUST}} = a.[\![\langle N, Act\rangle]\!]^{\text{MUST}}$$

$$[\![\tau.\langle N, Act\rangle]\!]^{\text{MUST}} = [\![\langle N, Act\rangle]\!]^{\text{MUST}}$$

$$[\![\langle N, Act\rangle \backslash a]\!]^{\text{MUST}} = [\![\langle N, Act\rangle]\!]^{\text{MUST}} \backslash a$$

$$[\![\langle N, Act\rangle [f]]\!]^{\text{MUST}} = [\![\langle N, Act\rangle]\!]^{\text{MUST}}[f]$$

$$[\![\langle N, Act\rangle - a]\!]^{\text{MUST}} = [\![\langle N, Act\rangle]\!]^{\text{MUST}} - a$$

$$[\![\langle N_1, Act\rangle ; \langle N_2, Act\rangle]\!]^{\text{MUST}} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}} ; [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}$$

$$[\![\langle N_1, Act\rangle \oplus \langle N_2, Act\rangle]\!]^{\text{MUST}} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}} \oplus [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}$$

$$[\![\langle N_1, Act\rangle \; || \; \langle N_2, Act\rangle]\!]^{\text{MUST}} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}} \; || \; [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}$$

$$[\![\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle]\!]^{\text{MUST}} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}} \|_{A \cup \{\surd\}} [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}$$

$$[\![\langle N_1, Act\rangle \mid \langle N_2, Act\rangle]\!]^{\text{MUST}} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}} \mid [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}$$

$$[\![choice_{(a,a_L,a_R)}(\langle N, Act\rangle)]\!]^{\text{MUST}} = choice_{(a,a_L,a_R)}([\![\langle N, Act\rangle]\!]^{\text{MUST}})$$

To prove the equality for CSP-style parallel composition, we first state without proof the easily proved fact that:

$$\begin{aligned}
&\textit{pomset-failures}(\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle) = \\
&\quad \{\langle p, F\rangle : \text{ there are some } \langle p_1, F_1\rangle \in \textit{pomset-failures}(\langle N_1, Act\rangle), \langle p_2, F_2\rangle \in \textit{pomset-failures}(\langle N_2, Act\rangle) \\
&\qquad\quad \text{such that } p \in p_1\|_{A\cup\{\surd\}} p_2,\ F - (A \cup \{\surd\}) \subseteq F_1 \cap F_2 \text{ and } F \cap (A \cup \{\surd\}) \subseteq F_1 \cup F_2\}
\end{aligned}$$

$$\begin{aligned}
&\textit{pomset-divergences}(\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle) = \\
&\quad \bigcup\{\langle p_1, D_1\rangle \|_{A\cup\{\surd\}} \langle p_2, D_2\rangle : \langle p_1, D_1\rangle \in \textit{pomset-divergences}(\langle N_1, Act\rangle) \cup \textit{pomset-failures}(\langle N_1, Act\rangle), \\
&\qquad\qquad \langle p_2, D_2\rangle \in \textit{pomset-divergences}(\langle N_2, Act\rangle) \cup \textit{pomset-failures}(\langle N_2, Act\rangle), \\
&\qquad\qquad D_1 \text{ and } D_2 \text{ are (possibly empty) downward-closed subsets of} \\
&\qquad\qquad \text{Events}_{p_1} \text{ and Events}_{p_2}, \text{ respectively, and } D_1 \cup D_2 \neq \emptyset\}))
\end{aligned}$$

For one direction of the desired equality, let $\langle r, D_r\rangle \in \textit{snd}([\![\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle]\!]^{\text{MUST}})$; then $\langle r, D_r\rangle \in \textit{augment}(\textit{extend}_{Act}(\langle p, D_p\rangle))$ for some pomset-divergence $\langle p, D_p\rangle$ of $\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle$. It then follows easily from the highlighted fact above, the definition of $[\![\cdot]\!]^{\text{MUST}}$, and Definition 3.2.29 that $\langle r, D_r\rangle \in [\![\langle N_1, Act\rangle]\!]^{\text{MUST}} \|_{A\cup\{\surd\}} [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}$. The proof for pomset-failures is very similar and omitted. For the other direction, let

$$\langle r, D_r\rangle \in \textit{snd}([\![\langle N_1, Act\rangle]\!]^{\text{MUST}} \|_{A\cup\{\surd\}} [\![\langle N_2, Act\rangle]\!]^{\text{MUST}});$$

then $\langle r, D_r\rangle \in \textit{augment}(\textit{extend}_{Act}(\langle q, D_q\rangle))$ for some pomset-divergence $\langle q, D_q\rangle$ such that $\langle q, D_q\rangle \in \langle q_1, D_{q_1}\rangle \|_{A\cup\{\surd\}} \langle q_2, D_{q_2}\rangle$ for some $\langle q_1, D_{q_1}\rangle \in [\![\langle N_1, Act\rangle]\!]^{\text{MUST}}$ and $\langle q_2, D_{q_2}\rangle \in [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}$. In turn, $\langle q_i, D_{q_i}\rangle \in \textit{augment}(\textit{extend}_{Act}(\langle p_i, D_{p_i}\rangle))$ for some $\langle p_1, D_{p_1}\rangle$ and $\langle p_2, D_{p_2}\rangle$ that are pomset-divergences/pomset-failures of N_1 and N_2 respectively. Now, Proposition 3.2.31 implies that there are prefixes $\langle p_1', D_{p_1'}\rangle, \langle p_2', D_{p_2'}\rangle$ of $\langle p_1, D_{p_1}\rangle, \langle p_2, D_{p_2}\rangle$ respectively such that $\langle q, D_q\rangle \in \textit{augment}(\textit{extend}_{Act}(\langle p_1', D_{p_1'}\rangle \|_{A\cup\{\surd\}} \langle p_2', D_{p_2'}\rangle))$. By Proposition 3.2.14, $\langle p_1', D_{p_1'}\rangle, \langle p_2', D_{p_2'}\rangle$ are pomset-divergences/pomset-failures of N_1, N_2, hence the highlighted fact above together with the definition of $[\![\cdot]\!]^{\text{MUST}}$ implies that $\langle q, D_q\rangle \in \textit{snd}([\![\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle]\!]^{\text{MUST}}$. It now follows from Proposition 3.2.18 that $\langle r, D_r\rangle \in \textit{snd}([\![\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle]\!]^{\text{MUST}}$. The proof for pomset-failures in $\textit{fst}([\![\langle N_1, Act\rangle]\!]^{\text{MUST}} \|_{A\cup\{\surd\}} [\![\langle N_2, Act\rangle]\!]^{\text{MUST}})$ then follows easily from the highlighted fact above; we omit the details.

We now prove the equality for hiding. Since the definition of failure sets "looks through" firings of τ-transitions and failure sets are closed under subsets, it is straightforward to show that

$$\textit{pomset-failures}(\langle N, Act\rangle - a) = \{\langle p - a, F\rangle : \langle p, F \cup \{a\}\rangle \in \textit{pomset-failures}(\langle N, Act\rangle)\}$$

We recall that by definition of WT Nets, only a finite number of transitions are enabled under any reachable marking. Thus, it is possible for unbounded-length sequences of a-labeled events to be enabled after any prefix d of a pomset p only if either a divergence is enabled immediately after d or a divergence is enabled "along the way to d," *i.e.*, immediately after some pomset d' with $\langle d', \{\text{Events}_{d'}\}\rangle \sqsubseteq \langle d, \{\text{Events}_d\}\rangle$. In either case, it then follows easily from the definition of pomset-divergences that:

$$
\begin{aligned}
& \mathit{extend}_{Act}(\mathit{pomset\text{-}divergences}(\langle N, Act\rangle - a))) \\
& \quad = \\
& \mathit{extend}_{Act}(\{\langle p, D \cup D_p\rangle - a : \langle p, D\rangle \in \mathit{pomset\text{-}divergences}(\langle N, Act\rangle) \cup \mathit{pomset\text{-}failures}(\langle N, Act\rangle), \\
& \qquad D \cup D_p \neq \emptyset,\ \langle p, D \cup D_p\rangle \text{ is a pomset-divergence}, \\
& \qquad \text{and for all } n \geq 0, \text{ there is some } p_n \text{ with} \\
& \qquad \langle p_n, D\rangle \in \mathit{pomset\text{-}divergences}(\langle N, Act\rangle)) \cup \mathit{pomset\text{-}failures}(\langle N, Act\rangle)) \\
& \qquad \text{such that} \\
& \qquad \langle p, D_p\rangle \sqsubseteq \langle p_n, \{\mathrm{Events}_{p_n}\}\rangle, \text{ all events in } p_n - p \text{ are } a\text{-labeled}, \\
& \qquad \text{and for every } d \in D_p, \\
& \qquad \text{there is some } n\text{-length chain of } a\text{-labeled events in } p_n - p \\
& \qquad \text{whose downward closure restricted to } p \text{ is } d\})
\end{aligned}
$$

To prove one direction of the desired equality, let $\langle r, D_r\rangle \in \mathit{snd}([\![\langle N, act\rangle - a]\!]^{\mathrm{MUST}})$; then $\langle r, D_r\rangle \in \mathit{augment}(\mathit{extend}_{Act}(\langle p, D_p\rangle))$ for some pomset-divergence $\langle p, D_p\rangle$ of $\langle N, Act\rangle - a$. It then follows easily from the highlighted fact above, the definition of $[\![\cdot]\!]^{\mathrm{MUST}}$, and Definition 3.2.29 that $\langle r, D_r\rangle \in \mathit{snd}([\![\langle N, act\rangle]\!]^{\mathrm{MUST}} - a)$. The proof for pomset-failures $\langle r, F_r\rangle \in \mathit{fst}([\![\langle N, act\rangle - a]\!]^{\mathrm{MUST}})$ is very similar and is omitted.

For the other direction, let $\langle PF, PD, Act\rangle = [\![\langle N, Act\rangle]\!]^{\mathrm{MUST}}$ and let $\langle r, D_r\rangle \in \mathit{snd}([\![\langle N, Act\rangle]\!]^{\mathrm{MUST}} - a)$. Then by Definition 3.2.29, $\langle r, D_r\rangle \in \mathit{augment}(\mathit{extend}_{Act}(\langle p, D \cup D_p\rangle - a))$ for some pomset-divergence $\langle p, D \cup D_p\rangle$ and some sequence $\langle\langle p_n, D\rangle : n \geq 0\rangle$ such that

$$
\begin{aligned}
& \langle p, D\rangle \in PD \cup PF, \\
& \text{and for all } n \geq 0, \text{ there is some } p_n \text{ with } \langle p_n, D\rangle \in PD \cup PF \\
& \text{such that } \langle p, D_p\rangle \sqsubseteq \langle p_n, \{\mathrm{Events}_{p_n}\}\rangle, \text{ all events in } p_n - p \text{ are } a\text{-labeled}, \\
& \text{and for every } d \in D_p, \\
& \text{there is some } n\text{-length chain of } a\text{-labeled events in } p_n - p \\
& \text{whose downward closure restricted to } p \text{ is } d
\end{aligned}
$$

For one case, suppose that $\langle p, D\rangle$ and all of the $\langle p_n, D\rangle$ are in $\mathit{pomset\text{-}divergences}(\langle N, Act\rangle) \cup \mathit{pomset\text{-}failures}(\langle N, Act\rangle)$. Then it follows by the highlighted equality above that

$$\langle p, D \cup D_p\rangle - a \in \mathit{extend}_{Act}(\mathit{pomset\text{-}divergences}(\langle N, Act\rangle - a)),$$

and thus that $\langle r, D_r\rangle \in \mathit{snd}([\![\langle N, act\rangle - a]\!]^{\mathrm{MUST}})$.

For another case, suppose that all of the $\langle p_n, D\rangle \in \mathit{augment}(\mathit{pomset\text{-}divergences}(\langle N, Act\rangle)) \cup \mathit{augment}(\mathit{pomset\text{-}failures}(\langle N, Act\rangle))$. Then there is some sequence $\langle\langle q_n, D_n\rangle : n \geq 0\rangle$ such that each $\langle p_n, D\rangle \succeq \langle q_n, D_n\rangle \in \mathit{pomset\text{-}divergences}(\langle N, Act\rangle) \cup \mathit{pomset\text{-}failures}(\langle N, Act\rangle)$. Since Events_p is finite, there must be some subsequence $\langle\langle q_{n_k}, D_{n_k}\rangle : k \geq 0\rangle$ and some set such that for all $i \geq 0$, all the D_{n_i} are identical to some common D' and the ordering of all the q_{n_i} restricted to Events_p is identical. Let q be the pomset with this common ordering and with the same events and labels as p; it is easy to see that $\langle q, D'\rangle \preceq \langle p, D\rangle$. Furthermore, assuming without loss of generality that $n_0 > |p|$, it is easy to see that there is some set D_q of downward-closed sets of Events_q such that $D_p \subseteq \{\mathit{down}_p(d) : d \in D_q\}$, and $\langle q, D'\rangle$, D_q, and the sequence $\langle\langle q_{n_k}, D_{n_k}\rangle : k \geq 0\rangle$ are in the set on the right-hand side of the highlighted equality. By the first case, $\langle q, D' \cup D_q\rangle - a \in \mathit{extend}_{Act}(\mathit{pomset\text{-}divergences}(\langle N, Act\rangle - a))$. It is easy to see that $\langle q, D' \cup D_q\rangle - a \sqsubseteq \langle p, D \cup D_p\rangle - a$; thus by Proposition 3.2.18, $\langle p, D \cup D_p\rangle - a$ and $\langle r, D_r\rangle$ are in $snd([\![\langle N, act\rangle - a]\!]^{\text{MUST}})$.

For the last case, suppose that some $\langle p_i, D\rangle \notin augment(pomset\text{-}divergences(\langle N, Act\rangle)) \cup augment(pomset\text{-}failures(\langle N, Act\rangle))$. If for all such i, there is some $j > i$ with $\langle p_j, D\rangle \in augment(pomset\text{-}divergences(\langle N, Act\rangle)) \cup augment(pomset\text{-}failures(\langle N, Act\rangle))$, then substituting p_j for p_i yields a sequence that satisfies the earlier case. Otherwise, there must be some $\langle p_i, D\rangle \notin augment(pomset\text{-}divergences(\langle N, Act\rangle)) \cup augment(pomset\text{-}failures(\langle N, Act\rangle))$ such that for all $j > i$,

$$\langle p_j, D\rangle \notin augment(pomset\text{-}divergences(\langle N, Act\rangle)) \cup augment(pomset\text{-}failures(\langle N, Act\rangle)).$$

It is clear that there must be some sequence $\langle\langle q_n, D_n\rangle : n \geq 0\rangle$ such that each $\langle q_n, D_n\rangle \in min_{\sqsubseteq}(pomset\text{-}divergences(\langle N, Act\rangle))$ and $\langle p_n, D\rangle \in augment(extend_{Act}(\langle q_n, D_n\rangle))$. Let $p\text{-}related(D_n)$ be the set of $d' \cap \text{Events}_p$ such that $d' \in D_n$. It is easy to see that the number of distinct sets $p\text{-}related(D_n)$ is finite; hence there must be some subsequence $\langle\langle q_{n_k}, D_{n_k}\rangle : k \geq 0\rangle$ such that all the $p\text{-}related(D_{n_k})$ sets are equal. By Proposition 3.2.14, $pomset\text{-}divergences(\langle N, Act\rangle)$ is a prefix-closed set, and so we can assume without loss of generality that for all $n_k \geq 0$, there is no event $x \in q_{n_k}$ such that $down_{q_{n_k}}(x) \supseteq d'$ for some $d' \in D_{n_k}$. Using a straightforward finiteness argument on the length of chains that are unbounded in the p_{n_k} but bounded in the q_{n_k}, it is easy to show that there is some some subsequence $\langle\langle q'_{n_l}, D'_{n_l}\rangle : l \geq 0\rangle$ of $\langle\langle q_{n_k}, D_{n_k}\rangle : k \geq 0\rangle$, some set $D_{q'}$ of prefixes of q'_{n_0}, and some D' in the set on the right-hand side of the equality such that all the $D'_{n_l} = D'$. Furthermore, it is easy to show that $\langle p, D \cup D_p\rangle - a \in augment(extend_{Act}(\langle q'_{n_0}, D' \cup D_{q'}\rangle - a))$. By the first case, $\langle q'_{n_0}, D_{n_0} \cup D_{q'}\rangle - a \in augment(pomset\text{-}divergences(\langle N, Act\rangle - a))$. Hence, by Proposition 3.2.18, $\langle p, D \cup D_p\rangle - a \in snd([\![\langle N, act\rangle - a]\!]^{\text{MUST}})$ and hence so is $\langle r, D_r\rangle$. The proof that $fst([\![\langle N, act\rangle - a]\!]^{\text{MUST}}) \supseteq fst([\![\langle N, act\rangle]\!]^{\text{MUST}} - a)$ is similar and is left to the reader.

Proposition 2.2.19 and the equalities for alphabet expansion and shrinking, CSP-style parallel composition, choice refinements, and hiding together immediately imply the compositionality of CCS-style parallel composition. ■

It is easy to show that for *sequential nets*, $[\![\cdot]\!]^{\text{MUST}}$-equivalence and $[\![\cdot]\!]^{\text{TEST}}$-equivalence respectively coincide with MUST-equivalence and Testing-equivalence. Thus, as a simple consequence of Proposition 3.1.6, the $[\![\cdot]\!]^{\text{MUST}}$ and $[\![\cdot]\!]^{\text{TEST}}$ semantics are compositional for split refinements on *sequential nets*.

However, in general:

Proposition 3.2.33 $[\![\cdot]\!]^{\text{MUST}}$ and $[\![\cdot]\!]^{\text{TEST}}$ are not compositional for split refinements or the CCS choice operator, $+_M$.

Proof. For the proof for split refinements, let $\langle N_1, Act\rangle$ and $\langle N_2, Act\rangle$ be the nets illustrated in Figure 3-4, and let $Act = \{a, a_+, a_-, b\}$; this example is due to Frits Vaandrager [38]. It is straightforward to show that $[\![\langle N_1, Act\rangle]\!]^{\text{MUST}} = [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}$ and that $[\![\langle N_1, Act\rangle]\!]^{\text{TEST}} = [\![\langle N_2, Act\rangle]\!]^{\text{TEST}}$.
However,

$$\langle a_+, \{b\}\rangle \in snd([\![split_{(a,a_+,a_-)}(\langle N_1, Act\rangle)]\!]^{\text{MUST}}) - snd([\![split_{(a,a_+,a_-)}(\langle N_2, Act\rangle)]\!]^{\text{MUST}}).$$

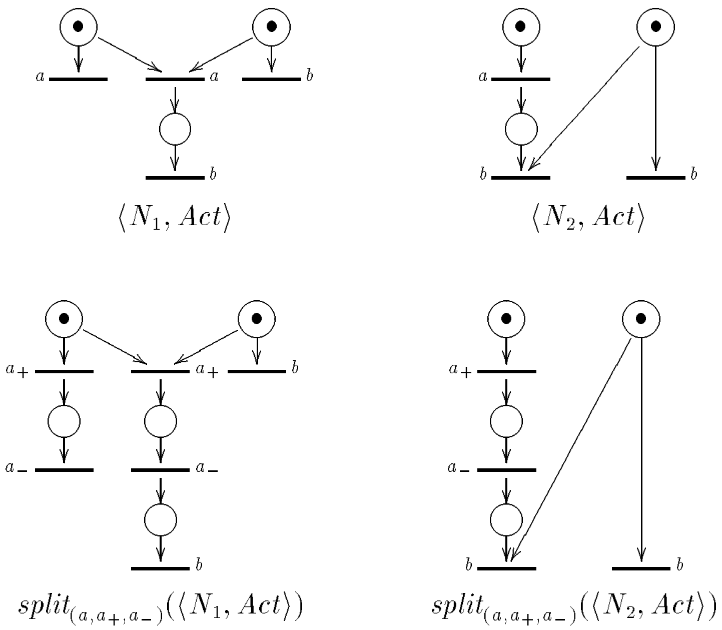


Figure 3-4: Standard Example for Split Refinements

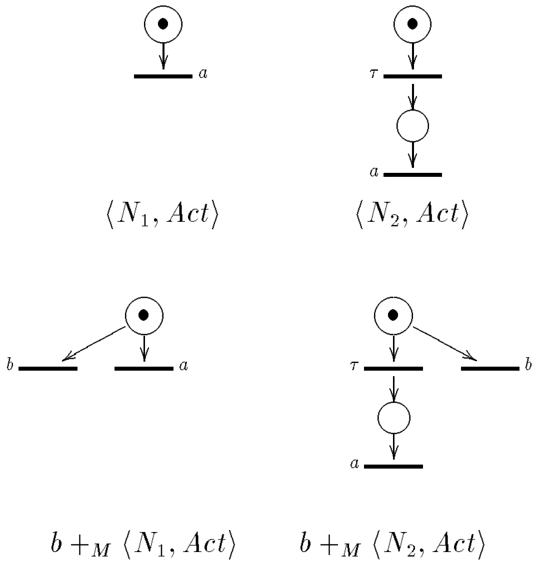


Figure 3-5: Standard Example for CCS choice

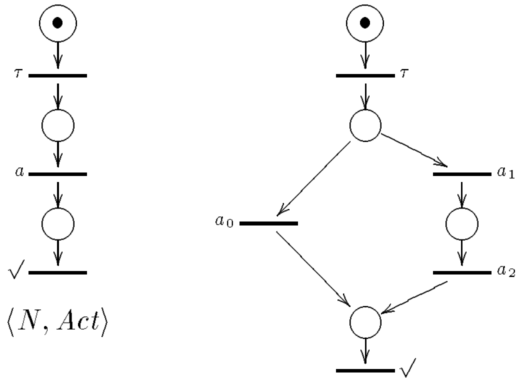


Figure 3-6: An Example of Duplicate-Splitting

We note that neither $\langle N_1, Act\rangle$ nor $\langle N_2, Act\rangle$ is a sequential net, since both of them can fire an a-labeled transition concurrently with a b-labeled transition.

For the proof of $+_M$, let $\langle N_1, Act\rangle$ and $\langle N_2, Act\rangle$ be the nets illustrated in Figure 3-5, and let $Act = \{a, b\}$. It is straightforward to show that $[\![\langle N_1, Act\rangle]\!]^{\text{MUST}} = [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}$ and that $[\![\langle N_1, Act\rangle]\!]^{\text{TEST}} = [\![\langle N_2, Act\rangle]\!]^{\text{TEST}}$. However,

$$\langle\emptyset, \{b\}\rangle \in snd([\![\langle N_2, Act\rangle +_M b]\!]^{\text{MUST}}) - snd([\![\langle N_1, Act\rangle +_M b]\!]^{\text{MUST}}).$$

■

The difficulty with split refinements is that they make visible the failure sets of the net after transitions have "half-fired", while the $[\![\cdot]\!]^{\text{MUST}}$ semantics does not keep track of this information. To correct this difficulty in our semantics, we first "duplicate-split" our nets; in particular, we "duplicate" every visible transition, then simultaneously "split" every duplicate transition into two consecutive transitions labeled a_1 and a_2, where a is the label of the original transition. Furthermore, we relabel with a_0 every visible transition of the original net, where a is the label of the original transition. We leave all τ-labeled and $\surd$-labeled transitions untouched. Figure 3-6 gives an example.

More formally:

Definition 3.2.34 Let $\langle N, Act\rangle$ be a WT Net. Then $\langle P, Act'\rangle = dupl\text{-}split(\langle N, Act\rangle)$ is defined as:

$$Act' = \{a_i : a \in Act - \{\surd\} \text{ and } 0 \leq i \leq 2\} \cup \{\surd\}$$

$$S_P = S_N \uplus \{(*, t) : t \in T_N \text{ and } l_N(t) \notin \{\surd, \tau\}\}$$

$$T_P = T_N \uplus \{(t,1),(t,2) : t \in T_N \text{ and } l_N(t) \notin \{\surd, \tau\}\}$$

$$\begin{aligned} pre_P(t) &= pre_N(t) \\ pre_P((t,1)) &= pre_N(t) \\ pre_P((t,2)) &= \{(*,t)\} \end{aligned}$$

$$\begin{aligned} post_P(t) &= post_N(t) \\ post_P((t,1)) &= \{(*,t)\} \\ post_P((t,2)) &= post_N(t) \end{aligned}$$

$$\begin{aligned} l_P(t) &= \begin{cases} l_N(t)_0 & \text{if } l_N(t) \notin \{\surd, \tau\} \\ l_N(t) & \text{otherwise} \end{cases} \\ l_P((t,1)) &= l_N(t)_1 \\ l_P((t,2)) &= l_N(t)_2 \end{aligned}$$

$$Start_P = Start_N$$

We note that:

Proposition 3.2.35 WT Nets are closed under *dupl-split*.

The proof is simple and is left to the reader.

The difficulty with the $+_M$ operator is that the $[\![\cdot]\!]^{\text{MUST}}$ semantics does not keep track of initial firings of τ-transitions; to correct this difficulty, we simply $+_M$ the *dupl-split* nets with a fresh, distinguished action γ, and take the $[\![\cdot]\!]^{\text{MUST}}$ semantics of the resulting net:

Definition 3.2.36 Let $\langle N, Act\rangle$ be a WT Net and assume without loss of generality that $\gamma \notin Act$. Then:

$$\begin{aligned} [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} &\stackrel{\text{def}}{=} [\![\gamma +_M (\textit{dupl-split}(\langle N, Act\rangle)\ \textit{grow}\ \{\gamma\})]\!]^{\text{MUST}} \\ [\![\langle N, Act\rangle]\!]^{\text{TEST}}_{\text{split-}\gamma} &\stackrel{\text{def}}{=} \langle [\![\langle N, Act\rangle]\!]^{\text{MAY}}, [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}\rangle \end{aligned}$$

Theorem 3.2.37 $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$, and $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ on WT Nets are respectively adequate for MAY-equivalence, MUST-equivalence, and Testing-equivalence.

Proof. From the definition of *dupl-split* and $\gamma +_M$ and Proposition 3.2.18, it is straightforward to show that

$$\begin{aligned} \mathcal{F}(\langle N, Act\rangle) = \{\langle v, F\rangle : {} & \langle v, F\rangle \text{ is a linearly-ordered pomset-failure over } Act \\ & \text{and } \langle v[f], \{f(a) : a \in F\}\rangle \in \textit{fst}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}), \\ & \text{where } f(a) = a_0 \text{ for all } a \in Act - \{\surd\} \text{ and } f(\surd) = \surd\} \end{aligned}$$

$$\mathcal{D}(\langle N, Act\rangle) = \{v : v \text{ is a linearly-ordered pomset over } Act$$

and $\langle v[f], \{\text{Events}_v\}\rangle \in \mathit{snd}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma})$,
where $f(a) = a_0$ for all $a \in Act - \{\surd\}$ and $f(\surd) = \surd\}$

$$Act = \{a : a_0 \in \mathit{third}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma})\} \cup \{\surd\}$$

from which the adequacy of $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$ follows easily. The adequacy of $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ is an immediate consequence of this fact and Theorem 3.2.12. ∎

The α_1-labeled transitions in *dupl-split*$(\langle N, Act\rangle)$ yield essential information about the failures of the net $\langle N, Act\rangle$ after some transitions are "half-split." On the other hand, the α_2-labeled transitions yield no new information. In fact, as we will show below, the pomset-failures and pomset-divergences of *dupl-split*$(\langle N, Act\rangle)$ that contain any α_2-labeled events can be fully recovered from those contain no α_2-labeled events by "splitting" some α_0-labeled events.

We first observe that:

Proposition 3.2.38 Let $\langle N, Act\rangle$ be a WT Net, and let p be a pomset-trace of *dupl-split*$(\langle N, Act\rangle)$. Then:

- For any $a \in Act$, if p does not contain any a_2-labeled events, then every a_1-labeled event is a maximal event in p.
- Let $\langle p, F\rangle$ be a pomset-failure of *dupl-split*$(\langle N, Act\rangle)$. For any $a \in Act$, if p does not contain any a_2-labeled events and does contain some a_1-labeled event, then $a_2 \notin F$.
- Let $\langle p, D\rangle$ be a pomset-divergence of *dupl-split*$(\langle N, Act\rangle)$. For any $a \in Act$, if p does not contain any a_2-labeled events, then no $d \in D$ contains any a_1-labeled events.

The proposition is a simple consequence of the definitions of pomset-traces, pomset-divergences, and *dupl-split*; the details are left to the reader.

Definition 3.2.39 Let Act' be a finite alphabet such that for some finite alphabet Act, $Act' = \{\gamma, \surd\} \cup \{a_i : a \in Act \text{ and } 0 \le i \le 2\}$. Let PF be a set of pomset-failures over Act', and let PD be a set of pomset-divergences over Act'. Then:

$$\begin{array}{rcl} \mathit{1\text{-}2\text{-}respect}(PF) & \stackrel{\text{def}}{=} & \{\langle p, F\rangle \in PF : \text{for every label } a \in Act, \\ & & \quad p \text{ has no } a_2\text{-labeled events} \\ & & \quad \text{and all } a_1\text{-labeled events in } p \text{ are maximal in } p\} \\ \mathit{1\text{-}2\text{-}respect}(PD) & \stackrel{\text{def}}{=} & \{\langle p, D\rangle \in PD : \text{for every label } a \in Act, \\ & & \quad p \text{ has no } a_2\text{-labeled events,} \\ & & \quad \text{all } a_1\text{-labeled events in } p \text{ are maximal in } p, \\ & & \quad \text{and no } d \in D \text{ contains any } a_1\text{-labeled events}\} \end{array}$$

Definition 3.2.40 Let Act' be a finite alphabet such that for some finite alphabet Act, $Act' = \{\gamma, \surd\} \cup \{a_i : a \in Act \text{ and } 0 \le i \le 2\}$. Let p be a pomset over Act', let $\langle p, F\rangle$ be a pomset-failure over Act', let $\langle p, D\rangle$ be a pomset-divergence, and let $X \subseteq \{x \in \text{Events}_p : l_p(x) = a_0 \text{ for some } a \in Act\}$. Then *0-split*$_X(p)$ is defined to be the pomset q with all events in X split, *i.e.*:

- $\text{Events}_q = (\{(y,0) : y \in \text{Events}_p - X) \cup \{(y,1),(y,2) : y \in X\}$.
- $l_q((y,i)) = \begin{cases} l_p(y) & \text{if } i = 0 \\ l_p(y)_i & \text{otherwise} \end{cases}$
- $(y,i) <_q (y',j)$ iff either $y <_p y'$ or ($y =_p y'$ and $i < j$).

Furthermore, $\textit{0-split}_X(\langle p, D\rangle) \stackrel{\text{def}}{=} \langle \textit{0-split}_X(p), \{\textit{0-split}_X(d) : d \in D\}\rangle$. We then define:

$$\textit{0-split}(p) \stackrel{\text{def}}{=} \{\textit{0-split}_X(p) : X \subseteq \{x \in \text{Events}_p : l_p(x) = a_0 \text{ for some } a \in Act\}\}$$

$$\textit{0-split}(\langle p, F\rangle) \stackrel{\text{def}}{=} \{\langle q, F\rangle : q \in \textit{0-split}(p)\}$$

$$\textit{0-split}(\langle p, D\rangle) \stackrel{\text{def}}{=} \{\textit{0-split}_X(\langle p, D\rangle) : X \subseteq \{x \in \text{Events}_p : l_p(x) = a_0 \text{ for some } a \in Act\}\}$$

We lift *0-split* to sets of individuals by point-wise union.

We remark that pomsets, pomset-failures, and pomset-divergences are preserved under *0-split*; the details are left to the reader.

As promised, the pomset-failures and pomset-divergences of duplicate-split nets can be recovered from *1-2-respecting* pomsets by *0-splitting*.

Proposition 3.2.41 Let $\langle N, Act\rangle$ be a WT Net. Then:

$$\begin{aligned}&\textit{pomset-failures}(\textit{dupl-split}(\langle N, Act\rangle)) = \\ &\quad \textit{0-split}(\textit{1-2-respect}(\textit{pomset-failures}(\textit{dupl-split}(\langle N, Act\rangle))))\end{aligned}$$

$$\begin{aligned}&\textit{pomset-divergences}(\textit{dupl-split}(\langle N, Act\rangle)) = \\ &\quad \textit{0-split}(\textit{1-2-respect}(\textit{pomset-divergences}(\textit{dupl-split}(\langle N, Act\rangle))))\end{aligned}$$

The proof is a straightforward consequence of the definitions of pomset-failures, pomset-divergences, and duplicate-splitting; the details are left to the reader.

The presence of α_2-labeled events does complicate split and choice refinements since corresponding α_1 and α_2 events in a pomset-trace might not "match up" correctly during refinement; so, we restrict attention to pomsets without α_2-labeled events. Similar to Proposition 3.2.41, we will be able to fully recover the refined α_2-labeled events from the refined α_0-labeled events.

Definition 3.2.42 Let Act' be a finite alphabet such that for some finite alphabet Act, $Act' = \{\gamma, \surd\} \cup \{a_i : a \in Act \text{ and } 0 \le i \le 2\}$. Let p be a pomset such that no event in p is labeled b_2 for any $b \in Act$, let $\langle p, F\rangle$ be a pomset-failure over Act', let $\langle p, D\rangle$ be a pomset-divergence over Act', and let a, a_L, a_R be labels in Act. Then

$$\begin{aligned}\textit{0-1-choice}_{(a,a_L,a_R)}(\langle p, F\rangle) \stackrel{\text{def}}{=} \{\langle q, F_q\rangle : {}& q \in \textit{choice}_{(a_0,a_{L0},a_{R0})}(\textit{choice}_{(a_1,a_{L1},a_{R1})}(p)) \\ & F_q \subseteq F_p \cup \{a_0, a_1, a_2\} \\ & \text{and if } a_{L0} \in F_q \text{ or } a_{R0} \in F_q \text{ then } a_0 \in F_p \\ & \quad \text{if } a_{L1} \in F_q \text{ or } a_{R1} \in F_q \text{ then } a_1 \in F_p \\ & \quad \text{if } a_{L2} \in F_q \text{ then there is no } a_{L1}\text{-labeled event in } q \\ & \quad \text{if } a_{R2} \in F_q \text{ then there is no } a_{R1}\text{-labeled event in } q\}))\end{aligned}$$

Furthermore,

$$\textit{0-1-choice}_{(a,a_L,a_R)}(\langle p,D\rangle) \stackrel{\text{def}}{=} \textit{choice}_{(a_0,a_{L0},a_{R0})}(\textit{choice}_{(a_1,a_{L1},a_{R1})}(\langle p,D\rangle))$$

In defining split refinements, we want to "fully split" each a_0-labeled event into the sequence $a_{+0}.a_{-0}$. Furthermore, we want a_1-labeled events to simulate half-firings of splits, and hence we have three choices for each a_1-labeled event: relabel the event with a_{+1}, relabel the event with a_{+0}, or split the event into $a_{+0}.a_{-1}$. (The other two possibilities, $a_{+1}.a_{+2}$ and $a_{+1}.a_{+2}.a_{-1}$, can be obtained from *0-splitting*.) These choices are reflected below:

Definition 3.2.43 Let Act' be a finite alphabet such that for some finite alphabet Act, $Act' = \{\gamma,\surd\} \cup \{a_i : a \in Act \text{ and } 0 \le i \le 2\}$. Let p be a pomset such that no event in p is labeled b_2 for any $b \in Act$, let $\langle p,F\rangle$ be a pomset-failure over Act', let $\langle p,D\rangle$ be a pomset-divergence over Act', and let a, a_+, a_- be labels in Act. Furthermore, let X_0, X_1, X_2 be a partition of the set of a_1-labeled events of p. Then $\textit{0-1-split}_{(a,a_+,a_-)}$ is defined to be the pomset q such that

- $\text{Events}_q = \{(y,0) : y \in \text{Events}_p \text{ and } l_p(y) \notin \{a_0,a_1\}\}$
  $\cup \{(y,1) : y \in \text{Events}_p \text{ and } l_p(y) \in \{a_0,a_1\}\}$
  $\cup \{(y,2) : y \in \text{Events}_p \text{ and either } l_p(y) = a_0 \text{ or } y \in X_2\}$.
- $l_q((y,i)) = \begin{cases} l_p(y) & \text{if } i = 0 \\ a_{+0} & \text{if } i = 1 \text{ and } y \notin X_0 \\ a_{+1} & \text{if } i = 1 \text{ and } y \in X_0 \\ a_{-0} & \text{if } i = 2 \text{ and } y \notin X_2 \\ a_{-1} & \text{if } i = 2 \text{ and } y \in X_2 \end{cases}$
- $(y,i) <_q (y',j)$ iff either $y <_p y'$ or $(y =_p y'$ and $i < j)$.

We then define:

$$\textit{0-1-split}_{(a,a_+,a_-)}(p) \stackrel{\text{def}}{=} \{\textit{0-1-split}_{(a,a_+,a_-,X_0,X_1,X_2)}(p) : X_0,X_1,X_2 \text{ partition } \{x \in \text{Events}_p : l_p(x) = a_1\}\}$$

$$\begin{aligned}
\textit{0-1-split}_{(a,a_+,a_-)}(\langle p,D\rangle) \stackrel{\text{def}}{=} \{\langle p',D'\rangle : {} & p' = \textit{0-1-split}_{(a,a_+,a_-,X_0,X_1,X_2)}(p) \text{ and} \\
& D' = \{\textit{0-1-split}_{(a,a_+,a_-,X_0,X_1,X_2)}(d) : d \in D\} \\
& \text{for some } X_0,X_1,X_2 \text{ that partition } \{x \in \text{Events}_p : l_p(x) = a_1\}\} \\
\textit{0-1-split}_{(a,a_+,a_-)}(\langle p,F\rangle) \stackrel{\text{def}}{=} \{\langle p',F'\rangle : {} & p' = \textit{0-1-split}_{(a,a_+,a_-,X_0,X_1,X_2)}(p) \\
& \text{for some } X_0,X_1,X_2 \text{ that partition } \{x \in \text{Events}_p : l_p(x) = a_1\}, \\
& F' \subseteq F \cup \{a_0,a_1,a_2\}, \\
& \text{and if } a_{+0} \in F' \text{ or } a_{+1} \in F' \text{ then } a_0 \in F \text{ and } a_1 \in F \\
& \quad \text{if } X_0 \neq \emptyset \text{ then } a_{+2} \notin F' \\
& \quad \text{if } X_1 \neq \emptyset \text{ then } F' \cap \{a_{-0},a_{-1}\} = \emptyset \\
& \quad \text{if } X_2 \neq \emptyset \text{ then } a_{-2} \notin F'\}
\end{aligned}$$

The following definition will be helpful in proving the compositionality of the $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$ and $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ semantics for $+_M$:

Definition 3.2.44 Let PF be a set of pomset-failures over a finite alphabet Act. Then $init(PF) \stackrel{\mathrm{def}}{=} \{a \in Act\colon \langle a, \emptyset\rangle \in PF\}$.

The following definition will be used heavily in our proof of compositionality for $[\![\cdot]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$. In this definition, the presence of γ in failure sets is used to indicate that some initial τ-transitions have been fired.

Definition 3.2.45 Let Act be a finite alphabet containing the distinguished symbol $\surd$, let $Act' = \{a_i\colon a \in Act - \{\surd\}$ and $0 \leq i \leq 2\} \cup \{\gamma, \surd\}$, let PF, PF_1, PF_2 be sets of pomset-failures over Act', and let PD, PD_1, PD_2 be sets of pomset-divergences over Act'. Let $a, a_L, a_R, a_+, a_- \in Act - \{\surd\}$, let A be a subset of Act containing $\surd$, let f be a function from Act to Act such that for all $\alpha \in Act$, $f(\alpha) = \surd$ iff $\alpha = \surd$. Let $\langle PF_\gamma, PD_\gamma, Act'\rangle \stackrel{\mathrm{def}}{=} [\![\langle\gamma, Act'\rangle]\!]^{\mathrm{MUST}}$, and for all $a \in Act - \{\surd\}$, let $\langle PF_{a.\surd}, PD_{a.\surd}, Act'\rangle \stackrel{\mathrm{def}}{=} [\![\langle N_{a.\surd}, Act'\rangle]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$, where $N_{a.\surd}$ is a net that can perform exactly an a-transition causally followed by a $\surd$-transition, after which the net deadlocks.

The following definitions use the operators defined in Definition 3.2.29. The $||$ operator remains the same as in Definition 3.2.29.

$$\langle PF, PD, Act'\rangle \ \mathbf{grow}\ A \stackrel{\mathrm{def}}{=} \langle PF, PD, Act'\rangle \ \mathit{grow}\ A$$

$$\langle PF, PD, Act'\rangle \ \mathbf{shrink}\ A \stackrel{\mathrm{def}}{=} \langle PF, PD, Act'\rangle \ \mathit{shrink}\ A$$

$$a \ \mathbf{pref}\ \langle PF, PD, Act'\rangle \stackrel{\mathrm{def}}{=} \langle PF_\gamma, PD_\gamma, Act'\rangle +_M (\langle PF_{a.\surd}, PD_{a.\surd}, Act'\rangle; (\langle PF, PD, Act'\rangle \backslash \gamma))$$

$$\tau \ \mathbf{pref}\ \langle PF, PD, Act'\rangle \stackrel{\mathrm{def}}{=} \langle PF_\gamma, PD_\gamma, Act'\rangle +_M (\langle PF, PD, Act'\rangle \backslash \gamma)$$

$$\langle PF, PD, Act'\rangle \ \mathbf{rst}\ a \stackrel{\mathrm{def}}{=} (((\langle PF, PD, Act'\rangle \backslash a_0)\backslash a_1)\backslash a_2$$

$$\begin{aligned}\langle PF, PD, Act'\rangle \ \mathbf{rename\ with}\ f \stackrel{\mathrm{def}}{=} {} & \langle PF, PD, Act'\rangle[f'],\\ & \text{where } f'(a_i) = (f(a))_i \text{ for all } a \in Act \text{ and } 0 \leq i \leq 2,\\ & \text{and } f' \text{ is the identity on } \{\gamma, \surd\}\end{aligned}$$

$$\langle PF, PD, Act'\rangle \ \mathbf{hide}\ a \stackrel{\mathrm{def}}{=} (((\langle PF, PD, Act'\rangle - a_0) - a_1) - a_2$$

$$\langle PF_1, PD_1, Act'\rangle \ \mathbf{seq}\ \langle PF_2, PD_2, Act'\rangle \stackrel{\mathrm{def}}{=} \langle PF_1, PD_1, Act'\rangle; (\langle PF, PD, Act'\rangle \backslash \gamma)$$

$$\langle PF_1, PD_1, Act'\rangle \ \mathbf{internal\ choice}\ \langle PF_2, PD_2, Act'\rangle \stackrel{\mathrm{def}}{=} \tau \ \mathbf{pref}\ \langle PF_1, PD_1, Act'\rangle +_M \tau \ \mathbf{pref}\ \langle PF, PD, Act'\rangle$$

$$\begin{array}{lcl}
\langle PF_1, PD_1, Act\rangle +_M \langle PF_2, PD_2, Act\rangle \stackrel{\text{def}}{=} \langle PF', PD', Act\rangle & & \\
\qquad\qquad\text{where} & & \\
PF' & = & \{\langle\emptyset, F\rangle : \langle\emptyset, F\cup\{\gamma\}\rangle \in PF_1 \cup PF_2\} \\
 & & \cup\, \{\langle p, F\rangle \in PF_1\cup PF_2 :\ \text{either } p\neq\emptyset \text{ or } F\cap(init(PF_1)\cup init(PF_2)) = \emptyset\} \\
PD' & = & PD_1 \cup PD_2
\end{array}$$

$$\begin{array}{rcl}
\langle PF_1, PD_1, Act'\rangle\ \mathbf{CSP\text{-}parallel}_A\ \langle PF_2, PD_2, Act' \stackrel{\text{def}}{=} \langle PF', PD', Act'\rangle\rangle & & \\
\text{where} & & \\
\langle PF, PD, Act'\rangle & = & \textit{1-2-respect}(\langle PF_1, PD_1, Act'\rangle)\|_{A'}\,\textit{1-2-respect}(\langle PF_2, PD_2, Act'\rangle) \\
A' & = & \{a_i : a\in A-\{\surd\} \text{ and } 0\le i\le 2\}\cup\{\gamma,\surd\} \\
\text{and} & & \\
PF' & = & \textit{augment}(\textit{0-split}(PF))\cup \textit{implied-failures}_{Act'}(PD') \\
PD' & = & \textit{augment}(\textit{extend}_{Act'}(\textit{0-split}(PD)))
\end{array}$$

$$\begin{array}{rcl}
\mathbf{choice}_{(a,a_L,a_r)}(\langle PF, PD, Act'\rangle) \stackrel{\text{def}}{=} \langle PF', PD', Act'\rangle & & \\
\text{where} & & \\
\langle PF'', PD'', Act'\rangle & = & \textit{0-1-choice}_{(a,a_L,a_R)}(\textit{1-2-respect}(\langle PF, PD, Act'\rangle)) \\
\text{and} & & \\
PF' & = & \textit{augment}(\textit{0-split}(PF''))\cup \textit{implied-failures}_{Act'}(PD') \\
PD' & = & \textit{augment}(\textit{extend}_{Act'}(\textit{0-split}(PD'')))
\end{array}$$

$$\begin{array}{rcl}
\mathbf{split}_{(a,a_+,a_-)}(\langle PF, PD, Act'\rangle) \stackrel{\text{def}}{=} \langle PF', PD', Act'\rangle & & \\
\text{where} & & \\
\langle PF'', PD'', Act'\rangle & = & \textit{0-1-split}_{(a,a_L,a_R)}(\textit{1-2-respect}(\langle PF, PD, Act'\rangle)) \\
\text{and} & & \\
PF' & = & \textit{augment}(\textit{0-split}(PF''))\cup \textit{implied-failures}_{Act'}(PD') \\
PD' & = & \textit{augment}(\textit{extend}_{Act'}(\textit{0-split}(PD'')))
\end{array}$$

$\langle PF_1, PD_1, Act'\rangle$ **CCS-parallel** $PF_2, PD_2, Act' \stackrel{\text{def}}{=}$
$(((\langle PF_1', PD_1', Act''\rangle\ \mathbf{CSP\text{-}parallel}_{A\cup\{\gamma,\surd\}}\ \langle PF_2', PD_2', Act''\rangle)\ \mathbf{hide}\ A)\ \mathbf{shrink}\ Act'$
where
$\langle PF_1', PD_1', Act''\rangle = \sigma(\langle PF_1, PD_1, Act'\rangle\ \mathbf{grow}\ A)$
$\langle PF_2', PD_2', Act''\rangle = \sigma'(\langle PF_2, PD_2, Act'\rangle\ \mathbf{grow}\ A)$
and $\{a_1, \overline{a_1}, \ldots, a_k, \overline{a_k}\} = Act' - \{\gamma, \surd\}$,
$A = \{a_1', \overline{a_1'}, \ldots, a_k', \overline{a_k'}\}$ are distinct symbols not in Act',
σ is the sequence $\mathbf{choice}_{(a_1,a_1,a_1')}\cdot\mathbf{choice}_{(\overline{a_1},\overline{a_1},\overline{a_1'})}\ldots\mathbf{choice}_{(a_k,a_k,a_k')}\cdot\mathbf{choice}_{(\overline{a_k},\overline{a_k},\overline{a_k'})}$,
and σ' is the sequence $\mathbf{choice}_{(a_1,a_1,\overline{a_1'})}\cdot\mathbf{choice}_{(\overline{a_1},\overline{a_1},a_1')}\ldots\mathbf{choice}_{(a_k,a_k,\overline{a_k'})}\cdot\mathbf{choice}_{(\overline{a_k},\overline{a_k},a_k')}$

We now show:

Theorem 3.2.46 $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$ and $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ are compositional for split refinements, choice refinements, alphabet expansion and shrinking, and all of our CCS/CSP operators.

Proof. Let Act be a finite alphabet containing $\surd$, let $a, a_L, a_R, a_+, a_- \in Act$, let $A \subseteq Act$, let f be a function from Act to Act such that for all $\alpha \in Act$, $f(\alpha) = \surd$ iff $\alpha = \surd$, and let Act' be a finite set of labels containing $\surd$. Furthermore, let $\langle N, Act\rangle, \langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets.

The following identities hold, where the operations on the right-hand side of the equations are those defined in Definition 3.2.45:

$$[\![\langle N, Act\rangle \ \mathit{grow}\ Act']\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} \ \mathbf{grow}\ Act'$$

$$[\![\langle N, Act\rangle \ \mathit{shrink}\ Act']\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} \ \mathbf{shrink}\ Act'$$

$$[\![a.\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} = a \ \mathbf{pref}\ [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}$$

$$[\![\tau.\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} = \tau \ \mathbf{pref}\ [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}$$

$$[\![\langle N, Act\rangle \backslash a]\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} \ \mathbf{rst}\ a$$

$$[\![\langle N, Act\rangle [f]]\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} \ \mathbf{rename\ with}\ f$$

$$[\![\langle N, Act\rangle - a]\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} \ \mathbf{hide}\ a$$

$$[\![\langle N_1, Act\rangle ; \langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} \ \mathbf{seq}\ [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}$$

$$[\![\langle N_1, Act\rangle +_M \langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} +_M [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}$$

$$[\![\langle N_1, Act\rangle \oplus \langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} \ \mathbf{internal\ choice}\ [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}$$

$$[\![\langle N_1, Act\rangle \,||\, \langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} \,||\, [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}$$

$$[\![\langle N_1, Act\rangle ||_A \langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} \ \mathbf{CSP\text{-}parallel}_{A \cup \{\gamma, \surd\}}\ [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}$$

$$[\![\langle N_1, Act\rangle \mid \langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} \ \mathbf{CCS\text{-}parallel}\ \langle N_2, Act\rangle$$

$$[\![\mathit{choice}_{(a, a_L, a_R)}(\langle N, Act\rangle)]\!]^{\text{MUST}}_{\text{split-}\gamma} = \mathbf{choice}_{(a, a_L, a_R)}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma})$$

$$[\![\mathit{split}_{(a, a_+, a_-)}(\langle N, Act\rangle)]\!]^{\text{MUST}}_{\text{split-}\gamma} = \mathbf{split}_{(a, a_+, a_-)}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma})$$

The proof for CSP-style parallel composition is essentially the same as that in Theorem 3.2.32, except it uses the following easily proved fact:

$$\begin{aligned}
&\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle)\ \textit{grow}\ \{\gamma\})) = \\
&\quad \textit{0-split}(\{\langle p, F\rangle : \langle p_1, F_1\rangle \in \textit{1-2-respect}(\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N_1, Act\rangle)\ \textit{grow}\ \{\gamma\}))), \\
&\qquad \langle p_2, F_2\rangle \in \textit{1-2-respect}(\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N_2, Act\rangle)\ \textit{grow}\ \{\gamma\}))), \\
&\qquad \text{and } p \in p_1 \|_{A'} p_2,\ F - A' \subseteq F_1 \cap F_2 \text{ and } F \cap A' \subseteq F_1 \cup F_2, \\
&\qquad \text{where } A' = \{a_i : a \in A \text{ and } 0 \le i \le 2\} \cup \{\surd\})
\end{aligned}$$

$$\begin{aligned}
&\textit{pomset-divergences}(\gamma +_M (\textit{dupl-split}(\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle)\ \textit{grow}\ \{\gamma\})) = \\
&\quad \textit{0-split}(\bigcup\{\langle p_1, D_1\rangle \|_{A'} \langle p_2, D_2\rangle : \\
&\qquad \langle p_1, D_1\rangle \in \textit{1-2-respect}(\textit{pomset-divergences}(\gamma +_M (\textit{dupl-split}(\langle N_1, Act\rangle)\ \textit{grow}\ \{\gamma\}))) \\
&\qquad\qquad \cup\ \textit{1-2-respect}(\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N_1, Act\rangle)\ \textit{grow}\ \{\gamma\}))), \\
&\qquad \langle p_2, D_2\rangle \in \textit{1-2-respect}(\textit{pomset-divergences}(\gamma +_M (\textit{dupl-split}(\langle N_2, Act\rangle)\ \textit{grow}\ \{\gamma\}))) \\
&\qquad\qquad \cup\ \textit{1-2-respect}(\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N_2, Act\rangle)\ \textit{grow}\ \{\gamma\}))), \\
&\qquad D_1 \text{ and } D_2 \text{ are (possibly empty) downward-closed subsets of} \\
&\qquad \text{Events}_{p_1} \text{ and Events}_{p_2}, \text{ respectively, and } D_1 \cup D_2 \neq \emptyset, \\
&\qquad \text{and } A' = \{a_i : a \in A \text{ and } 0 \le i \le 2\} \cup \{\surd\})
\end{aligned}$$

The proofs for choice refinement and split refinement follow straightforwardly from Proposition 3.2.18 and the easily proved facts that:

$$\begin{aligned}
&\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\textit{choice}_{(a,a_L,a_R)}(\langle N, Act\rangle)))) = \\
&\quad \textit{0-split}(\textit{0-1-choice}_{(a,a_L,a_R)}(\textit{1-2-respect}(\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N, Act\rangle)\ \textit{grow}\ \{\gamma\})))))
\end{aligned}$$

$$\begin{aligned}
&\textit{pomset-divergences}(\gamma +_M (\textit{dupl-split}(\textit{choice}_{(a,a_L,a_R)}(\langle N, Act\rangle)))) = \\
&\quad \textit{0-split}(\textit{0-1-choice}_{(a,a_L,a_R)}(\textit{1-2-respect}(\textit{pomset-divergences}(\gamma +_M (\textit{dupl-split}(\langle N, Act\rangle)\ \textit{grow}\ \{\gamma\})))))
\end{aligned}$$

$$\begin{aligned}
&\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\textit{split}_{(a,a_+,a_-)}(\langle N, Act\rangle)))) = \\
&\quad \textit{0-split}(\textit{0-1-split}_{(a,a_+,a_-)}(\textit{1-2-respect}(\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N, Act\rangle)\ \textit{grow}\ \{\gamma\})))))
\end{aligned}$$

$$\begin{aligned}
&\textit{pomset-divergences}(\gamma +_M (\textit{dupl-split}(\textit{choice}_{(a,a_+,a_-)}(\langle N, Act\rangle)))) = \\
&\quad \textit{0-split}(\textit{0-1-split}_{(a,a_+,a_-)}(\textit{1-2-respect}(\textit{pomset-divergences}(\gamma +_M (\textit{dupl-split}(\langle N, Act\rangle)\ \textit{grow}\ \{\gamma\})))))
\end{aligned}$$

The proof for hiding is analogous to that in Theorem 3.2.32. The proofs of the remaining equalities are left to the reader. We remark that Proposition 2.2.18 and the equalities for prefixing and CCS choice together imply the compositionality of internal choice. Furthermore, Proposition 2.2.19 and the equalities for alphabet expansion and shrinking, CSP-style parallel composition, choice refinements, and hiding together imply the compositionality of CCS-style parallel composition.

The compositionality of $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ then follows easily from the above proofs together with Theorem 3.2.30. ∎

In fact, $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$ and $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ make just the right distinctions with respect to $[\![\cdot]\!]^{\text{MUST}}$, $[\![\cdot]\!]^{\text{TEST}}$, and our WT Net operators:

Theorem 3.2.47 $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$ and $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ are fully abstract for split refinements, choice refinements, and all of our CCS/CSP operators with respect to $[\![\cdot]\!]^{\text{MUST}}$ and $[\![\cdot]\!]^{\text{TEST}}$, respectively.

Proof. From the definition of *dupl-split* and γ, it is easy to see that

$$\begin{aligned} \mathit{fst}(\llbracket\langle N, Act\rangle\rrbracket^{\text{MUST}}) = \{\langle p, F\rangle : {} & \langle p, F\rangle \text{ is a pomset-failure over } Act \\ & \text{and } \langle p[f], \{f(a) : a \in F\}\rangle \in \mathit{fst}(\llbracket\langle N, Act\rangle\rrbracket^{\text{MUST}}_{\text{split-}\gamma}), \\ & \text{where } f(a) = a_0 \text{ for all } a \in Act - \{\surd\} \text{ and } f(\surd) = \surd\} \end{aligned}$$

$$\begin{aligned} \mathit{snd}(\llbracket\langle N, Act\rangle\rrbracket^{\text{MUST}}) = \{\langle p, D\rangle : {} & p \text{ is a pomset-divergence over } Act \\ & \text{and } \langle p[f], D\rangle \in \mathit{snd}(\llbracket\langle N, Act\rangle\rrbracket^{\text{MUST}}_{\text{split-}\gamma}), \\ & \text{where } f(a) = a_0 \text{ for all } a \in Act - \{\surd\} \text{ and } f(\surd) = \surd\} \end{aligned}$$

$$Act = \{a : a_0 \in \mathit{third}(\llbracket\langle N, Act\rangle\rrbracket^{\text{MUST}}_{\text{split-}\gamma})\} \cup \{\surd\}$$

from which adequacy follows easily.

Theorem 3.2.46 has shown that $\llbracket\cdot\rrbracket^{\text{MUST}}_{\text{split-}\gamma}$ and $\llbracket\cdot\rrbracket^{\text{TEST}}_{\text{split-}\gamma}$ are compositional for all the WT Net operators. To prove full abstraction, we observe that $\gamma +_M (\mathit{dupl\text{-}split}(\cdot)\ \mathit{grow}\ \{\gamma\})$ can be programmed by CCS choice and a finite sequence of choice and split refinements, together with some alphabet-expansion and shrinking. In particular, assuming that $\{a^1_i : a^1 \in Act \text{ and } 0 \leq i \leq 2\} \cap Act = \emptyset$,

$$\gamma +_M (\mathit{dupl\text{-}split}(\langle N, Act\rangle)\ \mathit{grow}\ \{\gamma\}) = \gamma +_M ((\sigma(\langle N, Act\rangle\ \mathit{grow}\ Act'))\ \mathit{shrink}\ Act')$$

where σ is the sequence $\mathit{split}_{(a^1,a^1_1,a^1_2)} \ldots \mathit{split}_{(a^k,a^k_1,a^k_2)} \cdot \mathit{choice}_{(a^1,a^1,a^1_0)} \ldots \mathit{choice}_{(a^k,a^k,a^k_0)}$,

$Act - \{\surd\} = \{a^1, \ldots, a^k\}$, $Act' = \{a^1_i : a^1 \in Act \text{ and } 0 \leq i \leq 2\} \cup \{\surd, \gamma\}$, and equality refers to net isomorphism. If $a^1_i \in Act$ for some $a^1 \in Act$ and some $0 \leq i \leq 2$, the equality above can be suitably modified to use different "fresh" variables and renaming. The theorem is then a simple consequence of this equality and the definition of $\llbracket\cdot\rrbracket^{\text{MUST}}_{\text{split-}\gamma}$ and $\llbracket\cdot\rrbracket^{\text{TEST}}_{\text{split-}\gamma}$. ∎

However, as we will prove in the next section, $\llbracket\cdot\rrbracket^{\text{MAY}}$, $\llbracket\cdot\rrbracket^{\text{MUST}}_{\text{split-}\gamma}$, and $\llbracket\cdot\rrbracket^{\text{TEST}}_{\text{split-}\gamma}$ are not *fully abstract* for our WT Net operators with respect to MAY-equivalence, MUST-equivalence, and Testing-equivalence, respectively. The complication is that these semantics make strictly more distinctions than our net contexts.

We remark here that keeping track of *concurrent divergences* is necessary for compositionality with respect to parallel composition. In particular, suppose we modify the definition of pomset-divergences, $\langle p, D\rangle$, so that D must be a singleton set. Then the redefined $\llbracket\cdot\rrbracket^{\text{MUST}}_{\text{split-}\gamma}$ semantics based on this modified version of pomset-divergences will not be compositional for parallel composition, which was the difficulty faced by Vogler [47, 49]. Our $\llbracket\cdot\rrbracket^{\text{MUST}}_{\text{split-}\gamma}$ semantics avoids this difficulty by keeping track of concurrent divergences, and resolves an open problem posed in [49]. The difficulty with keeping track of only single divergences is illustrated in Figure 3-7. It is easy to see that $\langle N_1, Act\rangle$ and $\langle N_2, Act\rangle$ have the same meanings under the redefined $\llbracket\cdot\rrbracket^{\text{MUST}}_{\text{split-}\gamma}$ semantics and that $\langle N_3, Act\rangle$ and $\langle N_4, Act\rangle$ have the same meanings under the redefined $\llbracket\cdot\rrbracket^{\text{MUST}}_{\text{split-}\gamma}$ semantics, where $Act = \{a, b, c, d\}$. However, $\langle p, \{e_a, e_c\}\rangle$ is a pomset-single-divergence of $\langle N_2, Act\rangle \mid\mid \langle N_4, Act\rangle$, while it is not an augmentation of any extension of

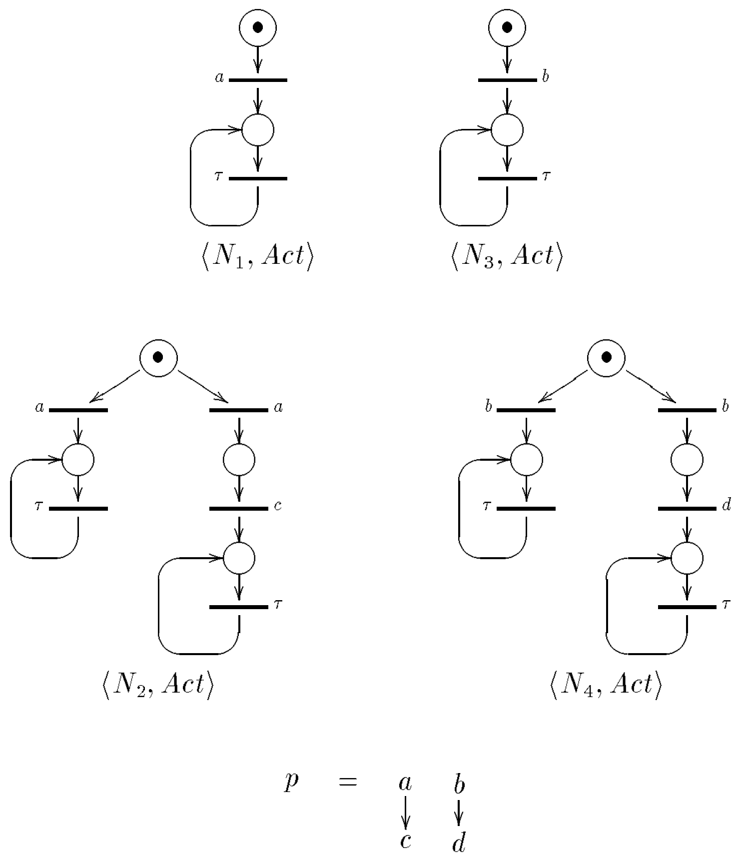


Figure 3-7: An Example of the Necessity of Concurrent Divergences

any pomset-single-divergence $\langle q, \{d\}\rangle$ of $\langle N_1, Act\rangle \mid\mid \langle N_3, Act\rangle$.

3.3 Fully Abstract Semantics

It turns out that $[\![\cdot]\!]^{\text{MAY}}$, $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$, and $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ make more distinctions than are apparent to a single experimenter. Namely, single experimenters can only detect differences between pomsets with *interval orderings* [32, 47]. We repeat the definition here:

Definition 3.3.1 A partial order $\leq$ is is an *interval ordering* iff whenever both $w < x$ and $y < z$, then either $w < z$ or $y < x$. A pomset p is an *interval pomset* iff $\leq_p$ is an interval ordering.

It is well-known (*cf.* [14]) that:

Lemma 3.3.2 ([14]) Every interval ordering, $\leq_p$, is order-isomorphic to a set of intervals of the real line, where by definition, (interval w) $<$ (interval x) iff every point in (interval w) strictly precedes every point in (interval x).

We define a corresponding version of "interval pomset-divergences":

Definition 3.3.3 A pomset-divergence $\langle p, D\rangle$ is an *interval pomset-divergence* iff p is an interval pomset and $D = \{d\}$ for some d that contains all the non-maximal events of p, *i.e.*, $d \supseteq \text{Events}_p - max(p)$.

We define the interval-MAY-, interval-MUST-, and interval-Testing semantics by restricting the $[\![\cdot]\!]^{\text{MAY}}$, $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$, and $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ semantics to interval pomsets and interval pomset-divergences:

Definition 3.3.4 Let $\mathcal{P}$ be a set of pomsets, let $\mathcal{PF}$ be a set of pomset-failures, and let $\mathcal{PD}$ be a set of pomset-divergences. Then $intervals(\mathcal{P})$ is the set of interval pomsets $p \in \mathcal{P}$, $intervals(\mathcal{PF})$ is the set of $\langle p, F\rangle \in \mathcal{PF}$ such that p is an interval pomset, and $intervals(\mathcal{PD})$ is the set of interval pomset-divergences $\langle p, D\rangle \in PD$.

For any alphabet Act, let $intervals(\langle \mathcal{P}, Act\rangle) \stackrel{\text{def}}{=} \langle intervals(\mathcal{P}), Act\rangle$, and let $intervals(\langle \mathcal{PF}, \mathcal{PD}, Act\rangle) \stackrel{\text{def}}{=} \langle intervals(\mathcal{PF}), intervals(\mathcal{PD}), Act\rangle$.

Definition 3.3.5 For any WT Net $\langle N, Act\rangle$,

$$\begin{array}{rcl}
[\![N]\!]^{\text{MAY}}_{\text{intvl}} & \stackrel{\text{def}}{=} & intervals([\![N]\!]^{\text{MAY}}) \\
[\![N]\!]^{\text{MUST}}_{\text{intvl-}\gamma} & \stackrel{\text{def}}{=} & intervals([\![N]\!]^{\text{MUST}}_{\text{split-}\gamma}) \\
[\![N]\!]^{\text{TEST}}_{\text{intvl-}\gamma} & \stackrel{\text{def}}{=} & \langle [\![N]\!]^{\text{MAY}}_{\text{intvl}}, [\![N]\!]^{\text{MUST}}_{\text{intvl-}\gamma}\rangle
\end{array}$$

We have:

Theorem 3.3.6 The $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$, $[\![\cdot]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$, and $[\![\cdot]\!]^{\text{TEST}}_{\text{intvl-}\gamma}$ semantics are respectively adequate for MAY-equivalence, MUST-equivalence, and Testing-equivalence.

Proof. We first note that all linear orderings are interval orderings. The proof is then identical to that of Theorem 3.2.37. ∎

The following facts will be useful in proving the compositionality of the interval semantics:

Proposition 3.3.7 Let p, p_1, p_2, q be pomsets.

1. If p is an interval pomset and q is a prefix of p, then q is an interval pomset.

2. $a.p$ is an interval pomset iff p is an interval pomset.

3. $p[f]$ is an interval pomset iff p is an interval pomset.

4. $p_1; p_2$ is an interval pomset iff $p_1; \surd$ and p_2 are interval pomsets.

5. $q \in choice_{(a,a_L,a_R)}(p)$ is an interval pomset iff p is an interval pomset.

6. If q is an interval pomset and $q = p - a$ for some pomset p, then there is some interval pomset $p' \succeq p$ such that $q = p' - a$.

7. If q is an interval pomset and $q \in augment(p_1 \|_A p_2)$, then there are interval pomsets p'_1, p'_2 with $p'_1 \succeq p_1$, $p'_2 \succeq p_2$ such that $q \in augment(p'_1 \|_A p'_2)$.

8. If q is an interval pomset and $q \in augment(split_{(a,a_1,a_2)}(p))$ for some pomset p, then there is some interval pomset $p' \succeq p$ such that $q \in augment(split_{(a,a_1,a_2)}(p'))$.

9. If q is an interval pomset and $q \in augment(\textit{0-split}(p))$ for some pomset p, then there is some interval pomset $p' \succeq p$ such that $q \in augment(\textit{0-split}(p'))$.

Proof. We prove the case for hiding. Suppose q is an interval pomset and $q = p - a$ for some pomset p. Let p' have the same events with the same labels as p, and let $\leq_{p'}$ be a partial order that is maximal with respect to the following conditions: (i) $\leq_{p'}$ contains $\leq_p$, and (ii) $\leq_{p'}$ agrees with $\leq_p$ on all non-a-labeled events of p. Clearly, such a pomset p' exists since $\leq_p$ satisfies conditions (i) and (ii). Furthermore, it is easy to see by construction of p' that $p' - a = p - a = q$. Let $x <_{p'} y$ and $z <_{p'} w$, and suppose for the sake of contradiction that $x \not<_{p'} w$ and $z \not<_{p'} y$. Then by maximality of $\leq_{p'}$, there must be some non-a-labeled events x', y', z', w' such that $x' \leq_p x$, $y \leq_p y'$, $z' \leq_p z$, $w \leq_p w'$, and $x' \not<_p w'$ and $z' \not<_p y'$, contradicting the fact that $p - a$ is an interval ordering. Hence, either $x <_{p'} w$ or $z <_{p'} y$ after all, and so p' is an interval pomset.

The remaining cases are straightforward and are left to the reader. ∎

Similarly:

Proposition 3.3.8 Let $\langle p, D_p\rangle, \langle p_1, D_1\rangle, \langle p_2, D_2\rangle, \langle q, D_q\rangle$ be pomset-divergences.

1. If $\langle p, D_p\rangle$ is an interval pomset-divergence and $\langle q, D_q\rangle$ is a prefix of $\langle p, D_p\rangle$, then $\langle q, D_q\rangle$ is an interval pomset.

2. $a.\langle p, D_p\rangle$ is an interval pomset-divergence iff $\langle p, D_p\rangle$ is an interval pomset-divergence.

3. $\langle p, D_p\rangle[f]$ is an interval pomset-divergence iff $\langle p, D_p\rangle$ is an interval pomset-divergence.

4. $\langle p_1; p_2, \{d \cup \mathrm{Events}_{p_1} : d \in D_2\}\rangle$ is an interval pomset-divergence iff $p_1; \surd$ is an interval pomset and $\langle p_2, D_2\rangle$ is an interval pomset-divergence.

5. $\langle q, D_q\rangle \in \textit{0-1-choice}_{(a,a_L,a_R)}(\langle p, D_p\rangle)$ is an interval pomset-divergence iff $\langle p, D_p\rangle$ is an interval pomset-divergence.

6. If $\langle q, D_q\rangle$ is an interval pomset-divergence and $\langle q, D_q\rangle = \langle p, D_p\rangle - a$ for some pomset-divergence $\langle p, D_p\rangle$, then there is some interval pomset-divergence $\langle p', D_{p'}\rangle \in augment(extend_{Act}(\langle p, D_p\rangle))$ such that $\langle q, D_q\rangle = \langle p', D_{p'}\rangle - a$.

7. If $\langle q, D_q\rangle$ is an interval pomset-divergence and $\langle q, D_q\rangle \in augment(\langle p_1, D_1\rangle \|_A \langle p_2, D_2\rangle)$, then there are interval pomset-divergences $\langle p'_1, D'_1\rangle, \langle p'_2, D'_2\rangle$ with $\langle p'_1, D'_1\rangle \succeq \langle p_1, D_1\rangle$, $\langle p'_2, D'_2\rangle \succeq \langle p_2, D_2\rangle$ such that $\langle q, D_q\rangle \in augment(\langle p'_1, D'_1\rangle \|_A \langle p'_2, D'_2\rangle)$.

8. If $\langle q, D_q\rangle$ is an interval pomset-divergence and $\langle q, D_q\rangle \in augment(\textit{0-1-split}_{(a,a_+,a_-)}(\langle p, D_p\rangle))$ for some pomset-divergence $\langle p, D_p\rangle$, then there is some interval pomset-divergence $\langle p', D_{p'}\rangle \succeq \langle p, D_p\rangle$ such that $\langle q, D_q\rangle \in augment(\textit{0-1-split}_{(a,a_+,a_-)}(\langle p', D_{p'}\rangle))$.

Proof. We prove the case for hiding. Suppose $\langle q, D_q\rangle$ is an interval pomset-divergence and $\langle q, D_q\rangle = \langle p, D_p\rangle - a$ for some pomset-divergence $\langle p, D_p\rangle$. Let p' be defined as in the proof of Proposition 3.3.7 and let $D_{p'} = \{d'\}$, where $d' = down_{p'}(d) \cup (\text{Events}_{p'} - max(p'))$ for some $d \in D$. By the earlier proof and by construction of $D_{p'}$, clearly, $\langle p', D_{p'}\rangle$ is an interval pomset-divergence and $\langle p, D_p\rangle \sqsubseteq \langle p, D_{p'}\rangle \preceq \langle p', D_{p'}\rangle$. Since $\langle q, D_q\rangle$ is an interval pomset-divergence, $D_q = \{d_q\}$ for some $d_q \supseteq \text{Events}_q - max(q)$; it is then easy to see from the construction of $\langle p', D_{p'}\rangle$ that $\langle p', D_{p'}\rangle - a = \langle q, D_q\rangle$, proving this case.

The remaining cases are straightforward and are left to the reader. ∎

We will also use the following fact about interval pomset-divergences:

Lemma 3.3.9 Let $\langle p, D_p\rangle$ be a (possibly non-interval) pomset-divergence and let $\langle q, D_q\rangle$ be an interval pomset-divergence with $\langle p, D_p\rangle \sqsubseteq \langle q, D_q\rangle$. Then there is some interval pomset-divergence $\langle p', \{d'\}\rangle$ with $\langle p', \{d'\}\rangle \sqsubseteq \langle q, D_q\rangle$ such that p' is a prefix of p and $d' \supseteq d$ for some $d \in D_p$.

Proof. By the definition of interval pomset-divergences, $D_q = \{d_q\}$ for some $d_q \supseteq \text{Events}_q - max(q)$.

We first show that there is some $d \in D_p$ such that $\langle p, \{d\}\rangle \sqsubseteq \langle q, \{d_q\}\rangle$. The proof is by induction on $n = |\text{Events}_q - \text{Events}_p|$. The base case of $n = 0$ is obvious. For the other base case, let $n = 1$, and let $\{x\} = \text{Events}_q - \text{Events}_p$. Clearly, there is some $d \in D_p$ with $d \subseteq down_q(x)$; furthermore, $d \subseteq down_q(x) \subseteq \text{Events}_q - max(q) \subseteq d_q$, and so $\langle p, \{d\}\rangle \sqsubseteq \langle q, \{d_q\}\rangle$.

For the inductive step, suppose that $n > 1$. Let $\{x_1, \ldots, x_n\} = \text{Events}_q - \text{Events}_p$, and assume *wlog* that $x_n \in max(q)$. It is easy to see that $\langle q - x_n, \{d_q - x_n\}\rangle$ is an interval pomset-divergence and that $\langle p, D_p\rangle \sqsubseteq \langle q - x_n, \{d_q - x_n\}\rangle$; by the inductive hypothesis, there is some $d_1 \in D_p$ with $\langle p, \{d_1\}\rangle \sqsubseteq \langle q - x_n, \{d_q - x_n\}\rangle$. If $d_1 \subseteq down_q(x_n)$, then clearly $\langle p, \{d_1\}\rangle \sqsubseteq \langle q, \{d_q\}\rangle$. Otherwise, there is some $y \in d_1$ such that $y \not<_q x_n$; we recall that $y <_q x_i$ for all $1 \leq i < n$ since $\langle p, \{d_1\}\rangle \sqsubseteq \langle q - x_n, \{d_q - x_n\}\rangle$. Now consider any $z <_q x_n$; since q is an interval ordering, it follows that $z <_q x_i$ for all $1 \leq i < n$. Thus, $down_q(x_n) \subseteq down_q(x_i)$ for all $1 \leq i < n$. Since $\langle p, D_p\rangle \sqsubseteq \langle q, \{d_q\}\rangle$, there is some $d \in D_p$ with $d \subseteq down_q(x_n) \subseteq down_q(x_i)$ for all $1 \leq i < n$. Furthermore, $d \subseteq down_q(x_n) \subseteq \text{Events}_q - max(q) \subseteq d_q$, and so $\langle p, \{d\}\rangle \sqsubseteq \langle q, D_q\rangle$ as desired.

Now let p' be the restriction of p to the set $\{x \in \text{Events}_p : d \not\subseteq down_p(x)\}$, which is easily seen to be a downward-closed subset of Events_p. Furthermore, let $d' = d \cup (\text{Events}_{p'} - max(p'))$, which is easily seen to be a downward-closed subset of $\text{Events}_{p'}$. Since $d \subseteq d_q$ and $\text{Events}_{p'} - max(p') \subseteq \text{Events}_q - max(q) \subseteq d_q$, it is easy to see that $d' \subseteq d_q$. Let $x \in d'$ and let $z \in \text{Events}_q - \text{Events}_{p'}$. For one case, let $x \in d$; then it is easy to see that $x <_q z$. For the other case, let $x \in d' - d$; then there is some $x' \in p'$ such that $x <_{p'} x'$ and $d \not\subseteq down_p(x')$; so there is some $y \in d$ with $y \not<_p x'$. It is easy to see that $y <_q z$, and since q is an interval pomset, it follows that $x <_q z$, proving that $\langle p', \{d'\}\rangle \sqsubseteq \langle q, \{d_q\}\rangle$. Clearly, prefixes of interval pomsets are also interval pomsets, from which it follows easily from the construction of d' that $\langle p', \{d'\}\rangle$ is an interval pomset-divergence, proving the lemma. ∎

Theorem 3.3.10 The $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$, $[\![\cdot]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$, and $[\![\cdot]\!]^{\text{TEST}}_{\text{intvl-}\gamma}$ semantics are compositional for split refinements, choice refinements, alphabet expansion and shrinking, and all of our CCS/CSP operators.

Proof. Let Act be a finite alphabet containing $\surd$, let $a, a_L, a_R, a_+, a_- \in Act$, let $A \subseteq Act$, let f be a function from Act to Act such that for all $\alpha \in Act$, $f(\alpha) = \surd$ iff $\alpha = \surd$, and let Act' be a finite set of labels containing $\surd$. Furthermore, let $\langle N, Act\rangle, \langle N_1, Act\rangle, \langle N_2, Act\rangle$ be WT Nets.

The following identities, where the operations on the right-hand side of the equations are those defined in Definition 3.2.28, follow immediately from the augmentation-closure of the $[\![\cdot]\!]^{\mathrm{MAY}}$ semantics, Proposition 3.3.7, and Theorem 3.2.30. The details are straightforward and are left to the reader.

$$[\![\langle N, Act\rangle \ \mathit{grow}\ Act']\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} \ \mathit{grow}\ Act'$$

$$[\![\langle N, Act\rangle \ \mathit{shrink}\ Act']\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} \ \mathit{shrink}\ Act'$$

$$[\![a.\langle N, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = a.[\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}$$

$$[\![\tau.\langle N, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}$$

$$[\![\langle N, Act\rangle \backslash a]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} \backslash a$$

$$[\![\langle N, Act\rangle[f]]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}[f]$$

$$[\![\langle N, Act\rangle - a]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} - a$$

$$[\![\langle N_1, Act\rangle ; \langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = [\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} ; [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}$$

$$[\![\langle N_1, Act\rangle \oplus \langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = [\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} \oplus [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}$$

$$[\![\langle N_1, Act\rangle +_M \langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = [\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} +_M [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}$$

$$[\![\langle N_1, Act\rangle \mid\mid \langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = \mathit{intervals}([\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} \mid\mid [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}})$$

$$[\![\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = \mathit{intervals}([\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} \|_{A \cup \{\surd\}} [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}})$$

$$[\![\langle N_1, Act\rangle \mid \langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = \mathit{intervals}([\![\langle N_1, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} \mid [\![\langle N_2, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}})$$

$$[\![\mathit{choice}_{(a,a_L,a_R)}(\langle N, Act\rangle)]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = \mathit{choice}_{(a,a_L,a_R)}([\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}})$$

$$[\![\mathit{split}_{(a,a_1,a_2)}(\langle N, Act\rangle)]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = \mathit{intervals}(\mathit{split}_{(a,a_1,a_2)}([\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}))$$

The following identities follow immediately from the augmentation-closure of the $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$ semantics, Proposition 3.2.14, Proposition 3.2.18, Proposition 3.3.8, Proposition 3.3.9, and Theorem 3.2.46. The operations on the right-hand side of the equations are those defined in Definition 3.2.45. We prove the case for CSP-style parallel composition; the remaining equalities are left to the reader.

$$[\![\langle N, Act\rangle \ \mathit{grow}\ Act']\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}\ \textbf{grow}\ Act')$$

$$[\![\langle N, Act\rangle \ \mathit{shrink}\ Act']\!]^{\text{MUST}}_{\text{intvl-}\gamma} = [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}\ \textbf{shrink}\ Act'$$

$$[\![a.\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}(a\ \textbf{pref}\ [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma})$$

$$[\![\tau.\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}(\tau\ \textbf{pref}\ [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma})$$

$$[\![\langle N, Act\rangle\backslash a]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}\ \textbf{rst}\ a)$$

$$[\![\langle N, Act\rangle[f]]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}\ \textbf{rename with}\ f)$$

$$[\![\langle N, Act\rangle - a]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}\ \textbf{hide}\ a)$$

$$[\![\langle N_1, Act\rangle;\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}([\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}\ \textbf{seq}\ [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma})$$

$$[\![\langle N_1, Act\rangle +_M \langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} +_M [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$$

$$[\![\langle N_1, Act\rangle \oplus \langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = [\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}\ \textbf{internal choice}\ [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$$

$$[\![\langle N_1, Act\rangle \mid\mid \langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}([\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} \mid\mid [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma})$$

$$[\![\langle N_1, Act\rangle||_A\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}([\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}\ \textbf{CSP-parallel}_{A\cup\{\gamma,\surd\}}\ [\![\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma})$$

$$[\![\langle N_1, Act\rangle \mid \langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}([\![\langle N_1, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}\ \textbf{CCS-parallel}\ \langle N_2, Act\rangle)$$

$$[\![\mathit{choice}_{(a,a_L,a_R)}(\langle N, Act\rangle)]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}(\textbf{choice}_{(a,a_L,a_R)}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}))$$

$$[\![\mathit{split}_{(a,a_+,a_-)}(\langle N, Act\rangle)]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \mathit{intervals}(\textbf{split}_{(a,a_+,a_-)}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}))$$

We prove the equality for CSP-style parallel composition. It is easy to see that one direction follows easily from Theorem 3.2.46 and the monotonicity of all the operations. To prove the other direction, we first recall from the proof of Theorem 3.2.46 that:

$$\begin{aligned}
&\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle)\ \textit{grow}\ \{\gamma\})) = \\
&\quad \textit{0-split}(\{\langle p, F\rangle : \langle p_1, F_1\rangle \in \textit{1-2-respect}(\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N_1, Act\rangle)\ \textit{grow}\ \{\gamma\}))), \\
&\qquad \langle p_2, F_2\rangle \in \textit{1-2-respect}(\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N_2, Act\rangle)\ \textit{grow}\ \{\gamma\}))), \\
&\qquad \text{and } p \in p_1 \|_{A'} p_2,\ F - A' \subseteq F_1 \cap F_2 \text{ and } F \cap A' \subseteq F_1 \cup F_2, \\
&\qquad \text{where } A' = \{a_i : a \in A \text{ and } 0 \le i \le 2\} \cup \{\surd\})
\end{aligned}$$

$$\begin{aligned}
&\textit{pomset-divergences}(\gamma +_M (\textit{dupl-split}(\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle)\ \textit{grow}\ \{\gamma\})) = \\
&\quad \textit{0-split}(\bigcup\{\langle p_1, D_1\rangle \|_{A'} \langle p_2, D_2\rangle : \\
&\qquad \langle p_1, D_1\rangle \in \textit{1-2-respect}(\textit{pomset-divergences}(\gamma +_M (\textit{dupl-split}(\langle N_1, Act\rangle)\ \textit{grow}\ \{\gamma\}))) \\
&\qquad\qquad \cup\ \textit{1-2-respect}(\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N_1, Act\rangle)\ \textit{grow}\ \{\gamma\}))), \\
&\qquad \langle p_2, D_2\rangle \in \textit{1-2-respect}(\textit{pomset-divergences}(\gamma +_M (\textit{dupl-split}(\langle N_2, Act\rangle)\ \textit{grow}\ \{\gamma\}))) \\
&\qquad\qquad \cup\ \textit{1-2-respect}(\textit{pomset-failures}(\gamma +_M (\textit{dupl-split}(\langle N_2, Act\rangle)\ \textit{grow}\ \{\gamma\}))), \\
&\qquad D_1 \text{ and } D_2 \text{ are (possibly empty) downward-closed subsets of} \\
&\qquad \text{Events}_{p_1} \text{ and Events}_{p_2}, \text{ respectively, and } D_1 \cup D_2 \neq \emptyset, \\
&\qquad \text{and } A' = \{a_i : a \in A \text{ and } 0 \le i \le 2\} \cup \{\surd\})
\end{aligned}$$

Let $\langle r, D_r\rangle \in \textit{snd}([\![\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma})$; then

$$\langle r, D_r\rangle \in \textit{intervals}(\textit{augment}(\textit{extend}_{Act}(\langle p, D_p\rangle)))$$

for some pomset-divergence $\langle p, D_p\rangle$ of $\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle$. By Lemma 3.2.16 and Lemma 3.3.9, there is some *interval* pomset-divergence $\langle q, \{d'\}\rangle$ such that $\langle r, D_r\rangle \in \textit{augment}(\textit{extend}_{Act}(\langle q, \{d'\}\rangle))$, q is an augmentation of a prefix of p and $d' \supseteq d$ for some $d \in D_p$. By Proposition 3.2.14,

$$\langle q, \{d'\}\rangle \in \textit{extend}_{Act}(\textit{augment}(\textit{pomset-divergences}(\langle N_1, Act\rangle \|_A \langle N_2, Act\rangle))).$$

It then follows easily from the highlighted fact above and the definitions of *augment* and *0-split* that $\langle q, \{d'\}\rangle \in \textit{extend}_{Act}(\textit{augment}(\textit{0-split}(\langle p', D_{p'}\rangle)))$ for some $\langle p', D_{p'}\rangle \in \langle p_1, D_1\rangle \|_{A'} \langle p_2, D_2\rangle)$, where $\langle p_1, D_1\rangle, \langle p_2, D_2\rangle$ are appropriate pomset-divergences or pomset-failures.

It follows from Lemma 3.3.9 and Proposition 3.3.8 that there are some *interval* pomset-divergences $\langle q', D_{q'}\rangle \succeq \langle p', D_{p'}\rangle$, $\langle q_1, D_{q_1}\rangle \succeq \langle p_1, D_1\rangle$, $\langle q_2, D_{q_2}\rangle \succeq \langle p_2, D_2\rangle$ such that $\langle q, \{d'\}\rangle \in \textit{extend}_{Act}(\textit{augment}(\textit{0-split}(\langle q', D_{q'}\rangle)))$ and $\langle q', D_{q'}\rangle \in \textit{augment}(\langle q_1, D_{q_1}\rangle \|_{A'} \langle q_2, D_{q_2}\rangle)$. From the definition of *0-split* and *augment* and Lemma 3.2.16, it is easy to see that

$$\langle q, \{d'\}\rangle \in \textit{augment}(\textit{extend}_{Act}(\textit{0-split}(\langle q_1, D_{q_1}\rangle \|_{A'} \langle q_2, D_{q_2}\rangle)))$$

The desired equality then follows easily. The proof for pomset-failures is similar, except that it uses Proposition 3.3.7 instead of Proposition 3.3.8.

Proposition 2.2.18 and the equalities for prefixing and CCS choice together imply the compositionality of internal choice. Proposition 2.2.19 and the equalities for alphabet expansion and shrinking, CSP-style parallel composition, choice refinements, and hiding together imply the compositionality of CCS-style parallel composition. ■

Theorem 3.3.11 The $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$, $[\![\cdot]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$, and $[\![\cdot]\!]^{\text{TEST}}_{\text{intvl-}\gamma}$ semantics are respectively fully abstract for MAY-equivalence, MUST-equivalence, and Testing-equivalence with respect to alphabet expansion, split refinements, choice refinements, and CCS choice. Furthermore, only split and choice refinements are necessary for $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$.

Proof. By Theorem 3.3.6 and Theorem 3.3.10, it remains to prove distinguishability. Let $\langle N_1, Act_1\rangle, \langle N_2, Act_2\rangle$ be WT Nets. For one case, let $\langle PT_1, Act_1\rangle = [\![\langle N_1, Act_1\rangle]\!]^{\text{MAY}}_{\text{intvl}}$, let $\langle PT_2, Act_2\rangle = [\![\langle N_2, Act_2\rangle]\!]^{\text{MAY}}_{\text{intvl}}$, and suppose that $\langle PT_1, Act_1\rangle \neq \langle PT_2, Act_2\rangle$. If $Act_1 \neq Act_2$, it is easy to see that the nets are MAY-inequivalent. Otherwise, $Act_1 = Act_2 = Act$ and $PT_1 \neq PT_2$; we assume *wlog* that there is some interval pomset $p \in PT_1 - PT_2$. Let $n = |\text{Events}_p|$, and let $Act' = \{a^j, a^j_1, a^j_2 \colon a \in Act \text{ and } 1 \leq j \leq n\}$ be distinct symbols not in Act. Finally, let $C[\cdot]$ be the following net context:

$$C[\cdot] = \sigma(\delta(\cdot\ grow\ Act')),$$

where δ is the sequence of choice refinements $\langle choice_{(a,a^1,\ldots,a^n)} \colon a \in Act\rangle$ (which can be programmed by repeated use of binary choice refinements), and σ is the sequence of split refinements $\langle split_{(a^1,a^1_1,a^1_2)} \ldots split_{(a^n,a^n_1,a^n_2)} \colon a \in Act\rangle$.

We will perform corresponding split and choice refinements on pomsets. Using Definition 3.2.25, we can overload notation and let δ also represent the obvious sequence of choice refinements on pomsets. For concreteness, we will "fully split" all events in p, and so, using Definition 3.2.26, we let $\sigma' = \langle split_{(a^1,a^1_1,a^1_2,\emptyset)} \ldots split_{(a^n,a^n_1,a^n_2,\emptyset)} \colon a \in Act\rangle$.

Since $n = \text{Events}_p$, it is easy to see that there is some pomset $q \in \sigma'(\delta(p))$ such that q is an augmentation of a pomset-trace of $C[N_1]$ and all labels in q are distinct. Furthermore, since we implicitly equate isomorphic pomsets, it is easy to see that we can assume *wlog* that $\text{Events}_q = \{(y,1),(y,2) \mid y \in \text{Events}_p\}$, $l_q((y,i)) = (l_p(y))^k_i$ for some $1 \leq k \leq n$, and $(y,i) <_q (y',j)$ iff either $y <_p y'$ or ($y =_p y'$ and $i < j$). Clearly, there is a unique (injective) mapping I from events x of p to labels a^i, where $I(x) = a^i$ iff $l_p(x) = a$ and $l_q((x,1)) = a^i_1$. For any event x of p, we can regard the (unique) $I(x)_1$-labeled and $I(x)_2$-labeled events of q as respectively representing the "beginning" and "end" of the interval corresponding to x. Now, since p is an interval pomset, it follows by Lemma 3.3.2 that there is a linearization v of q such that (the unique) a^i_2-labeled event precedes (the unique) b^j_1-labeled event in v iff $I^{-1}(a^i) <_p I^{-1}(b^j)$.

Clearly, $v \in traces(C[N_1])$. If $v \in traces(C[N_2])$, there would be some pomset-trace p' of N and some $q' \in \sigma'(\delta(p'))$ such that v is a linearization of q'; thus, all events in q' must have distinct labels. Clearly, there is a unique (injective) mapping I' from events x of p' to labels a^i, where $I'(x) = a^i$ iff $l_{p'}(x) = a$ and $l_{q'}((x,1)) = a^i_1$. It is then easy to see that $I^{-1} \circ I'$ is a label-preserving order-augmenting bijection from p' to p. But by definition of $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$, this would imply that $p \in PT_2$, a contradiction. Thus, $v \in traces(C[N_1]) - traces(C[N_2])$ after all, and so by Proposition 3.1.4, $C[N_1]$ and $C[N_2]$ are MAY-inequivalent, proving this case.

To prove that $[\![\cdot]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$ is fully abstract, let $\langle PF_1, PD_1, Act'_1\rangle = [\![\langle N_1, Act_1\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$, let $\langle PF_2, PD_2, Act'_2\rangle = [\![\langle N_2, Act_2\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$, and suppose that $\langle PF_1, PD_1, Act'_1\rangle \neq \langle PF_2, PD_2, Act'_2\rangle$. If $Act'_1 \neq Act'_2$, it is easy to see that $Act_1 \neq Act_2$ and hence the nets are MUST-inequivalent.

For the next case, suppose $PD_1 \neq PD_2$; we assume *wlog* that there is some interval pomset-divergence $\langle p, \{d_p\}\rangle \in PD_1 - PD_2$. Using Proposition 3.2.15, Proposition 3.2.17, and Lemma 3.3.9, we can assume *wlog* that there is some $d \subseteq d_p$ such that $\langle p, \{d\}\rangle$ is an augmentation of a pomset-divergence $\langle r, \{d_r\}\rangle$ of $\gamma +_M (dupl\text{-}split(\langle N_1, Act\rangle)\ grow\ \{\gamma\})$; the details are straightforward and are left to the reader. From the definition of $\gamma +_M$ and the definition of WT Nets, r, and hence p, does not have any γ-labeled or $\surd$-labeled events. By Lemma 3.2.41, we can assume wlog that r, and hence p, does not contain any α_2-labeled events for any $\alpha \in Act$.

By Proposition 3.2.38, it is easy to see that for all $\alpha \in Act$, every α_1-labeled event is maximal in r; thus, we can assume *wlog* that for all $\alpha \in Act$, every α_1-labeled event is also maximal in p. Again using Proposition 3.2.38, it follows that d_r, and hence d, contains only α_0-events; thus, we can assume *wlog* that d_p contains *all and only* the α_0-events of p. Let Act', σ, and δ be as in the previous case, and let

$$C[\cdot] = \gamma +_M \sigma(\delta(\cdot\ \mathit{grow}\ Act' \cup \{\gamma\})).$$

Furthermore, let $n = |\mathrm{Events}_p|$, and let δ' be the sequence $\langle \mathit{choice}_{(a_0, a^1, \ldots a^n)} : a \in Act \rangle$ followed by the sequence $\langle \mathit{choice}_{(a_1, a_1^1, \ldots a_1^n)} : a \in Act \rangle$. We let σ' be as in the previous case.

It is easy to see that there is some pomset-divergence $\langle q, \{d_q\}\rangle \in \sigma'(\delta'(\langle p, \{d_p\}\rangle))$ such that all labels in q are distinct, d_q contains all and only the α_2-labeled events of q for all $\alpha \in Act$, and for some $d' \subseteq d_q$, $\langle q, \{d'\}\rangle$ is an augmentation of a pomset-divergence of $C[N_1]$. Furthermore, in q all α_0-labeled events of p have been "fully split," while all α_1-labeled events of p have been relabeled. Again, since we implicitly equate isomorphic pomsets, it is easy to see that we can assume *wlog* that

- $\mathrm{Events}_q = \{(y,1),(y,2) \mid y \in \mathrm{Events}_p \text{ and } l_p(y) = a_0 \text{ for some } a \in Act\} \cup$
  $\{(y,1) \mid y \in \mathrm{Events}_p \text{ and } l_p(y) = a_1 \text{ for some } a \in Act\}$
- $l_q((y,i)) = (l_p(y))_i^k$ for some $1 \le k \le n$, and
- $(y,i) <_q (y',j)$ iff either $y <_p y'$ or ($y =_p y'$ and $i < j$).

Clearly, there is a unique mapping I from events x of p to labels a^i, where $I(x) = a^i$ iff $l_p(x) = a$ and $l_q((x,1)) = a_1^i$. Using similar reasoning as in the previous case, it follows by Lemma 3.3.2 that since p is an interval pomset, there is a linearization v of q such that (the unique) a_2^i-labeled event precedes (the unique) b_1^j-labeled event in v iff $I^{-1}(a^i) <_p I^{-1}(b^j)$. Clearly, $v \in \mathcal{D}(C[N_1])$. If $v \in \mathcal{D}(C[N_2])$, let v' be the minimal prefix of v with $v' \in \mathcal{D}(C[N_2])$. Then there must be some pomset-divergence $\langle p', \{d_{p'}\}\rangle$ of $\gamma +_M (\mathit{dupl\text{-}split}(\langle N_2, Act\rangle)\ \mathit{grow}\ \{\gamma\})$ such that p' contains no α_2-labeled events and some $\langle q', \{d_{q'}\}\rangle \in \sigma'(\delta'(p'))$ such that v' is a linearization of q'. By the same reasoning as before, we can assume without loss of generality that all α_1-labeled events in p' are maximal, and $d_{p'}$ contains exactly the a_0 labeled events of p for all $a \in Act$. Clearly, there is a unique mapping I' from events x of p' to labels a^i, where $I'(x) = a^i$ iff $l_{p'}(x) = a$ and $l_{q'}((x,1)) = a_1^i$. It is then easy to see that $I^{-1} \circ I'$ is a label-preserving order-augmenting bijection from p' to a downward-closed subset of p. Furthermore, for all $x \in p - I^{-1}(I'(p'))$ and all $y \in d_p$, it is easy to see that the $I'(y)_2$-labeled event exists and is in v', and the $l_q(x,1)$-labeled event is in $v - v'$. Hence, $I^{-1}(I'(y)) <_p x$. Furthermore, since d_p and $d_{p'}$ contain exactly the the α_0-labeled events of p and p', respectively, it is easy to see that $I^{-1}(I'(d_{p'})) \subseteq d_p$, and so $I^{-1}(I'(\langle p', \{d_{p'}\}\rangle)) \sqsubseteq \langle d, \{d_p\}\rangle$. But by definition of $[\![\cdot]\!]^{\mathrm{MUST}}_{\mathrm{intvl}\text{-}\gamma}$, this would imply that $\langle p, \{d_p\}\rangle \in PD_2$, a contradiction. Thus, $v \notin \mathcal{D}(C[N_2])$ after all, and so by Proposition 3.1.4, $C[N_1]$ and $C[N_2]$ are MUST-inequivalent, proving this case.

For the last case, suppose that $PD_1 = PD_2$ but $PF_1 \neq PF_2$; we assume *wlog* that there is some interval pomset-failure $\langle p, F_p\rangle \in PF_1 - PF_2$ such that $\langle p, \{\mathrm{Events}_p\}\rangle \notin PD_1 \cup PD_2$. Thus, $\langle p, F_p\rangle$ is an augmentation of a pomset-failure $\langle r, F_p\rangle$ of $\gamma +_M (\mathit{dupl\text{-}split}(\langle N_1, Act\rangle)\ \mathit{grow}\ \{\gamma\})$. It is easy to see from the definition of $[\![\cdot]\!]^{\mathrm{MUST}}_{\mathrm{intvl}\text{-}\gamma}$ that r, and hence p, cannot contain any γ-labeled event, while it follows from the definition of WT Nets that r, and hence p, can contain at most one $\surd$-labeled event. Furthermore, from Lemma 3.2.41, we can assume without loss of generality that r, and hence p, does not contain any α_2 labeled events, for any $\alpha \in Act$. Finally, using Proposition 3.2.38, it is easy to see that for all $\alpha \in Act$, any α_1-labeled event is maximal in r; thus, we can assume *wlog* that for all $\alpha \in Act$, all α_1-labeled events are also maximal in p. Let Act', σ, δ, and $C[\cdot]$ be as in the previous case. Furthermore, let $n = |\mathrm{Events}_p|$, and let δ' be the sequence $\langle choice_{(a_0,a^1,\ldots a^n)} : a \in Act - \{\surd\}\rangle$ followed by the sequence $\langle choice_{(a_1,a_1^1,\ldots a_1^n)} : a \in Act - \{\surd\}\rangle$ followed by $choice_{(\surd,\surd^1,\surd^1)}$. We let σ' be as in the previous cases. It is easy to see that there is some pomset $q \in \sigma'(\delta'(p))$ such that all labels in q are distinct and $\langle q, F_q\rangle$ is an augmentation of a pomset-failure of $C[N_1]$, where $F_q = \{a_1^1 : a_0 \in F_p\} \cup (\{\gamma, \surd\} \cap F_p)$. Furthermore, in q all α_0-labeled events of p have been "fully split," while all α_1-labeled events of p have been "half-split." Again, since we implicitly equate isomorphic pomsets, we can assume that q has the same form as in the previous case.

Clearly, there is a unique mapping I from events x of p to labels a^i, where $I(x) = a^i$ iff $l_p(x) = a$ and $l_q((x,1)) = a_1^i$. Using similar reasoning as in the previous case, it follows by Lemma 3.3.2 that since p is an interval pomset, there is a linearization v of q such that (the unique) a_2^i-labeled event precedes (the unique) b_1^j-labeled event in v iff $I^{-1}(a^i) <_p I^{-1}(b^j)$. Clearly, $\langle v, F_q\rangle \in \mathcal{F}(C[N_1])$. If $v \in \mathcal{D}(C[N_2])$, then it is easy to show by the same reasoning as the previous case that $\langle p, \{\mathrm{Events}_p\}\rangle \in PD_2$, a contradiction. Thus, if $\langle v, F_q\rangle \in \mathcal{F}(C[N_2])$ there would be some pomset-failure $\langle p', F_{p'}\rangle$ of $\gamma +_M (\textit{dupl-split}(\langle N_2, Act\rangle)\ \textit{grow}\ \{\gamma\})$; and some $q' \in \sigma'(\delta'(p'))$ such that v' is a linearization of q' and $F_q \subseteq \{a_1^1 : a_0 \in F_{p'}\} \cup (\{\gamma, \surd\} \cap F_{p'})$. Clearly, there is a unique mapping I' from events x of p' to labels a^i, where $I'(x) = a^i$ iff $l_{p'}(x) = a$ and $l_{q'}((x,1)) = a_1^i$. It is then easy to see that $I^{-1} \circ I'$ is a label-preserving order-augmenting bijection from p' to p. Furthermore, it is easy to see that $F_p \subseteq F_{p'}$. But by definition of pomset-failures and $[\![\cdot]\!]^{\mathrm{MUST}}_{\text{intvl-}\gamma}$, this would imply that $\langle p, F_p\rangle \in PF_2$, a contradiction. Thus, $\langle v, F_q\rangle \notin \mathcal{F}(C[N_2])$ after all, and so by Proposition 3.1.4, $C[N_1]$ and $C[N_2]$ are MUST-inequivalent, proving this case and the theorem. ∎

We now observe that the $[\![\cdot]\!]^{\mathrm{MAY}}$,$[\![\cdot]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$, and $[\![\cdot]\!]^{\mathrm{TEST}}_{\text{split-}\gamma}$ semantics make strictly more distinctions than the $[\![\cdot]\!]^{\mathrm{MAY}}_{\text{intvl}}$,$[\![\cdot]\!]^{\mathrm{MUST}}_{\text{intvl-}\gamma}$, and $[\![\cdot]\!]^{\mathrm{TEST}}_{\text{intvl-}\gamma}$ semantics, respectively, and hence are not fully abstract:

Theorem 3.3.12 The $[\![\cdot]\!]^{\mathrm{MAY}}$, $[\![\cdot]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$, and $[\![\cdot]\!]^{\mathrm{TEST}}_{\text{split-}\gamma}$ semantics are respectively *not* fully abstract for MAY-equivalence, MUST-equivalence, and Testing-equivalence with respect to the WT Net operators.

Proof. Let N_1 and N_2 be the nets pictured in Figure 3-8, let $Act = \{a, b\}$, and let p be the pomset pictured in Figure 3-8. It is straightforward to show that $\langle N_1, Act\rangle +_M \langle N_2, Act\rangle$ and $\langle N_2, Act\rangle$ have equivalent $[\![\cdot]\!]^{\mathrm{TEST}}_{\text{intvl-}\gamma}$ meanings. However, they have different $[\![\cdot]\!]^{\mathrm{MAY}}$ meanings, since p is a non-interval pomset-trace of $\langle N_1, Act\rangle$ but not of $\langle N_2, Act\rangle$. Furthermore, they have different $[\![\cdot]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$ meanings, since $\langle p[f], \emptyset\rangle$, where $f(a) = a_0$ and $f(b) = b_0$, is a non-interval pomset-failure of the $\gamma +_M$ *dupl-split* version of $\langle N_1, Act\rangle +_M \langle N_2, Act\rangle$ but not of the $\gamma +_M$ *dupl-split* version of $\langle N_2, Act\rangle$. Thus, it is an immediate consequence of Theorem 3.3.11 and the definitions of the semantics that $[\![\cdot]\!]^{\mathrm{MAY}}$, $[\![\cdot]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$, and $[\![\cdot]\!]^{\mathrm{TEST}}_{\text{split-}\gamma}$ cannot be fully abstract. ∎

In addition to process equivalence under experiments, Hennessy [19] presents a natural form of MAY-, MUST-, or Testing-*approximation*, in which a process, p, is said to MAY-approximate

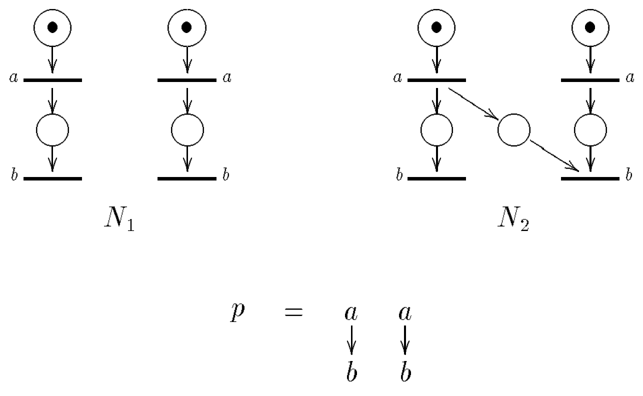


Figure 3-8: Interval Example

(MUST-approximate) a process, q, iff q may (must) pass every experiment that p may (must) pass, but not necessarily the converse. Our $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$, $[\![\cdot]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$, and $[\![\cdot]\!]^{\text{TEST}}_{\text{intvl-}\gamma}$ semantics are, in fact, also fully abstract with respect to MAY-, MUST-, or Testing-*approximation*, where the $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$ semantics is ordered by set-theoretic containment of the pomset-traces, and the $[\![\cdot]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$ semantics is ordered by component-wise reverse containment of the pomset-failures and pomset-divergences. These orderings will be presented in detail in Chapter 4. The proofs of full abstraction for process approximation are identical to the proof of 3.3.11.

Chapter 4

The Semantic Domains

In this chapter, we show that all the semantics presented in Chapter 3 map WT Nets to elements of complete partial orders (cpo's) and and that all our process operations correspond to continuous functions on these cpo's. In order to prove these properties, we give an abstract characterization of each of our spaces of process meanings. These results together provide a semantic foundation for inductive (fixed-point) reasoning about recursively-defined processes in the standard manner (*cf.* [18, 19]).

We also prove in this chapter that all our semantic domains are algebraic cpo's, in the sense that all elements are fully determined by the compact (finitely-specified) elements that approximate them. Furthermore, all compact elements of these cpo's are definable as the meanings of WT Nets. These results, although technically rather hard, are important because they guarantee that our full abstraction results from Chapter 3 will continue to hold for recursively-defined processes.

4.1 Standard Definitions

We begin with some standard definitions about algebraic complete partial orders and continuous functions, *cf.*, [18, 19].

Definition 4.1.1 A *partial order* is a pair $\langle \mathcal{D}, \sqsubseteq_{\mathcal{D}} \rangle$, where $\mathcal{D}$ is a set and $\sqsubseteq_{\mathcal{D}}$ is a binary relation on $\mathcal{D}$ that is reflexive, anti-symmetric, and transitive.

A element $x \in \mathcal{D}$ is *the least element of* $\mathcal{D}$ iff $x \sqsubseteq_{\mathcal{D}} y$ for every $y \in \mathcal{D}$. Let A be a subset of $\mathcal{D}$ and x an element of A. Then x is an *upper bound* of A iff $y \sqsubseteq_{\mathcal{D}} x$ for every $y \in A$. We say that x is a *least upper bound* of A iff, in addition, $x \sqsubseteq_{\mathcal{D}} z$ for every upper bound z of A. It follows from the anti-symmetry of $\sqsubseteq_{\mathcal{D}}$ that least upper bounds, if they exist, are unique. We use $\bigsqcup_{\mathcal{D}} A$ to denote the least upper bound of A, when it exists.

A is a *directed subset* of $\mathcal{D}$ iff it is non-empty and for all pairs of elements $x_1, x_2 \in A$, there is some $x_3 \in A$ such that x_3 is an upper bound of the set $\{x_1, x_2\}$.

The partial order $\langle \mathcal{D}, \sqsubseteq_{\mathcal{D}} \rangle$ is a *complete partial order (cpo)* iff it has a least element and every directed subset of $\mathcal{D}$ has a least upper bound.

An element $x \in \mathcal{D}$ is *compact* iff for every directed set $A \subseteq \mathcal{D}$ such that $x \sqsubseteq_{\mathcal{D}} \bigsqcup A$, there is some $y \in A$ with $x \sqsubseteq_{\mathcal{D}} y$. A cpo $\langle \mathcal{D}, \sqsubseteq_{\mathcal{D}} \rangle$ is *algebraic* iff for every element $z \in \mathcal{D}$, the set $M_z = \{x \in \mathcal{D} : x \text{ is compact and } x \sqsubseteq_{\mathcal{D}} z\}$ is directed and $z = \bigsqcup M_z$.

Definition 4.1.2 Let $\langle \mathcal{D}, \sqsubseteq_{\mathcal{D}} \rangle$ and $\langle \mathcal{E}, \sqsubseteq_{\mathcal{E}} \rangle$ be cpo's and let f be a function from $\mathcal{D}$ to $\mathcal{E}$. Then f is *continuous* iff for every directed subset $A \subseteq \mathcal{D}$, $f(A)$ is a directed subset of $\mathcal{E}$ and $f(\bigsqcup_{\mathcal{D}} A) = \bigsqcup_{\mathcal{E}} f(A)$.

Definition 4.1.3 Let $\langle \mathcal{D}, \sqsubseteq_{\mathcal{D}} \rangle$ be a cpo and let $f : \mathcal{D} \rightarrow \mathcal{D}$ be a function. An element $x \in \mathcal{D}$ is called a *fixed point* of f iff $x = f(x)$. It is called the *least fixed point* if, in addition, $x \sqsubseteq_{\mathcal{D}} y$ for every fixed point y of f.

The following well-known theorem ensures that complete partial orders and continuous functions support fixed point reasoning about recursively-defined processes, *cf.*, [18, 19] for the proof.

Theorem 4.1.4 (standard) Let $\langle \mathcal{D}, \sqsubseteq_{\mathcal{D}} \rangle$ be a cpo and let $f : \mathcal{D} \rightarrow \mathcal{D}$ be a continuous function. Then f has a least fixed point (in $\mathcal{D}$).

4.2 The Unsplit Semantics

This section gives an abstract characterization of the $[\![\cdot]\!]^{\mathrm{MAY}}$, $[\![\cdot]\!]^{\mathrm{MUST}}$, and $[\![\cdot]\!]^{\mathrm{TEST}}$ semantics on WT Nets, shows that they form algebraic cpo's, and proves that all the corresponding process operations from Chapter 3 are continuous functions on these cpo's.

The following proposition will be useful in proving these properties of our semantic domains:

Proposition 4.2.1 Let Act be a finite set of labels, let PT be a prefix-closed set of pomset-traces over Act, let PF be a prefix-closed set of pomset-failures over Act, and let PD be a prefix-closed set of pomset-divergences over Act. Then

- $augment(extend_{Act}(PD))$ is prefix-closed and extension-closed over Act.
- $augment(PT)$ is prefix-closed and $augment(PF)$ is prefix-closed.
- If $PF \supseteq \{\langle p, \emptyset \rangle : \langle p, D_p \rangle \in PD$ for some $D_p\}$, then $augment(PF) \cup implied\text{-}failures_{Act}(augment(extend_{Act}(PD)))$ is prefix-closed.

Proof. The extension-closure of $augment(extend_{Act}(PD))$ is a simple consequence of Proposition 3.2.15 and the prefix-closure of PD. To show that $augment(extend_{Act}(PD))$ is prefix-closed, suppose that $\langle p, D_p \rangle \sqsubseteq \langle q, D_q \rangle \preceq \langle r, D_r \rangle$, $\langle p, D_p \rangle \in PD$, and $\langle r', D_{r'} \rangle$ is a prefix of $\langle r, D_r \rangle$. By Proposition 3.2.17, there is some prefix $\langle q', D_{q'} \rangle$ of $\langle q, D_q \rangle$ such that $\langle q', D_{q'} \rangle \preceq \langle r', D_{r'} \rangle$. It is then a simple consequence of Proposition 3.2.17 and the prefix-closure of PD that $\langle r', D_{r'} \rangle \in augment(extend_{Act}(PD))$.

The prefix-closure of $augment(PT)$ and of $augment(PF)$ follows immediately from Proposition 3.2.17 and the prefix-closure of PT and PF.

For the last part of the proof, let $\langle r, F \rangle \in implied\text{-}failures_{Act}(augment(extend_{Act}(PD)))$ and let r' be a prefix of r. Thus, there is some $\langle p, D_p \rangle \in PD$ and some $\langle q, D_q \rangle$ with $\langle p, D_p \rangle \sqsubseteq \langle q, D_q \rangle \preceq \langle r, \{\mathrm{Events}_r\} \rangle$, and we can assume without loss of generality that $D_q = \{\mathrm{Events}_q\}$. Let q' be the restriction of q to r'; it is easy to see that q' is a prefix of q and $q' \preceq r'$. If q' is a prefix of p, then $\langle p, \emptyset \rangle \in PF$ and the prefix-closure of PF imply that $\langle q, \emptyset \rangle \in PF$, and so $\langle r, \emptyset \rangle \in augment(PF)$. For the other case, when q' is not a prefix of p, it follows by Proposition 3.2.17

that $\langle p', D_{p'}\rangle \sqsubseteq \langle q', \{\text{Events}_{q'}\}\rangle \preceq \langle r', \{\text{Events}_{r'}\}\rangle$ for some prefix $\langle p', D_{p'}\rangle$ of $\langle p, D_p\rangle$. Thus, it follows from the prefix-closure of PD that $\langle r', \emptyset\rangle \in \textit{implied-failures}_{Act}(\textit{augment}(\textit{extend}_{Act}(PD)))$, completing the proof. ∎

We now give the abstract characterizations of the $[\![\cdot]\!]^{\text{MAY}}$, $[\![\cdot]\!]^{\text{MUST}}$, and $[\![\cdot]\!]^{\text{TEST}}$ semantics.

Definition 4.2.2 Let Act be a finite set of labels containing the distinguished symbol $\surd$. A pair $\langle PT, Act\rangle$ is said to be MAY-respecting iff PT is a set of pomset-traces over Act such that

1. $\emptyset \in PT$.
2. PT is prefix-closed.
3. PT is augmentation-closed.
4. $p \in PT$ and p contains a $\surd$-labeled event implies that there is exactly one such event and this event is the unique maximum event of p.

Definition 4.2.3 Let Act be a finite set of labels containing the distinguished symbol $\surd$. A triple $\langle PF, PD, Act\rangle$ is said to be MUST-respecting iff PF is a set of pomset-failures over Act and PD is a set of pomset-divergences over Act such that the following properties hold:

1. Closure properties of PF:
   (a) $\langle\emptyset, \emptyset\rangle \in PF$.
   (b) PF is prefix-closed.
   (c) PF is augmentation-closed.
   (d) $\langle p, F\rangle \in PF$ and $F' \subset F$ implies that $\langle p, F'\rangle \in PF$.
   (e) $\langle p, F\rangle \in PF$, $c \in Act$, and $\langle p; c, \emptyset\rangle \notin PF$ implies that $\langle p, F \cup \{c\}\rangle \in PF$.
2. Closure properties of PD:
   (a) PD is prefix-closed.
   (b) PD is augmentation-closed.
   (c) PD is extension-closed under pomset-divergences over Act.
   (d) if $\langle p_0, D\rangle \in PD$, $r_1, \ldots, r_k$ are downward-closed subsets of p_0, and for all $n \geq 0$, there is some p_{n+1} with $\langle p_{n+1}, D\rangle \in PD$ and some $\{x_1, \ldots, x_k\} \subseteq \textit{max}(p_{n+1})$ such that $p_{n+1} - \{x_1, \ldots, x_k\} = p_n$ and $r_i = \textit{down}_{p_{n+1}}(x_i)$ for $1 \leq i \leq k$, then $\langle p_0, D \cup \{r_1, \ldots, r_k\}\rangle \in PD$.
   (e) $\langle p, D\rangle \in \textit{min}_{\sqsubseteq}(PD)$ implies that p contains no $\surd$-labeled events.
3. Mixed closure properties:
   (a) $\langle p, F\rangle \in PF$ and $\langle p, \{\text{Events}_p\}\rangle \notin PD$ and p contains a $\surd$-labeled event implies that there is exactly one such event, this event is the unique maximum event of p, and $\langle p, Act\rangle \in PF$.

(b) $\langle p, D\rangle \in PD$ and $F \subseteq Act$ implies that $\langle p, F\rangle \in PF$.

(c) if $\langle p_0, \emptyset\rangle \in PF$, $r_1, \ldots, r_k$ are downward-closed subsets of p_0, and for all $n \geq 0$, there is some $\langle p_{n+1}, \emptyset\rangle \in PF$ and some $\{x_1, \ldots, x_k\} \subseteq max(p_{n+1})$ such that $p_{n+1} - \{x_1, \ldots, x_k\} = p_n$ and $r_i = down_{p_{n+1}}(x_i)$ for $1 \leq i \leq k$, then $\langle p_0, \{r_1, \ldots, r_k\}\rangle \in PD$.

We remark that closure conditions (2d) and (3c) are necessary and sufficient to ensure that unbounded concurrency in PF or PD is possible only in the presence of appropriate causal divergences, which themselves may be concurrent.

Definition 4.2.4 Let Act be a finite set of labels containing the distinguished symbol $\surd$. A pair $\langle\langle PT, Act\rangle, \langle PF, PD, Act\rangle\rangle$ is said to be TEST-respecting iff $\langle PT, Act\rangle$ is MAY-respecting , $\langle PF, PD, Act\rangle$ is MUST-respecting , and

1. $p \in PT$ implies that $\langle p, \emptyset\rangle \in PF$.
2. $\langle p, F\rangle \in PF$ and $\langle p, \{\text{Events}_p\}\rangle \notin PD$ implies that $p \in PT$.
3. $\langle p, D\rangle \in min_{\sqsubseteq}(PD)$ implies that $p \in PT$.

Definition 4.2.5 Let Act be a finite set of labels containing the distinguished symbol $\surd$. Then $\mathcal{D}^{\text{MAY}}_{Act}$ is defined to be the set of all MAY-respecting pairs $\langle PT, Act\rangle$. Furthermore, $\sqsubseteq^{\text{MAY}}_{Act}$ is the binary relation on $\mathcal{D}^{\text{MAY}}_{Act}$ such that for every $\langle PT_1, Act\rangle$ and $\langle PT_2, Act\rangle$ in $\mathcal{D}^{\text{MAY}}_{Act}$, $\langle PT_1, Act\rangle \sqsubseteq^{\text{MAY}}_{Act} \langle PT_2, Act\rangle$ iff $PT_1 \subseteq PT_2$.

Definition 4.2.6 Let Act be a finite set of labels containing the distinguished symbol $\surd$. Then $\mathcal{D}^{\text{MUST}}_{Act}$ is defined to be the set of all MUST-respecting triples $\langle PF, PD, Act\rangle$. Furthermore, $\sqsubseteq^{\text{MUST}}_{Act}$ is the binary relation on $\mathcal{D}^{\text{MUST}}_{Act}$ such that for every $\langle PF_1, PD_1, Act\rangle$ and $\langle PF_2, PD_2, Act\rangle$ in $\mathcal{D}^{\text{MUST}}_{Act}$, $\langle PF_1, PD_1, Act\rangle \sqsubseteq^{\text{MUST}}_{Act} \langle PF_2, PD_2, Act\rangle$ iff $PF_1 \supseteq PF_2$ and $PD_1 \supseteq PD_2$.

Definition 4.2.7 Let Act be a finite set of labels containing the distinguished symbol $\surd$. Then $\mathcal{D}^{\text{TEST}}_{Act}$ is defined to be the set of all TEST-respecting pairs $\langle\langle PT, Act\rangle, \langle PF, PD, Act\rangle\rangle$. Furthermore, $\sqsubseteq^{\text{TEST}}_{Act}$ is the binary relation on $\mathcal{D}^{\text{TEST}}_{Act}$ such that for every $\langle \alpha_1, \beta_1\rangle$ and $\langle \alpha_2, \beta_2\rangle$ in $\mathcal{D}^{\text{TEST}}_{Act}$, $\langle \alpha_1, \beta_1\rangle \sqsubseteq^{\text{TEST}}_{Act} \langle \alpha_2, \beta_2\rangle$ iff $\alpha_1 \sqsubseteq^{\text{MAY}}_{Act} \alpha_2$ and $\beta_1 \sqsubseteq^{\text{MUST}}_{Act} \beta_2$.

We first show that $[\![\cdot]\!]^{\text{MAY}}$, $[\![\cdot]\!]^{\text{MUST}}$, and $[\![\cdot]\!]^{\text{TEST}}$ map WT Nets to elements of $\mathcal{D}^{\text{MAY}}_{Act}$, $\mathcal{D}^{\text{MUST}}_{Act}$, and $\mathcal{D}^{\text{TEST}}_{Act}$, respectively.

Theorem 4.2.8 Let $\langle N, Act\rangle$ be a WT Net. Then $[\![\langle N, Act\rangle]\!]^{\text{MAY}} \in \mathcal{D}^{\text{MAY}}_{Act}$, $[\![\langle N, Act\rangle]\!]^{\text{MUST}} \in \mathcal{D}^{\text{MUST}}_{Act}$, and $[\![\langle N, Act\rangle]\!]^{\text{TEST}} \in \mathcal{D}^{\text{TEST}}_{Act}$.

Proof. The proof for $[\![\cdot]\!]^{\text{MAY}}$ is a simple consequence of Definition 2.1.2, the definition of pomset-traces, Proposition 3.2.14, and Proposition 4.2.1; the details are left to the reader.

For the proof of $[\![\cdot]\!]^{\text{MUST}}$, let $\langle PF_N, PD_N, Act\rangle = [\![\langle N, Act\rangle]\!]^{\text{MUST}}$. All the closure conditions in Definition 4.2.6 except for (2d) and (3c) follow directly from the definition of $[\![\cdot]\!]^{\text{MUST}}$, the definition of WT Nets, Proposition 3.2.14, Proposition 3.2.15, and Lemma 4.2.1.

To prove that closure condition (2d) holds, let some sequence $\langle\langle p_n, D\rangle : n \geq 0\rangle$ and some set R of prefixes of p_0 be given that satisfy the hypothesis of (2d). For one case, suppose that all of the $\langle p_n, D\rangle \in augment(pomset\text{-}divergences(\langle N, Act\rangle))$. We recall that by definition of WT Nets, only a finite number of transitions are enabled under any reachable marking of $\langle N, Act\rangle$. Thus, it is possible for an unbounded number of concurrent events to be enabled after any prefix r of p_0 only if either a divergence is enabled immediately after r or a divergence is enabled "along the way to r," *i.e.*, immediately after some pomset r' with $\langle r', \{\text{Events}_{r'}\}\rangle \sqsubseteq \langle r, \{\text{Events}_r\}\rangle$. In either case, it then follows easily from the definition of pomset-divergences that $\langle p_0, D \cup R\rangle \in PD_N$.

For the other case, Proposition 3.2.14 and Proposition 3.2.15 imply that there is some sequence $\langle\langle q_n, D_n\rangle : n \geq 0\rangle$ such that every $\langle q_n, D_n\rangle \in augment(pomset\text{-}divergences(\langle N, Act\rangle))$ and $\langle q_n, D_n\rangle \sqsubseteq \langle p_n, D\rangle$. Since by Proposition 3.2.14 and Proposition 3.2.17, $augment(pomset\text{-}divergences(\langle N, Act\rangle))$ is prefix-closed, we can assume without loss of generality that for all $n \geq 0$, there is no event $x \in q_n$ such that $down_{q_n}(x) \supseteq d$ for some $d \in D_n$. Furthermore, since by hypothesis all events in $p_n - p_0$ are maximal in p_n, we can assume without loss of generality that every D_n is a non-empty set of prefixes of p_0. Thus, there are only a finite number of distinct D_n, and hence there is some subsequence $\langle\langle q_{n_k}, D_{n_k}\rangle : k \geq 0\rangle$ such that all the D_{n_k} are equal to some non-empty set D'. If there is some $x \in p_0$ and some q_{n_j} such that x is not an event of q_{n_j}, there must be some $d \in D'$ with $d \subseteq down_{p_0}(x)$; therefore, by our assumptions on the q_n, it follows that x is not an event of any of the q_{n_k}. We then have that for all n_k, the set $\text{Events}_{q_{n_k}} \cap \text{Events}_{p_0}$ is identical. Furthermore, for every $x_i \in (p_{n_j} - p_0) - q_{n_j}$, there must be some $d \in D'$ with $d \subseteq down_{p_{n_j}}(x_i) = r_i$ for some $r_i \in R$. Thus, it follows by our assumptions on the q_n that x_i is not an event of any of the q_{n_k}. Furthermore, since $augment(pomset\text{-}divergences(\langle N, Act\rangle))$ is prefix-closed, we can assume without loss of generality that q_{n_0} is a prefix of p_0 and that for all $i \geq 0$, $q_{n_{i+1}} - x = q_{n_i}$ for some $x \in max(q_{n_{i+1}})$.

It is now easy to see that either (i) all of the q_{n_k} are identical and are equal to some pomset q that is a prefix of p_0, and for every $r_i \in R$, there is some $d \in D'$ with $d \subseteq r_i$ or (ii) there is some $R' \subseteq R$ such that R' and the sequence $\langle\langle q_{n_k}, D'\rangle : k \geq 0\rangle$ satisfy the hypothesis of (2d), and for all $r_i \in R - R'$, there is some $d \in D'$ with $d \subseteq r_i$. If (i) holds, it follows easily by our construction of the $\langle q_{n_k}, D_{n_k}\rangle$ that $\langle q, D'\rangle \sqsubseteq \langle p, D \cup R\rangle$. If (ii) holds, it follows from the earlier case in this proof that $\langle q_{n_0}, D' \cup R'\rangle \in PD_N$. It follows from the construction of the q_{n_k} that $\langle q_{n_0}, D' \cup R'\rangle \sqsubseteq \langle p_0, D \cup R\rangle$, proving this case.

The proof of closure condition (3c) is quite similar and is left to the reader.

For the proof of $[\![\cdot]\!]^{\text{TEST}}$, it is straightforward to see from the definitions of pomset-traces, pomset-failures, and pomset-divergences of WT Nets, the definition of $[\![\cdot]\!]^{\text{TEST}}$, and Proposition 3.2.15 that the additional closure conditions hold. The proof of this case then follows easily from the previous cases. ∎

We now observe that:

Theorem 4.2.9 Let Act be a finite set of labels containing the distinguished symbol $\surd$. Then $\langle \mathcal{D}^{\text{MAY}}_{Act}, \sqsubseteq^{\text{MAY}}_{Act}\rangle$, $\langle \mathcal{D}^{\text{MUST}}_{Act}, \sqsubseteq^{\text{MUST}}_{Act}\rangle$, and $\langle \mathcal{D}^{\text{TEST}}_{Act}, \sqsubseteq^{\text{TEST}}_{Act}\rangle$ are complete partial orders.

Proof. It is easy to see that $\langle \mathcal{D}^{\text{MAY}}_{Act}, \sqsubseteq^{\text{MAY}}_{Act}\rangle$, $\langle \mathcal{D}^{\text{MUST}}_{Act}, \sqsubseteq^{\text{MUST}}_{Act}\rangle$, and $\langle \mathcal{D}^{\text{TEST}}_{Act}, \sqsubseteq^{\text{TEST}}_{Act}\rangle$ are partial orders.

For $\mathcal{D}^{\text{MAY}}_{Act}$, it follows immediately from the definition of MAY-respecting and $\mathcal{D}^{\text{MAY}}_{Act}$ that $\langle \emptyset, Act\rangle \in \mathcal{D}^{\text{MAY}}_{Act}$ and approximates every element $\langle PT, \Sigma\rangle \in \mathcal{D}^{\text{MAY}}_{Act}$ with $\Sigma = Act$. Let A

be a directed subset of $\mathcal{D}^{\text{MAY}}_{Act}$, and let $\{\langle PT_i, Act\rangle : i \geq 0\} = A$. It is very easy to see that $\langle \bigcup\{PT_i : i \geq 0\}, Act\rangle \in \mathcal{D}^{\text{MAY}}_{Act}$, proving this case.

For the proof of $\mathcal{D}^{\text{MUST}}_{Act}$, let PF^{Ω}_{Act} be the set of all pomset-failures over Act, and let PD^{Ω}_{Act} be the set of all pomset-divergences over Act. It is easy to see that $\langle PF^{\Omega}_{Act}, PD^{\Omega}_{Act}, Act\rangle \in \mathcal{D}^{\text{MUST}}_{Act}$ and approximates every element $\langle PF, PD, \Sigma\rangle \in \mathcal{D}^{\text{MUST}}_{Act}$ with $\Sigma = Act$.

We now show that every directed subset A of $\mathcal{D}^{\text{MUST}}_{Act}$ has a least upper bound in $\mathcal{D}^{\text{MUST}}_{Act}$. Let $\{\langle PF_i, PD_i, Act\rangle : i \geq 0\} \stackrel{\text{def}}{=} A$, let $PF_A = \bigcap\{PF_i : i \geq 0\}$, and let $PD_A = \bigcap\{PD_i : i \geq 0\}$. Clearly, it suffices to show that $\langle PF_A, PD_A, Act\rangle \in \mathcal{D}^{\text{MUST}}_{Act}$. All the closure properties follow trivially, *except* for (1e), (2e), and (3a).

(1e) $\langle p, F\rangle \in PF_A$, $c \in Act$, and $\langle p; c, \emptyset\rangle \notin PF_A$ implies that $\langle p, F \cup \{c\}\rangle \in PF_A$.

(2e) $\langle p, D\rangle \in min_{\sqsubseteq}(PD_A)$ for some D implies that p contains no $\surd$-labeled or γ-labeled events.

(3a) $\langle p, F\rangle \in PF_A$ and $\langle p, \{\text{Events}_p\}\rangle \notin PD_A$ and p contains a $\surd$-labeled event implies that there is exactly one such event, this event is the unique maximum event of p, and $\langle p, Act\rangle \in PF_A$.

To prove (1e), suppose for the sake of contradiction that for some $c \in Act$, $\langle p, F\rangle \in PF_A$, $\langle p; c, \emptyset\rangle \notin PF_A$, and $\langle p, F \cup \{c\}\rangle \notin PF_A$. Then $\langle p, F\rangle \in PF_i$ for all $\langle PF_i, PD_i, Act\rangle \in A$, $\langle p, F \cup \{c\}\rangle \notin PF_m$ for some $\langle PF_m, PD_m, Act\rangle \in A$, and $\langle p; c, \emptyset\rangle \notin PF_j$ for some $\langle PF_j, PD_j, Act\rangle \in A$. Since A is a directed set, there would be some $\langle PF_k, PD_k, Act\rangle \in A$ that is an upper bound of both $\langle PF_m, PD_m, Act\rangle$ and $\langle PF_j, PD_j, Act\rangle$. This would imply that $\langle p, F\rangle \in PF_k$, $\langle p, F \cup \{c\}\rangle \notin PF_k$, and $\langle p; c, \emptyset\rangle \notin PF_k$, a contradiction since $\langle PF_k, PD_k, Act\rangle \in A \subseteq \mathcal{D}^{\text{MUST}}_{Act}$. Thus, (1e) must hold after all for PF_A.

To prove (2e), it suffices to show that $\langle p, D\rangle \in min_{\sqsubseteq}(PD_A)$ implies that $\langle p, D\rangle \in min_{\sqsubseteq}(PD_j)$ for some $\langle PF_j, PD_j, Act\rangle \in Act$. Clearly, there are only a finite number of distinct pomset-divergences $\langle p_i, D_i\rangle$ with $\langle p_i, D_i\rangle \sqsubset \langle p, D\rangle$, and PD_A does not contain any of them. Hence for every such $\langle p_i, D_i\rangle$, there is some $\langle PF_i, PD_i, Act\rangle \in A$ such that $\langle p_i, D_i\rangle \notin PD_i$. By compactness, there is then some $\langle PF_j, PD_j, Act\rangle \in A$ with $\langle p, D\rangle \in min_{\sqsubseteq}(PD_j)$.

The proof of (3a) is very similar and is left to the reader.

The proof for $\mathcal{D}^{\text{TEST}}_{Act}$ is a straightforward combination and adaptation of the proofs of the previous two cases. The details are left to the reader. ∎

We now give a finite characterization of the compact elements of these domains:

Definition 4.2.10 Let Act be a finite set of labels containing the distinguished symbol $\surd$. A pair $\langle PT, Act\rangle$ is a *finite candidate* of $\mathcal{D}^{\text{MAY}}_{Act}$ iff $\langle PT, Act\rangle \in \mathcal{D}^{\text{MAY}}_{Act}$ and PT is a *finite* set.

A triple $\langle PF, PD, Act\rangle$ is a *finite candidate* of $\mathcal{D}^{\text{MUST}}_{Act}$ iff Act is a finite set of labels and there is some *finite* set $\mathcal{PF}_{\text{fin}}$ of pomset-failures over Act and some *finite* set $\mathcal{PD}_{\text{fin}}$ of pomset-divergences over Act such that:

- $\langle \mathcal{PF}_{\text{fin}}, \emptyset, Act\rangle \in \mathcal{D}^{\text{MUST}}_{Act}$.
- $\mathcal{PD}_{\text{fin}}$ is prefix-closed.
- $\langle p, D\rangle \in \mathcal{PD}_{\text{fin}}$ implies that $\langle p, \emptyset\rangle \in \mathcal{PF}_{\text{fin}}$.

- $PD = augment(extend_{Act}(\mathcal{PD}_{\text{fin}}))$.
- $PF = \mathcal{PF}_{\text{fin}} \cup implied\text{-}failures_{Act}(PD)$.

A tuple $\langle \alpha, \beta \rangle$ is a *finite candidate* of $\mathcal{D}^{\text{TEST}}_{Act}$ iff $\langle \alpha, \beta \rangle \in \mathcal{D}^{\text{TEST}}_{Act}$, α is a finite candidate of $\mathcal{D}^{\text{MAY}}_{Act}$, and β is a finite candidate of $\mathcal{D}^{\text{MUST}}_{Act}$.

Lemma 4.2.11 Let Act be a finite set of labels containing the distinguished symbol $\surd$. Then all finite candidates of $\mathcal{D}^{\text{MAY}}_{Act}$, $\mathcal{D}^{\text{MUST}}_{Act}$, and $\mathcal{D}^{\text{TEST}}_{Act}$ are compact elements of $\mathcal{D}^{\text{MAY}}_{Act}$, $\mathcal{D}^{\text{MUST}}_{Act}$, and $\mathcal{D}^{\text{TEST}}_{Act}$, respectively.

Proof. For the proof for $\mathcal{D}^{\text{MAY}}_{Act}$, let $\langle PT, Act \rangle$ be a finite candidate, and let $A \subseteq \mathcal{D}^{\text{MAY}}_{Act}$ be a directed set such that $\langle PT, Act \rangle \sqsubseteq^{\text{MAY}}_{Act} \bigsqcup A$. Since PT is finite, it follows immediately from the directedness of A that there is some $\langle PT_k, Act \rangle \in A$ with $PT \subseteq PT_k$.

For the proof for $\mathcal{D}^{\text{MUST}}_{Act}$, let $\langle PF, PD, Act \rangle$ be a finite candidate, and let $\mathcal{PF}_{\text{fin}}, \mathcal{PD}_{\text{fin}}$ be given. We first show that $\langle PF, PD, Act \rangle$ is in fact an element of $\mathcal{D}^{\text{MUST}}_{Act}$. Proposition 4.2.1 immediately implies that closure conditions (2a), (2b), and (2c) of Definition 4.2.6 hold for PD. Proposition 3.2.15, the given properties of $\mathcal{PD}_{\text{fin}}$, and the fact that $\langle \mathcal{PF}_{\text{fin}}, \emptyset, Act \rangle \in \mathcal{D}^{\text{MUST}}_{Act}$ immediately imply that closure condition (2e) holds. To show (2d), suppose there is some $\langle p_n, D \rangle \in PD$ for $n \geq 0$ and some $r_1, \ldots, r_k$ satisfying the hypothesis. By definition of PD and the finiteness of $\mathcal{PD}_{\text{fin}}$, clearly an infinite number of the $\langle p_n, D \rangle$ must be augmentations of extensions of some common $\langle p, D' \rangle \in \mathcal{PD}_{\text{fin}}$ such that $\langle p, D' \cup \{r_1, \ldots, r_k\} \rangle \in \mathcal{PD}_{\text{fin}}$. It is then easy to show that the other closure properties of PD imply that $\langle p_0, D \cup \{r_1, \ldots, r_k\} \rangle \in PD$; the details are left to the reader. The proof of closure condition (3c) for PF is similar.

From the definition of $\mathcal{D}^{\text{MUST}}_{Act}$, Proposition 4.2.1, and the fact that $\langle \mathcal{PF}_{\text{fin}}, \emptyset, Act \rangle \in \mathcal{D}^{\text{MUST}}_{Act}$, it is easy to see that all the other closure conditions hold, proving that $\langle PF, PD, Act \rangle \in \mathcal{D}^{\text{MUST}}_{Act}$.

Let $A \subseteq \mathcal{D}^{\text{MUST}}_{Act}$ be a directed set such that $\langle PF, PD, Act \rangle \sqsubseteq^{\text{MUST}}_{Act} \langle PF_A, PD_A, Act \rangle = \bigsqcup A$.

Let $k = max\{|p| : \langle p, \emptyset \rangle \in \mathcal{PF}_{\text{fin}}\}$, and let m be the number of distinct pomsets in $\mathcal{PF}_{\text{fin}}$, *i.e.*, $m = |\{p : \langle p, \emptyset \rangle \in \mathcal{PF}_{\text{fin}}\}|$. Let U be the following set of pomset-failures:

$$U = \{\langle p, F \rangle : \langle p, F \rangle \text{ is a pomset-failure over } Act, \langle p, F \rangle \notin PF, \text{ and } |p| \leq m \cdot 8^k\}.$$

Clearly, U is finite and does not intersect PF, and hence does not intersect PF_A. The directedness of A then implies that there is some $\langle PF_i, PD_i, Act \rangle \in A$ such that PF_i does not intersect U.

To show that $PF_i \subseteq PF$, let $\langle r, F_r \rangle$ be a pomset-failure over Act with $\langle r, F_r \rangle \notin PF$. If $|r| \leq m \cdot 8^k$, then clearly, $\langle r, F_r \rangle \notin PF_i$. Otherwise, when $|r| > m \cdot 8^k$; we will show that there is some prefix r' of r such that $|r'| \leq m \cdot 8^k$ and $\langle r', \emptyset \rangle \notin PF$. Let q be a prefix of r such that $\langle q, \emptyset \rangle \in \mathcal{PF}_{\text{fin}}$, and q is maximal with respect to these properties; clearly, such a pomset q exists since by Definition 4.2.10, $\langle \emptyset, \emptyset \rangle \in \mathcal{PF}_{\text{fin}}$. Furthermore, by definition of k, it follows that $|q| \leq k$.

We obtain a prefix r' of r by iterating the following procedure until termination. First set r' to be the prefix of r with carrier $\{x \in \text{Events}_r : depth_r(x) \leq k+1\}$. Pick some $x \in r' - q$ such that x is in some cut C of r' of size strictly larger than $k+1$. If there are $k+1$ distinct $x_1, \ldots, x_{k+1} \in (C - \{x\})$ such that for all q' with $\langle q', \emptyset \rangle \in \mathcal{PF}_{\text{fin}}$, $down_{r'}(x) \cap q' = down_{r'}(x_1) \cap q' = \ldots = down_{r'}(x_{k+1}) \cap q'$, then remove from r' the set of events $\{y \in \text{Events}_{r'} : x \leq_r y\}$, and re-set r' to be the resulting pomset. Repeat this procedure until there are no events x in r'

satisfying the above conditions. Clearly, the resulting pomset r' is a prefix of r. Furthermore, since $\mathcal{PF}_{\text{fin}}$ contains at most m different pomsets, each of size bounded by k, and each such pomset has at most 2^k prefixes, it is easy to see that the size of any cut in r' is bounded by $m \cdot 2^k \cdot (k+1)$. Furthermore, since the depth of r' is bounded by $k+1$, it follows that $|r'| \leq m \cdot 8^k$.

It is easy to see by construction of r' that q is a prefix of r' and that $r' \neq q$; thus $\langle r', \emptyset \rangle \notin \mathcal{PF}_{\text{fin}}$ by maximality of q. Suppose for the sake of contradiction that $\langle r', \{\text{Events}_{r'}\} \rangle \in PD$; then by Proposition 3.2.15 and the prefix-closure of $\mathcal{PD}_{\text{fin}}$, there is some $\langle p, D_p \rangle \in \mathit{augment}(\mathcal{PD}_{\text{fin}})$ such that $\langle p, D_p \rangle \sqsubseteq \langle r', \{\text{Events}_{r'}\} \rangle$. Thus, Definition 4.2.10 implies that $\langle p, \emptyset \rangle \in \mathcal{PF}_{\text{fin}}$ and hence that $|p| \leq k$. For every $y \in r - r'$, either (i) there is some $x \in r - r'$ with $x \leq_r y$, some cut C of r', and some $k+1$ distinct $x_1, \ldots, x_{k+1} \in C$ such that for all q' with $\langle q', \emptyset \rangle \in \mathcal{PF}_{\text{fin}}$, $\mathit{down}_{r'}(x) \cap q' = \mathit{down}_{r'}(x_1) \cap q' = \ldots = \mathit{down}_{r'}(x_{k+1}) \cap q'$ or (ii) $\mathit{depth}_r(y) > k+1$. Suppose (i) holds, then since p has concurrency bounded by k, there must be some such x_i with $x_i \notin p$. So, there is some $d \in D_p$ with $d \subseteq \mathit{down}_{r'}(x_i) \cap p$, and so by hypothesis, $\mathit{down}_{r'}(x_i) \cap p = \mathit{down}_{r'}(x) \cap p$, and so $d \subseteq \mathit{down}_{r'}(x) \subseteq \mathit{down}_r(y)$. If (ii) holds, then there must be some $z \in r$ such that $z \leq_r y$ and $\mathit{depth}_r(z) = k+1$. If $z \in r'$, then since the depth of p is bounded by k, $z \notin p$ and there is some $d \in D_p$ such that $d \subseteq \mathit{down}_{r'}(z) \subseteq \mathit{down}_r(y)$. If $z \notin r'$, then there is some $x \leq_r z$ satisfying the conditions of (i), so there is some $d \in D_p$ such that $d \subseteq \mathit{down}_{r'}(x) \subseteq \mathit{down}_r(z) \subseteq \mathit{down}_r(y)$. Thus, $\langle p, D_p \rangle \sqsubseteq \langle r, \{\text{Events}_r\} \rangle$, and so by Definition 4.2.6, $\langle r, \{\text{Events}_r\} \rangle \in PD$, a contradiction since $\langle r, F_r \rangle \notin PF$. Hence, $\langle r', \{r'\} \rangle \notin PD$, and since $\langle r', \emptyset \rangle \notin \mathcal{PF}_{\text{fin}}$, it follows that $\langle r', \emptyset \rangle \notin PF$ and is thus in U.

Now, $\langle r, \emptyset \rangle \in PF_i$ would imply that $\langle r', \emptyset \rangle \in PF_i$, since PF_i is an element of $\mathcal{D}^{\text{MUST}}_{Act}$ and thus prefix-closed. But this would imply that $PF_i \cap U \neq \emptyset$, a contradiction. Thus, $PF_i \subseteq PF$.

We now give the proof for pomset-divergences. Let V be the following set of pomset-divergences:

$$V = \{\langle p, D_p \rangle : \langle p, D_p \rangle \text{ is a pomset-divergence, } \langle p, D_p \rangle \notin PD, \text{ and } \langle p, \emptyset \rangle \in \mathcal{PF}_{\text{fin}}\}.$$

Clearly, V is finite and does not intersect PD, and hence does not intersect PD_A. The directedness of A then implies that there is some $\langle PF_j, PD_j, Act \rangle \in A$ such that PD_j does not intersect V and $PF_j \subseteq PF_i \subseteq PF$.

We now show that $PD_j \subseteq PD$. Let $\langle r, D_r \rangle$ be a pomset-divergence over Act with $\langle r, D_r \rangle \notin PD$. If $\langle r, \emptyset \rangle \notin PF$, then $\langle r, \emptyset \rangle \notin PF_j$, and so $\langle r, D_r \rangle \notin PD_j$ from the closure properties of elements of $\mathcal{D}^{\text{MUST}}_{Act}$. If $\langle r, \emptyset \rangle \in \mathcal{PF}_{\text{fin}}$, then $\langle r, D_r \rangle \in V$, and so $\langle r, D_r \rangle \notin PD_j$. For the last case, we have that $\langle r, \emptyset \rangle \notin \mathcal{PF}_{\text{fin}}$ and $\langle r, \emptyset \rangle \in PF$. We define an extension $\langle r', \{\text{Events}_{r'}\} \rangle$ of $\langle r, D_r \rangle$ as follows: let the carrier of r' be the carrier of r together with some disjoint set of events $\{(d, i) : d \in D_r \text{ and } 1 \leq i \leq k+1\}$. The ordering and labeling of r' agrees with r on events in r, all the new events (d, i) are maximal in r', and for all (d, i), $\mathit{down}_{r'}((d, i)) = d$. The labels of the (d, i) are arbitrarily chosen to be labels in Act. It is easy to see that $\langle r, D_r \rangle \sqsubset \langle r', \{\text{Events}_{r'}\} \rangle$ and that $\langle r', \emptyset \rangle \notin \mathcal{PF}_{\text{fin}}$ since $\langle r, \emptyset \rangle \notin \mathcal{PF}_{\text{fin}}$ and by Definition 4.2.10, $\mathcal{PF}_{\text{fin}}$ is prefix-closed. Suppose for the sake of contradiction that $\langle r', \{\text{Events}_{r'}\} \rangle \in PD$; then by Proposition 3.2.15 and the prefix-closure of $\mathcal{PD}_{\text{fin}}$ there is some $\langle p', D_{p'} \rangle \in \mathit{augment}(\mathcal{PD}_{\text{fin}})$ with $\langle p', D_{p'} \rangle \sqsubset \langle r', \{\text{Events}_{r'}\} \rangle$. Since the concurrency in p' is bounded by k, there must be some $(d, i) \in r' - p'$ for every $d \in D_r$. Hence, for every $d \in D_r$, there is some $d' \in D_{p'}$ with $d' \subseteq d$. Let p be p' with all (d, i) events removed, and let $D_p = \{d' \in D_{p'} : d' \subseteq p\}$. Then it is easy to see that D_p is non-empty, $\langle p, D_p \rangle$ is a prefix of $\langle p', D_{p'} \rangle$, and $\langle p, D_p \rangle \sqsubseteq \langle r, D_r \rangle$. But by Definition 4.2.6, this would imply that $\langle r, D_r\rangle \in PD$, a contradiction. Hence, $\langle r', \{\text{Events}_{r'}\}\rangle \notin PD$ after all, and since $\langle r', \emptyset\rangle \notin \mathcal{PF}_{\text{fin}}$, it follows that $\langle r', \emptyset\rangle \notin PF$.

Now $\langle r, D_r\rangle \in PD_j$ would imply by the closure properties that $\langle r', \{\text{Events}_{r'}\}\rangle \in PD_j$, and hence that $\langle r', \emptyset\rangle \in PF_j$, a contradiction since $PF_j \subseteq PF$. Thus,

$$\langle PF, PD, Act\rangle \sqsubseteq^{\text{MUST}}_{Act} \langle PF_j, PD_j, Act\rangle \in A,$$

proving this case.

The proof for $\mathcal{D}^{\text{TEST}}_{Act}$ is a simple consequence of Definition 4.2.7 and the previous two cases. ∎

Lemma 4.2.12 Let Act be a finite set of labels containing the distinguished symbol $\surd$. For every $\langle PT, Act\rangle \in \mathcal{D}^{\text{MAY}}_{Act}$, there is a directed set $A_1 \subseteq \mathcal{D}^{\text{MAY}}_{Act}$ of finite candidates of $\mathcal{D}^{\text{MAY}}_{Act}$ with $\bigsqcup A_1 = \langle PT, Act\rangle$. For every $\langle PF, PD, Act\rangle \in \mathcal{D}^{\text{MUST}}_{Act}$, there is a directed set $A_2 \subseteq \mathcal{D}^{\text{MUST}}_{Act}$ of finite candidates of $\mathcal{D}^{\text{MUST}}_{Act}$ with $\bigsqcup A_2 = \langle PF, PD, Act\rangle$. For every $\langle\langle PT, Act\rangle, \langle PF, PD, Act\rangle\rangle \in \mathcal{D}^{\text{TEST}}_{Act}$, there is a directed set $A_3 \subseteq \mathcal{D}^{\text{TEST}}_{Act}$ of finite candidates of $\mathcal{D}^{\text{TEST}}_{Act}$ with $\bigsqcup A_3 =$ $\langle\langle PT, Act\rangle, \langle PF, PD, Act\rangle\rangle$.

Proof. We first give the proof for $\mathcal{D}^{\text{MAY}}_{Act}$. For every $n \geq 0$, we define the n^{th} approximation, $\langle PT_n, Act\rangle$, to $\langle PT, Act\rangle$ as follows:

$$PT_n = \{p \in PT : |p| \leq n\}$$

We recall that by definition of $\mathcal{D}^{\text{MAY}}_{Act}$, Act is finite. It is then easy to see that each $\langle PT_n, Act\rangle$ is a finite candidate, each $\langle PT_n, Act\rangle \sqsubseteq^{\text{MAY}}_{Act} \langle PT, Act\rangle$, and that the $\langle PT_n, Act\rangle$ form a chain in $\mathcal{D}^{\text{MAY}}_{Act}$. Thus, $\bigsqcup\{\langle PT_n, Act\rangle : n \geq 0\} \sqsubseteq^{\text{MAY}}_{Act} \langle PT, Act\rangle$. For the other direction, let $p \in PT$; then $p \in PT_{|p|}$, proving this case.

We now prove the case for $\mathcal{D}^{\text{MUST}}_{Act}$. For every $n \geq 0$, we define n^{th} approximation, $\langle PF_n, PD_n, Act\rangle$, to $\langle PF, PD, Act\rangle$ as follows. The idea is that each n^{th} approximation is generated by the set of pomset-failures and pomset-divergences in PF and PD whose depth is bounded by n. However, PF and PD may have unbounded concurrency and hence may an infinite number of pomsets of any given depth n. In order to ensure that each $\mathcal{PF}_{\text{fin-}n}$ and $\mathcal{PD}_{\text{fin-}n}$ are finite, we use only the minimal elements of PF and PD; in order to ensure that $\mathcal{PF}_{\text{fin}}$ and $\mathcal{PD}_{\text{fin}}$ satisfy the conditions in Definition 4.2.6, we close under prefixing.

Since we want the resulting finite candidates to approximate $\langle PF, PD, Act\rangle$, we need to ensure that all pomset-failures and pomset-divergences in PF and PD are generated by the augmentation and extension closure of $\mathcal{PF}_{\text{fin}}$ and $\mathcal{PD}_{\text{fin}}$. Thus, in $\mathcal{PF}_{\text{fin}}$ and $\mathcal{PD}_{\text{fin}}$, we extend past all pomsets of depth *equal to* n by throwing in all failure sets and throwing in all divergences that causally follow any chain of length of n.

$$\begin{aligned} PF' = \{\langle p, F\rangle \in PF : {} & p \text{ is a prefix of some } q \text{ such that } \langle q, \emptyset\rangle \in PF, \\ & \text{and either } \langle q, \{\text{Events}_q\}\rangle \notin PD \text{ or } \langle q, D\rangle \in min_{\sqsubseteq}(PD) \text{ for some } D\} \end{aligned}$$

$$\begin{aligned} \mathcal{PF}_{\text{fin-}n} = augment(&\{\langle p, F\rangle \in PF' : depth(p) \leq n\} \\ & \cup \{\langle p, F\rangle : \langle p, \emptyset\rangle \in PF',\ depth(p) = n, \text{ and } F \subseteq Act\}) \end{aligned}$$

$$\mathcal{PD}_{\text{fin-}n} = \{\langle p, D \cup D' \rangle : \langle p, \emptyset \rangle \in \mathcal{PF}_{\text{fin-}n},\ D \cup D' \neq \emptyset,\ D' \subseteq \{down_p(x) \cup \{x\} : depth_p(x) = n\},$$
$$\text{and either } \langle p, D \rangle \in PD \text{ or } D = \emptyset\}$$

$$PD_n = augment(extend_{Act}(\mathcal{PD}_{\text{fin-}n}))$$

$$PF_n = \mathcal{PF}_{\text{fin-}n} \cup implied\text{-}failures_{Act}(PD_n)$$

We first show that $\langle PF_n, PD_n, Act \rangle$ is a finite candidate, recalling that by Definition 4.2.6, Act is finite. Therefore, an infinite number of distinct $\langle p_i, \emptyset \rangle \in \mathcal{PF}_{\text{fin-}n}$ would imply unbounded concurrency in $\mathcal{PF}_{\text{fin-}n}$, and so the closure conditions (2d) and (3c) of Definition 4.2.6 would immediately contradict the definitions of PF' and $\mathcal{PF}_{\text{fin-}n}$. Thus, $\mathcal{PF}_{\text{fin-}n}$ must be finite. We now show that $\langle \mathcal{PF}_{\text{fin-}n}, \emptyset, Act \rangle \in \mathcal{D}^{\text{MUST}}_{Act}$. It follows easily from the closure conditions of PF and Proposition 4.2.1 that closure conditions (1a)—(1d) hold for $\mathcal{PF}_{\text{fin-}n}$. For closure condition (1e), suppose that $\langle r, F \rangle \in \mathcal{PF}_{\text{fin-}n}$ and $c \in Act$ and $\langle r, F \cup \{c\} \rangle \notin \mathcal{PF}_{\text{fin-}n}$. Then $r \succeq p$ of some $\langle p, F \rangle \in PF'$ with $depth(p) \leq n$. Since $\langle r, F \cup \{c\} \rangle \notin \mathcal{PF}_{\text{fin-}n}$, it is easy to see that $\langle p, F \cup \{c\} \rangle \notin \mathcal{PF}_{\text{fin-}n}$ and $depth(p) < n$. Furthermore, the closure conditions of PF, PD and the definition of PF' imply that $\langle p, F \rangle \in PF$ and $\langle p, F \cup \{c\} \rangle \notin PF$, and hence that $\langle p, \{\text{Events}_p\} \rangle \notin PD$. Moreover, the closure condition (1e) on PF implies that $\langle p; c, \emptyset \rangle \in PF$. Clearly, $p; c \preceq r; c$ and since $depth(p) < n$, it follows that $depth(p; c) \leq n$. If $\langle p; c, \emptyset \rangle \notin PF'$, then by Proposition 3.2.15 and the definition of PF', there must be some $\langle q, D_q \rangle \in min_{\sqsubseteq}(PD)$ such that $q \neq p; c$ and $\langle q, D_q \rangle \sqsubset \langle p; c, \{\text{Events}_{p;c}\} \rangle$. Hence q is a prefix of p, and so $\langle q, D_q \rangle \sqsubseteq \langle p, \{\text{Events}_p\} \rangle$, implying that $\langle p, \{\text{Events}_p\} \rangle \in PD$, a contradiction. Thus, $\langle p; c, \emptyset \rangle \in PF'$, from which it is clear that $\langle r; c, \emptyset \rangle \in \mathcal{PF}_{\text{fin-}n}$.

It follows easily from the prefix-closure of PD and the prefix-closure of $\mathcal{PF}_{\text{fin-}n}$ that $\mathcal{PD}_{\text{fin-}n}$ is also prefix-closed. The construction of $\langle PF_n, PD_n, Act \rangle$ then immediately implies that it satisfies Definition 4.2.10 and hence is a finite candidate.

We now show that $\langle PF_n, PD_n, Act \rangle \sqsubseteq^{\text{MUST}}_{Act} \langle PF, PD, Act \rangle$. Let $\langle q, D_q \rangle \in PD$, and using Proposition 3.2.15 and the prefix-closure of PD, let $\langle r, D_r \rangle \in min_{\sqsubseteq}(PD)$ such that $\langle r, D_r \rangle \sqsubseteq \langle q, D_q \rangle$. Then $\langle r, D_r \rangle \in PF'$. If $depth(r) \leq n$, then clearly, $\langle r, D_r \rangle \in PD_n$ and hence so is $\langle q, D_q \rangle$. If $depth(r) \geq n$, let r' be the (necessarily unique) maximal prefix of r of depth n and let $D_{r'} = \{d \in D_r : d' \subseteq r'\}$. By the closure conditions of PF and PD, $\langle r', \emptyset \rangle \in PF$ and $\langle r', D_{r'} \rangle \in PD$ if $D_{r'}$ is non-empty. For one case, suppose that $D_{r'}$ is non-empty. Then by Proposition 3.2.15, there is some $\langle r'', D_{r''} \rangle \in min_{\sqsubseteq}(PD)$ such that $\langle r'', D_{r''} \rangle \sqsubseteq \langle r', D_{r'} \rangle$, and so it is easy to see by the prefix-closure of PF and the definition of $\mathcal{PF}_{\text{fin-}n}$ that $\langle r'', \emptyset \rangle \in \mathcal{PF}_{\text{fin-}n}$. It is easy to show that $\langle r'', D_{r''} \cup \{down_{r''}(x) \cup \{x\} : depth_{r''}(x) = n\} \rangle \in \mathcal{PD}_{\text{fin-}n}$ and $\langle r'', D_{r''} \cup \{down_{r''}(x) \cup \{x\} : depth_{r''}(x) = n\} \rangle \sqsubseteq \langle r', D_{r'} \cup \{down_{r'}(x) \cup \{x\} : depth_{r'}(x) = n\} \rangle \sqsubseteq \langle r, D_r \rangle \sqsubseteq \langle q, D_q \rangle$, so $\langle q, D_q \rangle \in PD_n$. The details are simple and are left to the reader, as is the other case, which is similar. Thus, $PD \subseteq PD_n$.

To show that $PF \subseteq PF_n$, let $\langle q, F \rangle \in PF$. If $\langle q, \{\text{Events}_q\} \rangle \in PD$, then $\langle q, \{\text{Events}_q\} \rangle \in PD_n$, and hence $\langle q, F \rangle \in PF_n$. Otherwise, $\langle q, F \rangle \in PF'$. For one case, if $depth(q) \leq n$, it is clear that $\langle q, F \rangle \in PF_n$. For the other case, using a proof similar to that for pomset-divergences, it is easy to show that $\langle q, \{\text{Events}_q\} \rangle \in PD_n$, and hence $\langle q, F \rangle \in PF_n$; the details are straightforward and are left to the reader. Thus, $\langle PF_n, PD_n, Act \rangle \sqsubseteq^{\text{MUST}}_{Act} \langle PF, PD, Act \rangle$.

To show that the set of n^{th}-approximations forms a directed set, we show that for every $n \geq 0$, $\langle PF_n, PD_n, Act\rangle \sqsubseteq^{\mathrm{MUST}}_{Act} \langle PF_{n+1}, PD_{n+1}, Act\rangle$. Let $\langle q, D_q\rangle \in PD_{n+1}$; then $\langle q, D_q\rangle$ is an augmentation of an extension of some $(r, D_r) \in \mathcal{PD}_{\text{fin-}n+1}$. For one case, suppose that $depth(r) \leq n$. Then it is easy to see that $(r, D_r) \in PD$, so $\langle r, D_r\rangle \in PD_n$ by the earlier proof, and therefore so is $\langle q, D_q\rangle$. The proof of the other case, when $depth(r) = n+1$, is very similar to the proof that $PD \supseteq PD_n$, and the details are left to the reader. Thus, $PD_{n+1} \subseteq PD_n$.

To show that $PF_{n+1} \subseteq PF_n$, let $\langle q, F\rangle \in PF_{n+1}$. If $\langle q, \{\text{Events}_q\}\rangle \in PD_{n+1}$, then by the above case, $\langle q, \{\text{Events}_q\}\rangle \in PD_n$, and hence $\langle q, F\rangle \in PF_n$. Otherwise, if $depth(q) \leq n$, it is clear that $\langle q, F\rangle \in PF_n$. For the other case, using a proof similar to that for pomset-divergences, it is easy to show that $\langle q, \{\text{Events}_q\}\rangle \in PD_n$, and hence that $\langle q, F\rangle \in PF_n$; the details are left to the reader. Thus, $\langle PF_n, PD_n, Act\rangle \sqsubseteq^{\mathrm{MUST}}_{Act} \langle PF_{n+1}, PD_{n+1}, Act\rangle$.

For the last part of the proof, we show that $\langle PF, PD, Act\rangle = \bigsqcup\{\langle PF_n, PD_n, Act\rangle : n \geq 0\}$. One direction follows immediately from the fact that every $\langle PF_n, PD_n, Act\rangle \sqsubseteq^{\mathrm{MUST}}_{Act} \langle PF, PD, Act\rangle$. For the other direction, let $\langle p, D_p\rangle \in \bigcap\{PD_n : n \geq 0\}$, and let $k = depth(p)$. Then $\langle p, D_p\rangle \in PD_{k+1}$ implies that $\langle p, D_p\rangle$ is an augmentation of an extension of some $\langle q, D_q\rangle \in PD_{\text{fin-}k+1}$. Since augmentation and extension only increase depth, $depth(q) \leq k$, from which it is easy to see that $\langle q, D_q\rangle \in PD$ and hence that $\langle p, D_p\rangle \in PD$. For the other case, let $\langle p, F\rangle \in \bigcap\{PF_n : n \geq 0\}$, and let $k = depth(p)$. Then $\langle p, F\rangle \in PF_{k+1}$ implies that $\langle p, F\rangle \in PF_{\text{fin-}k+1}$ or $\langle p, \{\text{Events}_p\}\rangle \in PD_{k+1}$, both of which imply that $\langle p, F\rangle \in PF$. Thus, $\langle PF, PD, Act\rangle = \bigsqcup\{\langle PF_n, PD_n, Act\rangle : n \geq 0\}$, proving this case.

We now prove the case for $\mathcal{D}^{\mathrm{TEST}}_{Act}$. For every $1 \leq n \leq m$, we define the $(n,m)^{\mathrm{th}}$ approximation, $\langle PF_{(n,m)}, PD_n, Act\rangle$ to $\langle PF, PD, Act\rangle$ as follows. First, $PF', \mathcal{PD}_{\text{fin-}n}, PD_n$ are defined as in the proof of the above case. Furthermore, in order to appropriately define the approximate pomset-traces, we may also need to replace some non-maximal pomsets of depth n in $PF - PF'$. Since we only want to construct finite sets of pomset-traces, we define the $(n,m)^{\mathrm{th}}$ approximation by replacing all such pomsets of depth bounded by n and *size* bounded by m. The formal definitions are as follows:

$$\mathcal{PF}_{\text{fin-}(n,m)} = \mathcal{PF}_{\text{fin-}n} \cup augment(\{\langle p, F\rangle \in PF : p \in PT,\ depth(p) \leq n \text{ and } |p| \leq m\})$$

$$PT_{(n,m)} = \{p : \langle p, \emptyset\rangle \in \mathcal{PF}_{\text{fin-}(n,m)}\}$$

$$PF_{(n,m)} = \mathcal{PF}_{\text{fin-}(n,m)} \cup \textit{implied-failures}_{Act}(PD_n)$$

The proof is a straightforward combination and adaptation of the proofs of the above cases; the details are left to the reader. ∎

We now have:

Theorem 4.2.13 Let *Act* be a finite set of labels containing the distinguished symbol $\surd$. Then $\mathcal{D}^{\mathrm{MAY}}_{Act}$, $\mathcal{D}^{\mathrm{MUST}}_{Act}$, and $\mathcal{D}^{\mathrm{TEST}}_{Act}$ are algebraic cpo's.

The theorem is a simple consequence of the definition of compact elements, Lemma 4.2.11, and Lemma 4.2.12 (*cf.* [18]).

We now show that all compact elements are definable as the $[\![\cdot]\!]^{\text{MAY}}$ meanings of WT Nets:

Theorem 4.2.14 Let Act be a finite set of labels containing the distinguished symbol $\surd$. For every compact element $\langle PT, Act\rangle \in \mathcal{D}^{\text{MAY}}_{Act}$, there is some WT Net $\langle N, Act\rangle$ with $[\![\langle N, Act\rangle]\!]^{\text{MAY}} = \langle PT, Act\rangle$.

Proof. Let $\langle PT, Act\rangle$ be a compact element of $\mathcal{D}^{\text{MAY}}_{Act}$. As a simple consequence of Lemma 4.2.12 and the definition of compactness (*cf.* [18]), $\langle PT, Act\rangle$ is a finite candidate of $\mathcal{D}^{\text{MAY}}_{Act}$. Thus, PT is finite set of pomsets over Act.

We first build a tree whose nodes are labeled with *valid pairs* of the form $\langle p, x_p\rangle$, where $p \in PT$ and $x_p \in max(p)$ whenever p is non-empty and $x_p = \bullet$ otherwise. We use $*$ as a wildcard character.

The tree is recursively built as follows. The root node is labeled $\langle \emptyset, \bullet\rangle$. A node labeled $\langle p, *\rangle$ has an arc to a node labeled with some valid pair $\langle p', x_{p'}\rangle$ iff $p' - x_{p'} = p$. Since PT is prefix-closed, it is easy to see inductively that there is some $\langle p, *\rangle$-labeled node in the tree for every $p \in PT$. Furthermore, it is easy to see that the tree is finite, and hence is also finitely-branching. Since every pomset implicitly represents its isomorphism class, we assume without loss of generality that any two children of any given node of the tree will have distinct second components in their labels, *i.e.*, if the labels of the children are respectively $\langle *, x\rangle$ and $\langle *, y\rangle$, then x and y are distinct symbols.

From this tree, we will recursively construct a loop-free net N in which every place has in-degree of at most one. This net preserves concurrency of events occurring within any branch of the tree; however, events on different branches of the tree will always be represented as conflicting transitions.

We now recursively construct the following net from the tree, level by level. For the first level, we begin with a net with one place, which is initially marked. For every child s of the root, we add to the net a new $l_p(x)$-labeled transition, named s, where the label of node s in the tree is $\langle p, x\rangle$. The single initially marked place is attached as the pre-set of each transition s. All of these transitions have empty post-sets.

For the induction step, we show how to define the $(k+1)$-level of the tree from the k-level segment of the tree. For every node s in the k^{th} level of the tree and each child s' of node s, we construct a new $l_{p'}(x')$-labeled transition, named s', where the label of node s' in the tree is $\langle p', x'\rangle$.

It is easy to see that for every maximal cause $x \in \text{Events}_{p'}$ of x', there is a unique $\langle *, x\rangle$-labeled node s'' along the path from the root to s'. We then hook up transition s' to the (already existing) transition s''. This is accomplished by creating a new, unmarked place for transition s'' and adding it both to the post-set of transition s'' and to the pre-set of transition s'. If $x' \in min(p')$, then a new initially marked place is added to the net and is attached as the preset of transition s'. Finally, transition s' is placed in conflict with every transition v in the net such that node v is not a predecessor of s' in the tree. This is accomplished by creating a new, initially marked place for every such transition v, and putting this new place in the pre-sets of both transition s' and v.

It is then straightforward to show inductively that PT is the set of pomset-traces of the net; the details are left to the reader.

Let Act be the alphabet of the net; clearly, all transitions of the net have labels from Act. Since the original tree is finite, an inductive argument shows that the net is 1-safe, has a finite number of initially marked places, and that all places and transitions have finite in-degree and out-degree. Thus, only a finite number of transitions are enabled under any reachable marking of the net. However, one complication is that the $\surd$-labeled transitions of N may not clean out all the tokens in the net. To correct this, we first recall that any pomset in PT can contain at most one $\surd$-labeled event, which must be the sole maximum event in the pomset. We then observe that any $\surd$-labeled transition, s, in the net, is thus enabled only after firing exactly the transitions corresponding to each of the predecessors of node s in the tree. By construction of the net, these transitions can only be fired in a sequence that is consistent with the ordering of the pomset corresponding to the node s. By Theorem 3.2.3, every such firing sequence results in the same final marking; thus, there is exactly one reachable marking of the net under which transition s is enabled. A simple modification of the preset of each $\surd$-labeled transition to include all such corresponding marked places then yields the desired WT Net. ∎

We now show that all compact elements are definable as the $[\![\cdot]\!]^{\text{MUST}}$ meanings of WT Nets:

Theorem 4.2.15 Let Act be a finite set of labels containing the distinguished symbol $\surd$. For every compact element $\langle PF, PD, Act\rangle \in \mathcal{D}^{\text{MUST}}_{Act}$, there is some WT Net $\langle N, Act\rangle$ with $[\![\langle N, Act\rangle]\!]^{\text{MUST}} = \langle PF, PD, Act\rangle$.

Proof. Let $\langle PF, PD, Act\rangle$ be a compact element of $\mathcal{D}^{\text{MUST}}_{Act}$. As a simple consequence of Lemma 4.2.12 and the definition of compactness (*cf.* [18]), $\langle PF, PD, Act\rangle$ is a finite candidate of $\mathcal{D}^{\text{MUST}}_{Act}$. By Definition 4.2.10, Act is a finite set of labels and there is some *finite* set $\mathcal{PF}_{\text{fin}}$ of pomset-failures over Act and some *finite* set $\mathcal{PD}_{\text{fin}}$ of pomset-divergences over Act such that:

- $\langle \mathcal{PF}_{\text{fin}}, \emptyset, Act\rangle \in \mathcal{D}^{\text{MUST}}_{Act}$.
- $\mathcal{PD}_{\text{fin}}$ is prefix-closed.
- $\langle p, D\rangle \in \mathcal{PD}_{\text{fin}}$ implies that $\langle p, \emptyset\rangle \in \mathcal{PF}_{\text{fin}}$.
- $PD = \mathit{augment}(\mathit{extend}_{Act}(\mathcal{PD}_{\text{fin}}))$.
- $PF = \mathcal{PF}_{\text{fin}} \cup \mathit{implied\text{-}failures}_{Act}(PD)$.

We first build a tree whose nodes are labeled with *valid triples* of the form $\langle p, D_p, x_p\rangle$ or $\langle p, F_p, x_p\rangle$, where $\langle p, D_p\rangle \in \mathcal{PD}_{\text{fin}}$ or $\langle p, F_p\rangle \in \mathcal{PF}_{\text{fin}}$, respectively, and $x_p \in \mathit{max}(p)$ whenever p is non-empty and $x_p = \bullet$ otherwise. Furthermore, we require that $F_p = F' - A$ for some F' and A such that $\langle p, F'\rangle \in \mathcal{PF}_{\text{fin}}$, $\langle p, F' \cup \{c\}\rangle \notin \mathcal{PF}_{\text{fin}}$ for all $c \in Act - F'$, $\emptyset \subseteq A \subseteq Act$, and for every $a \in A$, there is some $p'_a \in p$ *with* a such that $\langle p'_a, \emptyset\rangle \in \mathcal{PF}_{\text{fin}}$. We use $*$ as a wildcard character.

The tree is recursively built as follows. The root node is labeled $\langle \emptyset, Act - \mathit{init}(\mathcal{PF}_{\text{fin}}), \bullet\rangle$. A node v labeled $\langle p, F_p, x_p\rangle$ has an a-labeled arc to a node labeled with some valid triple $\langle p', F_{p'}, x_{p'}\rangle$ or $\langle p', D_{p'}, x_{p'}\rangle$ iff (i) $a \in Act - F_p$, (ii) $l_{p'}(x_{p'}) = a$, (iii) $p - x_{p'} = p$, and (iv) for every ancestor w of v, if $\langle q, F_q, *\rangle$ is the label of w, then either $\mathit{down}_{p'}(x_{p'}) \not\subseteq q$ or $a \notin F_q$.

We first show that for every node labeled $\langle p, F_p, x_p\rangle$, there is an a-labeled arc emanating from the node iff $a \in Act - F_p$. One direction follows immediately from the construction of the tree. For the other direction, let $a \in Act - F_p$, then by definition of the valid triples and the closure properties of $\mathcal{PF}_{\text{fin}}$, it is straightforward to show that $\langle p; a, \emptyset\rangle \in \mathcal{PF}_{\text{fin}}$. Let x_a be the (unique) maximum event of $p; a$; it is then easy to see that there must be an a-labeled arc from the $\langle p, F_p, x_p\rangle$-labeled node to a $\langle p; a, F', x_a\rangle$-labeled node for some F'.

We now show that for every $\langle p, D_p\rangle \in \mathcal{PD}_{\text{fin}}$, there is some node in the tree with label $\langle p, D_p, x_p\rangle$, and for every $\langle p, F\rangle \in \mathcal{PF}_{\text{fin}}$, there is some node in the tree with label $\langle p, F_p, x_p\rangle$ for some $F_p \supseteq F$. We prove the lemma by induction on the size of p. The base case of $p = \emptyset$ follows easily from the closure properties of $\mathcal{PF}_{\text{fin}}$. For one case of the induction step, let $\langle p, F\rangle \in \mathcal{PF}_{\text{fin}}$, let $x_p \in max(p)$, let a be the label of x_p, and choose some $F_p \supseteq F$ with $\langle p, F_p\rangle \in \mathcal{PF}_{\text{fin}}$ such that $\langle p, F_p \cup \{c\}\rangle \notin \mathcal{PF}_{\text{fin}}$ for all $c \in Act - F_p$. Since $\mathcal{PF}_{\text{fin}}$ is prefix-closed, there is by induction some node v in the tree with label $\langle r, F_r, x_p\rangle$, where $r = p - x_p$. From the construction of the tree, we can assume without loss of generality that $a \notin F_r$. If some $\langle q_i, F_{q_i}, x_{q_i}\rangle$-labeled node is an ancestor of v with $a \in F_{q_i}$ and $down_p(x_p) \subseteq q_i$, let it be the least such ancestor. Let q_i' be p restricted to $q_i \cup \{x_p\}$; clearly, q_i' is a prefix of p, and hence $\langle q_i', \emptyset\rangle \in \mathcal{PF}_{\text{fin}}$. Thus, there is a $\langle q_i, F_{q_i} - \{a\}, x_{q_i}\rangle$-labeled node reachable by the same path as the $\langle q_i, F_{q_i}, x_{q_i}\rangle$-labeled node, and there is a path from this $\langle q_i, F_{q_i} - \{a\}, x_{q_i}\rangle$-labeled node to some $\langle r, F_r, x_r\rangle$-labeled node. By repeating this argument down the path to $\langle r, F_r, x_r\rangle$, it is easy to prove that there is a-labeled arc from the $\langle r, F_r, x_r\rangle$-labeled node to a $\langle p, F_p, x_p\rangle$-labeled node. The induction step for $\langle p, D_p, x_p\rangle$-labeled nodes is similar and is omitted.

Furthermore, it is easy to see that the tree is finitely branching. Since every pomset implicitly represents its isomorphism class, we assume without loss of generality that any two children of any given node of the tree will have distinct third components in their labels, *i.e.*, if the labels of the children are respectively $\langle *, *, x\rangle$ and $\langle *, *, y\rangle$, then x and y are distinct symbols.

From this tree, we will recursively construct a loop-free net N in which every place has in-degree of at most one. This net preserves concurrency within any branch of the tree; however, transitions arising from different branches on the tree will always be conflicting.

We now recursively construct the following net from the tree, level by level; the procedure is analogous to that in the proof of Theorem 4.2.14. For the first level, we begin with a net with one place, which is initially marked. For every child s of the root, we add to the net a new $l_p(x)$-labeled transition, named s, where the label of node s in the tree is $(p, *, x)$. The single initially marked place is attached as the pre-set of each transition s. All of these transitions have empty post-sets.

For the induction step, we show how to define the $(k+1)$-level of the tree from the k-level segment of the tree. For every node s in the k^{th} level of the tree and each child s' of node s, we construct a new $l_{p'}(x')$-labeled transition, named s', where the label of node s' in the tree is $(p', *, x')$. It is easy to see that for every maximal cause $x \in \text{Events}_{p'}$ of x', there is a unique $\langle *, *, x\rangle$-labeled node s'' along the path from the root to s'. We then hook up transition s' to the (already existing) transition s''. This is accomplished by creating a new, unmarked place for transition s'' and adding it both to the post-set of transition s'' and to the pre-set of transition s'. If $x' \in min(p')$, then a new initially marked place is added to the net and is attached as the preset of transition s'. Finally, transition s' is placed in conflict with every transition v in the net such that node v is not a predecessor of s' in the tree. This is accomplished by creating a new, initially marked place for every such transition v, and putting this new place in the pre-sets of both transition s' and v.

The procedure is analogous for every node s' labeled with some $\langle p', D_{p'}, x_{p'}\rangle$, except that in addition, a new divergence (*i.e.*, a τ-transition in a self-loop) corresponding to each $d \in D_{p'}$ is hooked up in the obvious analogous manner.

It is then straightforward to show inductively that for any $\langle p, F_p, x_p\rangle$-labeled node in the tree, there is a pomset-trace corresponding to p after which exactly the actions in $Act - F_p$ are enabled. Thus, all pomset-failures in $\mathcal{PF}_{\text{fin}}$ are actual pomset-failures of the net. Furthermore, $\mathcal{PD}_{\text{fin}}$ is exactly the set of pomset-divergences of the net.

However, for some $\langle p, F_p, x_p\rangle$-labeled nodes or some $\langle p, D_p, x_p\rangle$-labeled nodes, the marking of the net reached after firing some proper *prefix* of p may generate a failure set that is "too big." We thus patch up the net by iterating the following procedure: pick some $\langle p, F_p, x_p\rangle$-labeled node or some $\langle p, D_p, x_p\rangle$-labeled node, some proper prefix q of p, and some branch of the tree starting at some $\langle q, F_q, x_q\rangle$-labeled node, and prune this branch so that each of its nodes are labeled with $\langle r, F_r, x_r\rangle$ for some r such that $down_r(x) \supseteq q$ for all $x \in r - q$. It is straightforward to show that there exists some such pruned branch whose leaves are leaves of the original tree. Now, for every occurrence of q in the net that has an incorrect failure set, add transitions following q so that it emulates the pruned branch. It is straightforward to show firing q in the resulting net always leads to a correct failure set and that the failure sets of the markings corresponding to the nodes of the tree are unaffected. Each iteration reduces the number of distinct pomsets q with incorrect failure sets, and hence $\mathcal{PF}_{\text{fin}}$ and $\mathcal{PD}_{\text{fin}}$ are respectively exactly the pomset-failures and pomset-divergences generated by this net.

An inductive argument then shows that the net is 1-safe, has a finite number of initially marked places, and that all places and transitions have finite in-degree and out-degree. Thus, the labeled transition system of the net is finitely branching. However, one complication is that the $\surd$-labeled transitions of N may not clean out all the tokens in the net; this difficulty is resolved exactly as in the proof of Theorem 4.2.14. It is then easy to show that the resulting net is a WT Net $\langle N, Act\rangle$ such that $[\![\langle N, Act\rangle]\!]^{\text{MUST}} = \langle PF, PD, Act\rangle$. ∎

We now show that all compact elements are definable as the $[\![\cdot]\!]^{\text{TEST}}$ meanings of WT Nets:

Theorem 4.2.16 Let Act be a finite set of labels containing the distinguished symbol $\surd$. For every compact element $\langle\langle PT, Act\rangle, \langle PF, PD, Act\rangle\rangle \in \mathcal{D}^{\text{TEST}}_{Act}$, there is some WT Net $\langle N, Act\rangle$ with $[\![\langle N, Act\rangle]\!]^{\text{TEST}} = \langle\langle PT, Act\rangle, \langle PF, PD, Act\rangle\rangle$.

Proof. Let $\langle\langle PT, Act\rangle, \langle PF, PD, Act\rangle\rangle$ be a compact element of $\mathcal{D}^{\text{TEST}}_{Act}$. As a simple consequence of Lemma 4.2.12 and the definition of compactness (*cf.* [18]), $\langle\langle PT, Act\rangle, \langle PF, PD, Act\rangle\rangle$ is a finite candidate of $\mathcal{D}^{\text{TEST}}_{Act}$. Thus, $\langle PT, Act\rangle$ is a finite candidate of $\mathcal{D}^{\text{MAY}}_{Act}$ and $\langle PF, PD, Act\rangle$ is a finite candidate of $\mathcal{D}^{\text{MUST}}_{Act}$. Let $\mathcal{PF}_{\text{fin}}, \mathcal{PD}_{\text{fin}}$ be the finite generating sets of PF, PD as given by Definition 4.2.10.

Using the same technique as in the proof of Theorem 4.2.15, we first build a tree whose nodes are labeled with *valid triples* over $\mathcal{PF}_{\text{fin}}$ and $augment(\mathcal{PD}_{\text{fin}})$, rather than $\mathcal{PD}_{\text{fin}}$. In addition, nodes can also be labeled with *valid pairs* $\langle p, x_p\rangle$, where $p \in PT$, $\langle p, \emptyset\rangle \notin \mathcal{PF}_{\text{fin}}$, and $x_p \in max(p)$. The nodes labeled with valid pairs are connected up as follows. A node labeled $\langle p, D_p, x_p\rangle$ has an a-labeled arc to a node labeled with some valid pair $\langle p', x_{p'}\rangle$ iff $\langle p, D_p\rangle \sqsubseteq \langle p', \{p'\}\rangle$, $p' - x_{p'} = p$, and $l_{p'}(x_{p'}) = a$. Finally, a node labeled $\langle p, x_p\rangle$ has an a-labeled arc to a node labeled with some valid pair $\langle p', x_{p'}\rangle$ iff $p' - x_{p'} = p$ and $l_{p'}(x_{p'}) = a$.

By the closure conditions of Definition 4.2.7, $\langle p, \emptyset\rangle \in PF$ for every $p \in PT$. Hence, $\langle p, \emptyset\rangle \notin \mathcal{PF}_{\text{fin}}$ for some $p \in PT$, then there must be some $\langle r, D_r\rangle \in \mathcal{PD}_{\text{fin}}$ such that $\langle p, \{\text{Events}_p\}\rangle \in augment(extend_{Act}(\langle r, D_r\rangle))$. Thus, $\langle p, F\rangle \in PF$ for every $F \subseteq Act$. By Proposition 3.2.15 and the prefix-closure of $\mathcal{PD}_{\text{fin}}$, there must be some $\langle r', D_{r'}\rangle \in \mathit{augment}(\mathcal{PD}_{\text{fin}})$ with $\langle r', D_{r'}\rangle \sqsubseteq \langle p, \{\text{Events}_p\}\rangle$. It is then easy to show inductively that for every $p \in PT$, the tree contains some node labeled with either $\langle p, *, *\rangle$ or $\langle *, *\rangle$, where $*$ is the wildcard symbol.

The construction of the net is then the obvious straightforward combination of the constructions in the proofs of Theorem 4.2.14 and Theorem 4.2.15, as is the remainder of the proof. ∎

The operations on $\mathcal{D}^{\text{MAY}}_{Act}$ and $\mathcal{D}^{\text{MUST}}_{Act}$ are given in Definition 3.2.28 and Definition 3.2.29, respectively. We do not restate the definitions here. Let the operations on $\mathcal{D}^{\text{TEST}}_{Act}$ be the natural pairwise combination of the operations on $\mathcal{D}^{\text{MAY}}_{Act}$ and $\mathcal{D}^{\text{MUST}}_{Act}$. Then:

Theorem 4.2.17 Let Act be a finite set of labels containing the distinguished symbol $\surd$. Then $\mathcal{D}^{\text{MAY}}_{Act}$, $\mathcal{D}^{\text{MUST}}_{Act}$, and $\mathcal{D}^{\text{TEST}}_{Act}$ are closed under prefixing, restriction, renaming, hiding, sequencing, internal choice, CCS choice, non-communicating parallel composition, CSP-style parallel composition, CCS-style parallel composition, split refinements, and choice refinements. Furthermore, all of these operations are continuous functions on the respective domains.

Let Act' be a finite set of labels containing $\surd$. Then *grow* Act' and *shrink* Act' are continuous functions from $\langle \mathcal{D}^{\text{MAY}}_{Act}, \sqsubseteq^{\text{MAY}}_{Act}\rangle$, $\langle \mathcal{D}^{\text{MUST}}_{Act}, \sqsubseteq^{\text{MUST}}_{Act}\rangle$, and $\langle \mathcal{D}^{\text{TEST}}_{Act}, \sqsubseteq^{\text{TEST}}_{Act}\rangle$ to $\langle \mathcal{D}^{\text{MAY}}_{Act'}, \sqsubseteq^{\text{MAY}}_{Act'}\rangle$, $\langle \mathcal{D}^{\text{MUST}}_{Act'}, \sqsubseteq^{\text{MUST}}_{Act'}\rangle$, and $\langle \mathcal{D}^{\text{TEST}}_{Act'}, \sqsubseteq^{\text{TEST}}_{Act'}\rangle$, respectively.

The proof for $\mathcal{D}^{\text{MAY}}_{Act}$ is completely routine but tedious and is left to the reader.

It is routine but tedious to verify that $\mathcal{D}^{\text{MUST}}_{Act}$ is closed under all of the operations except alphabet expansion and shrinking, *grow* Act' and *shrink* Act' are continuous functions from $\langle \mathcal{D}^{\text{MUST}}_{Act}, \sqsubseteq^{\text{MUST}}_{Act}\rangle$ to $\langle \mathcal{D}^{\text{MUST}}_{Act'}, \sqsubseteq^{\text{MUST}}_{Act'}\rangle$, all of the operations are monotone, and that all the operations *except* hiding and CCS-style parallel composition are continuous. The details are left to the reader. The proof for $\mathcal{D}^{\text{TEST}}_{Act}$ is a simple consequence of Definition 4.2.7 and the continuity of the operations on $\mathcal{D}^{\text{MAY}}_{Act}$ and $\mathcal{D}^{\text{MUST}}_{Act}$.

We prove the case for hiding on $\mathcal{D}^{\text{MUST}}_{Act}$, from which the proof for CCS-style parallel composition follows easily. The proof for hiding is a generalization of that in [8] for failures semantics, and uses the following lemma:

Lemma 4.2.18 Let $\langle q, D_q\rangle$ be a pomset-divergence over a finite alphabet Act, let $a \in Act$, and let $PDS = \{\langle p_n, D_n\rangle : n \geq 0\}$ be an infinite set of pomset-divergences such that $\langle p_n, D_n\rangle - a = \langle q, D_q\rangle$ for all $\langle p_n, D_n\rangle \in PDS$. Then there is some pomset-divergence $\langle r_0, D \cup R\rangle$ with $\langle r_0, D \cup R\rangle - a \sqsubseteq \langle q, D_q\rangle$ and some infinite sequence $r_1, r_2 \ldots$ of pomsets such that for $i \geq 0$:

- r_i is a prefix of r_{i+1}.
- All events in $r_i - r_0$ are a-labeled.
- For every $d \in R$, r_i contains an i-length chain of a-labeled events whose downward-closure restricted to r_0 is a subset of d.
- $\langle r_i, D\rangle$ is a prefix of some $\langle p_{n_i}, D_{n_i}\rangle \in PDS$.

Proof. Let $\langle r_0, D\rangle$ be a pair consisting of a pomset, r_0, together with a possibly empty set, D, of its prefixes such that

- $\langle r_0, D\rangle$ is a prefix of an infinite number of pomset-divergences $\langle p_{n_i}, D_{n_i}\rangle \in PDS$.

- For every pair $\langle r_0', D'\rangle$ that is a prefix of an infinite number of pomset-divergences in PDS, if $\langle r_0, D\rangle - a$ is a prefix of $\langle r_0', D'\rangle - a$, then $\langle r_0, D\rangle - a$ and $\langle r_0', D'\rangle - a$ are isomorphic.

Clearly, $\langle\emptyset, \emptyset\rangle$ is a prefix of every pomset-divergence. Since the size of all such $\langle p_0', D'\rangle - a$ is bounded by $\langle q, D_q\rangle$ it is easy to see that a pomset-divergence $\langle p_0, D\rangle$ exists that satisfies the above conditions; however, it is not necessarily unique.

Let $\langle q', D_{q'}\rangle = \langle r_0, D\rangle - a$ for some such pair $\langle r_0, D\rangle$; clearly, $\langle q', D_{q'}\rangle$ is a prefix of $\langle q, D_q\rangle$. Let R be the following set of downward-closed subsets of r_0:

$$\begin{aligned} R_1 &= \{down_{r_0}(d) : d \in D_q,\ down_{r_0}(d) \notin D, \text{ and } d \subseteq q'\} \\ R_2 &= \{down_{r_0}(down_q(x) \cap q') : x \in q - q'\} \\ R &= R_1 \cup R_2 \end{aligned}$$

It is straightforward to show that the maximality conditions of $\langle r_0, D\rangle$ imply that $\langle r_0, D\rangle$ is extended in PDS by concurrent chains of a-labeled events of unbounded length in PDS, and whose set of downward-closures is exactly R. It is also easy to see that $\langle r_0, D \cup R\rangle - a \sqsubseteq \langle q, D_q\rangle$, proving the lemma. ∎

We now prove the continuity of the hiding operation on $\mathcal{D}_{Act}^{\mathrm{MUST}}$:

Lemma 4.2.19 Let Act be a finite set of labels containing the distinguished symbol $\surd$. Then the hiding operation on $\mathcal{D}_{Act}^{\mathrm{MUST}}$ is continuous.

Proof. One direction follows immediately from the easy observation that hiding is a monotone function. For the other direction, let A be an infinite chain in $\mathcal{D}_{Act}^{\mathrm{MUST}}$, let $\{\langle PF_k, PD_k, Act\rangle : k \in I_A\} \stackrel{\text{def}}{=} A$ for some index set I_A, and let $\langle PF_A, PD_A, Act\rangle = \bigsqcup A$. For one case, let $\langle q, D_q\rangle \in \bigcap\{PD_k - a : k \in I_A\}$. Lemma 3.2.15 and the closure properties of the PD_k imply that for every PD_k with $k \in I_A$, there is some $\langle p^k, D^k\rangle \in PD_k \cup PF_k$ and some possibly empty set R^k of downward-closed subsets of Events_{p^k} such that $\langle p^k, D^k \cup R^k\rangle$ is a pomset-divergence, $\langle p^k, D^k \cup R^k\rangle - a \sqsubseteq \langle q, D_q\rangle$, and for all $n \geq 0$, there is some p_n^k with $\langle p_n^k, D\rangle \in PD_k \cup PF_k$ such that:

- $\langle p^k, R^k\rangle \sqsubseteq \langle p_n^k, \{\mathrm{Events}_{p_n^k}\}\rangle$.
- All events in $p_n^k - p^k$ are a-labeled.
- For every $d \in R^k$, there is some n-length chain of a-labeled events in $p_n^k - p^k$ whose downward closure restricted to p^k is d.

Furthermore, we can clearly assume without loss of generality that every p_n^k has size bounded by $|p^k| + |R^k| \times n$.

Let PDS be the set $\{\langle p^k, D^k \cup R^k\rangle : k \in I_A\}$. If PDS is finite, then it is easy to see that an infinite number of $\langle PT_k, PF_k, PD_k, Act\rangle \in A$ have the same $\langle p^k, D^k, R^k\rangle$ and the same sequence $p_1^k, p_2^k, \ldots$. Since A is a chain, this $\langle p^k, D^k, R^k\rangle$ and this sequence $p_1^k, p_2^k, \ldots$ must occur in every element of A, from which it follows easily that $\langle q, D_q\rangle \in PD_A$. If PDS is infinite, then clearly there must be some infinite subset PDS' of PDS such that for all $\langle p^i, D^i \cup R^i\rangle, \langle p^j, D^j \cup R^j\rangle \in PDS$, $\langle p^i, D^i \cup R^i\rangle - a = \langle p^j, D^j \cup R^j\rangle - a$. Thus, Lemma 4.2.18 gives the existence of a prefix $\langle r_0, D\rangle$ of some $\langle p^k, D^k \cup R^k\rangle \in PDS$, some set R of prefixes of r_0 with $\langle r_0, D \cup R\rangle - a \sqsubseteq \langle q, D_q\rangle$, and some appropriate sequence $r_1, r_2 \ldots$ such that every $\langle r_i, D\rangle$ is a prefix of some $\langle p^i, D^i \cup R^i\rangle \in PDS$. Thus it follows by closure properties of the $\langle PF_i, PD_i, Act\rangle$ that $\langle r_i, D\rangle \in PD_i \cup PF_i$. The definition of PDS and the chain condition then immediately implies that $\langle r_0, D\rangle$ and all of the $\langle r_i, D\rangle$ are in $PD_m \cup PF_m$ for every $\langle PF_m, PD_m, Act\rangle$, from which it follows easily that $\langle q, D_q\rangle \in PD_A$.

The proof that $\bigcap\{PF_k - a : k \in I_A\} \subseteq PF_A$ is very similar and is left to the reader. ■

4.3 The Split Semantics

This section gives abstract characterizations of the $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$ and $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ semantics on WT Nets, shows that they form algebraic cpo's, and proves that all the corresponding process operations from Chapter 3 are continuous functions on these cpo's.

We define $\langle \mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}, \sqsubseteq^{\text{MUST-split-}\gamma}_{Act'}\rangle$ as a sub-partial-order of $\langle \mathcal{D}^{\text{MUST}}_{Act}, \sqsubseteq^{\text{MUST}}_{Act}\rangle$ corresponding to $[\![\cdot]\!]^{\text{MUST}}$ meanings of $\gamma +_M$ *dupl-split* nets. In order to ensure that every compact element of $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ is definable as the $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$ meaning of some WT Net, we require that $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ satisfy some additional closure conditions.

First, we must ensure that Act is a "dupl-split alphabet," and that PF and PD are closed under "0-splitting" any a_0-labeled events. Dually, any minimal pomset-failure or pomset-divergence must be the result of "0-splitting" some *1-2-respect*ing pomset. We note that the definition of *1-2-respect*ing ensures that no a_1-labeled event must be a maximal cause of any divergence. Furthermore, any maximal a_1-labeled events corresponds to "half-fired" a_0-events and hence can be relabeled with a_0. Also, firing any a_1-labeled event additionally enables only a a_2-labeled event. The special role of $\surd$ and γ is also reflected in the closure conditions. In particular, (1e) reflects the presence of initial τ-moves.

Definition 4.3.1 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. A triple $\langle PF, PD, Act'\rangle$ is said to be MUST-split-respecting iff it is a MUST-respecting triple and satisfies the following properties:

1. Additional closure properties of PF:
   (a) *0-split*$(PF) \subseteq PF$.
   (b) $\langle p, F\rangle \in$ *1-2-respect*(PF) and $p' \in \sigma(p)$ implies that $\langle p', \emptyset\rangle \in PF$, where σ is the sequence of choice refinements $\langle choice_{(a_1,a_1,a_0)} : a \in Act - \{\gamma, \surd\}\rangle$.
   (c) $\langle p, F\rangle \in PF$, $c \in Act - \{\gamma, \surd\}$, and $\langle p, F \cup \{c_1\}\rangle \notin PF$ implies that there is some $p' \in p$ *with* c_1 such that $\langle p', F - \{c_2\}\rangle \in PF$.
   (d) $\langle \gamma, Act\rangle \in PF$.
   (e) $\langle \emptyset, F \cup \{a\}\rangle \in PF$ and $\langle a, \emptyset\rangle \in PF$ implies that $\langle \emptyset, F \cup \{a, \gamma\}\rangle \in PF$.
2. Additional closure properties of PD:
   (a) $\langle p, D\rangle \in min_{\sqsubseteq}(PD)$ implies that $\langle p, D\rangle \in$ *augment*(*0-split*(*1-2-respect*(PD))).
   (b) *0-split*$(PD) \subseteq PD$.

(c) $\langle p, D\rangle \in$ *1-2-respect*(PD) and $p' \in \sigma(p)$ implies that $\langle p', D\rangle \in PD$, where σ is the sequence of choice refinements $\langle choice_{(a_1,a_1,a_0)} : a \in Act - \{\gamma, \surd\}\rangle$.

(d) $\langle p, D\rangle \in min_{\sqsubseteq}(PD)$ implies that p contains no γ-labeled events.

3. Additional mixed properties:

(a) $\langle p, F\rangle \in PF$ and $\langle p, \{\text{Events}_p\}\rangle \notin PD$ implies that $\langle p, F\rangle \in augment(\mathit{0\text{-}split}(\mathit{1\text{-}2\text{-}respect}(PF)))$

(b) $\langle p, F\rangle \in PF$ and $\langle p, \{\text{Events}_p\}\rangle \notin PD$ and p contains a $\surd$-labeled event implies that this is the sole event in p.

(c) $\langle p, F\rangle \in PF$, $\langle p, \{\text{Events}_p\}\rangle \notin PD$, and $F \cap \{a_0, a_1\} \neq \emptyset$ for some $a \in Act - \{\gamma, \surd\}$ implies that $F \supseteq \{a_0, a_1\}$.

(d) $\langle p, F\rangle \in$ *1-2-respect*(PF), $\langle p, \{\text{Events}_p\}\rangle \notin PD$, and $a_2 \in F$ for some $a \in Act - \{\gamma, \surd\}$ implies that no event in p is a_1-labeled.

Definition 4.3.2 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. A pair $\langle\langle PT, Act\rangle, \langle PF, PD, Act'\rangle\rangle$ said to be TEST-split-respecting iff $\langle PT, Act\rangle$ is MAY-respecting , $\langle PF, PD, Act'\rangle$ is MUST-split-respecting , and

1. $p \in PT$ and $p' \in \sigma(\delta(p))$ implies that $\langle p', \emptyset\rangle \in PF$, where δ is the sequence of choice refinements $\langle choice_{(a,a_0,a')} : a \in Act - \{\gamma, \surd\}\rangle$, σ is the sequence of split refinements $\langle split_{(a',a_1,a_2)} : a \in Act - \{\gamma, \surd\}\rangle$, and all the a' are distinct symbols not in $Act \cup Act'$.

2. $\langle p, F\rangle \in PF$ and $\langle p, \{\text{Events}_p\}\rangle \notin PD$ implies that there is some $p' \in PT$ such that $p \in augment(\sigma(\delta(p')))$, where σ and δ are as above.

3. $\langle p, D\rangle \in min_{\sqsubseteq}(PD)$ implies that there is some $p' \in PT$ such that $p \in augment(\sigma(\delta(p')))$, where σ and δ are as above.

Definition 4.3.3 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ is the restriction of $\mathcal{D}^{\text{MUST}}_{Act'}$ to MUST-split-respecting triples and $\sqsubseteq^{\text{MUST-split-}\gamma}_{Act'}$ is the restriction of $\sqsubseteq^{\text{MUST}}_{Act'}$ to $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'} \times \mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$.

Definition 4.3.4 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then $\mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$ is defined to be the set of all TEST-split-respecting pairs $\langle\langle PT, Act\rangle, \langle PF, PD, Act'\rangle\rangle$. Furthermore, $\sqsubseteq^{\text{TEST-split-}\gamma}_{Act,Act'}$ is the binary relation on $\mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$ such that for every $\langle\alpha_1, \beta_1\rangle$ and $\langle\alpha_2, \beta_2\rangle$ in $\mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$, $\langle\alpha_1, \beta_1\rangle \sqsubseteq^{\text{TEST-split-}\gamma}_{Act,Act'} \langle\alpha_2, \beta_2\rangle$ iff $\alpha_1 \sqsubseteq^{\text{MAY}}_{Act} \alpha_2$ and $\beta_1 \sqsubseteq^{\text{MUST-split-}\gamma}_{Act'} \beta_2$.

We first show that $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$ and $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ map WT Nets to elements of these domains:

Theorem 4.3.5 Let $\langle N, Act\rangle$ be a WT Net, and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\gamma, \surd\}$. Then $[\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} \in \mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ and $[\![\langle N, Act\rangle]\!]^{\text{TEST}}_{\text{split-}\gamma} \in \mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$.

Proof. We first prove the case for $[\![\cdot]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$. Since WT Nets are closed under $+_M$ and *dupl-split*, the definition of $[\![\cdot]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$ and Theorem 4.2.8 together imply that $[\![\langle N, Act\rangle]\!]^{\mathrm{MUST}}_{\text{split-}\gamma} \in \mathcal{D}^{\mathrm{MUST}}_{Act'}$. The additional closure conditions of Definition 4.3.3 follow directly from the properties of $\gamma{+}_M$ and *dupl-split* nets, and are easy to verify. The details are left to the reader.

For $[\![\cdot]\!]^{\mathrm{TEST}}_{\text{split-}\gamma}$, it is straightforward to see from the definitions of pomset-traces, pomset-failures, and pomset-divergences of WT Nets, the definition of $[\![\cdot]\!]^{\mathrm{TEST}}_{\text{split-}\gamma}$, the definition of *dupl-split* and $\gamma{+}_M$, and Proposition 3.2.15 that the additional closure conditions hold. The theorem then follows easily from Theorem 4.2.8 and the above case. ∎

Theorem 4.3.6 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then $\langle \mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}, \sqsubseteq^{\text{MUST-split-}\gamma}_{Act'}\rangle$ and $\langle \mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}, \sqsubseteq^{\text{TEST-split-}\gamma}_{Act,Act'}\rangle$ are complete partial orders.

The proof of the theorem is an easy combination and adaptation of the proof of Theorem 4.2.9. The details are left to the reader.

We now give a finite characterization of the compact elements of $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ and $\mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$.

Definition 4.3.7 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. A triple $\langle PF, PD, Act'\rangle$ is a *finite candidate* of $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ iff $\langle PF, PD, Act'\rangle \in \mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ and $\langle PF, PD, Act'\rangle$ is a finite candidate of $\mathcal{D}^{\mathrm{MUST}}_{Act'}$. A tuple $\langle \alpha, \beta\rangle$ is a *finite candidate* of $\mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$ iff $\langle \alpha, \beta\rangle \in \mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$, α is a finite candidate of $\mathcal{D}^{\mathrm{MAY}}_{Act}$, and β is a finite candidate of $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$.

As an immediate consequence of Definitions 4.3.3 and 4.3.4 and Lemma 4.2.11, we have:

Lemma 4.3.8 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then all finite candidates of $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ and $\mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$ are compact elements of $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ and $\mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$, respectively.

We now show:

Lemma 4.3.9 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. For every $\langle PF, PD, Act'\rangle \in \mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$, there is a directed set $A_1 \subseteq \mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ of finite candidates with $\bigsqcup A_1 = \langle PF, PD, Act'\rangle$. For every $\langle \alpha, \beta\rangle \in \mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$, there is a directed set $A_2 \subseteq \mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$ of finite candidates with $\bigsqcup A_2 = \langle \alpha, \beta\rangle$.

Proof. The proof for $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ is a minor modification of that of Theorem 4.2.12, needed in order to ensure that closure condition (2a) of Definition 4.3.3 holds for every approximation. For every $n \geq 0$, we define n^{th} approximation, $\langle PF_n, PD_n, Act'\rangle$ to $\langle PF, PD, Act'\rangle$ as follows:

$$\begin{aligned} PF' = \{\langle p, F\rangle \in PF : {}& p \text{ is a prefix of some } q \text{ such that } \langle q, \emptyset\rangle \in PF, \\ & \text{and either } \langle q, \{\mathrm{Events}_q\}\rangle \notin PD \text{ or } \langle q, D\rangle \in \mathit{min}_{\sqsubseteq}(PD) \text{ for some } D\} \end{aligned}$$

$$\mathcal{PF}'_{\text{fin-}n} = \{\langle p, F\rangle \in PF' : \mathit{depth}(p) \leq n\} \cup \{\langle p, F\rangle : \langle p, \emptyset\rangle \in PF',\ \mathit{depth}(p) = n,\ \text{and } F \subseteq Act'\}$$

$$\begin{aligned}\mathcal{PD}_{\text{fin-}n} = \{\langle p, D \cup D'\rangle : {} & \langle p, \emptyset\rangle \in \mathcal{PF}'_{\text{fin-}n},\ D \cup D' \neq \emptyset,\ \text{ either } \langle p, D\rangle \in PD \text{ or } D = \emptyset,\\ & \text{and for all } d \in D',\ \text{either } d = \mathit{down}_p(x) \cup \{x\} \text{ for some } x \text{ such that}\\ & \mathit{depth}_p(x) = n,\ \text{ and } l_p(x) \neq a_1 \text{ for any } a \in \mathit{Act} - \{\gamma, \surd\},\\ & \text{or } d = \mathit{down}_p(y) \cup \{y\} \text{ for some } y \text{ such that}\\ & y \text{ is a maximal cause of some } x \text{ with } \mathit{depth}_p(x) = n,\\ & \text{and } l_p(x) = b_1 \text{ for some } b \in \mathit{Act} - \{\gamma, \surd\}\}\end{aligned}$$

$$PD_n = \mathit{augment}(\mathit{extend}_{\mathit{Act}'}(\mathcal{PD}_{\text{fin-}n}))$$

$$\mathcal{PF}_{\text{fin-}n} = \mathit{augment}(\mathcal{PF}'_{\text{fin-}n})$$

$$PF_n = \mathcal{PF}_{\text{fin-}n} \cup \mathit{implied\text{-}failures}_{\mathit{Act}'}(PD_n)$$

The proof of this case is then a straightforward adaptation of that of Theorem 4.2.12; the details are left to the reader.

The proof of $\mathcal{D}^{\text{TEST-split-}\gamma}_{\mathit{Act},\mathit{Act}'}$ is a straightforward adaptation of the proof of Theorems 4.2.12 and the above case; the details are left to the reader. ∎

We now have:

Theorem 4.3.10 Let Act be a finite alphabet containing $\surd$ and let $\mathit{Act}' = \{a_0, a_1, a_2 : a \in \mathit{Act} - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then $\mathcal{D}^{\text{MUST-split-}\gamma}_{\mathit{Act}'}$ and $\mathcal{D}^{\text{TEST-split-}\gamma}_{\mathit{Act},\mathit{Act}'}$ are algebraic cpo's.

The theorem is a simple consequence of Lemma 4.3.8 and Lemma 4.3.9 (*cf.* [18]).

We now show that all compact elements are definable as the meanings of WT Nets:

Theorem 4.3.11 Let Act be a finite alphabet containing $\surd$ and let $\mathit{Act}' = \{a_0, a_1, a_2 : a \in \mathit{Act} - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. For every compact element $\langle PF, PD, \mathit{Act}'\rangle \in \mathcal{D}^{\text{MUST-split-}\gamma}_{\mathit{Act}'}$, there is some WT Net $\langle N_1, \mathit{Act}\rangle$ with $[\![\langle N_1, \mathit{Act}\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma} = \langle PF, PD, \mathit{Act}'\rangle$. For every compact element $\langle\langle PT, \mathit{Act}\rangle, \langle PF, PD, \mathit{Act}'\rangle\rangle \in \mathcal{D}^{\text{TEST-split-}\gamma}_{\mathit{Act},\mathit{Act}'}$, there is some WT Net $\langle N_2, \mathit{Act}\rangle$ with $[\![\langle N_2, \mathit{Act}\rangle]\!]^{\text{TEST}}_{\text{split-}\gamma}$ $= \langle\langle PT, \mathit{Act}\rangle, \langle PF, PD, \mathit{Act}'\rangle\rangle$.

Proof. For the first case, let $\langle PF, PD, \mathit{Act}'\rangle$ be a compact element of $\mathcal{D}^{\text{MUST}}_{\mathit{Act}}$. As a simple consequence of Lemma 4.3.9 and the definition of compactness (*cf.* [18]), $\langle PF, PD, \mathit{Act}'\rangle$ is a finite candidate of $\mathcal{D}^{\text{MUST-split-}\gamma}_{\mathit{Act}'}$.

The construction of the tree is analogous to the proof of the proof of Theorem 4.2.15, except that nodes are labeled with pomset-failures only from *0-split*(*1-2-respect*($\mathcal{PF}_{\text{fin}}$)) (rather than $\mathcal{PF}_{\text{fin}}$) and with pomset-divergences only from *0-split*(*1-2-respect*($\mathcal{PD}_{\text{fin}}$)) (rather than $\mathcal{PD}_{\text{fin}}$). Furthermore, the root has τ-labeled arcs to nodes labeled with valid triples of the form $\langle\emptyset, \{\emptyset\}, \bullet\rangle$ or $\langle\emptyset, F \cup \{\gamma\}, \bullet\rangle$. Finally, an additional restriction is that a node labeled $\langle p, F_p, x_p\rangle$ has an a_1-labeled arc to a node labeled with $\langle p', F_{p'}, x_{p'}\rangle$, then $F_p - \{a_2\} \subseteq F_{p'}$. The remainder of the proof is a straightforward modification of the proof of Theorem 4.2.15; the details are left to the reader. In particular, using closure condition (3a), it is easy to rewire the net to simulate duplicate-splitting.

We remark that the "patching-up" process is done first on prefixes whose maximal nodes are all α_0-labeled or α_2-labeled. The failure sets for the remaining prefixes are then chosen appropriately.

The resulting net is then isomorphic to $\gamma +_M (\textit{dupl-split}(\langle N_1, Act\rangle)\ \textit{grow}\ \{\gamma\})$ for some WT Net $\langle N_2, Act\rangle$, proving this case.

The proof for $\mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$ is then a straightforward combination of the proofs of Theorems 4.2.16 and the previous case; the details are left to the reader. ■

The operations on $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ are given in Definition 3.2.45. We do not restate the definitions here. Let the operations on $\mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$ be the natural pairwise combination of the operations on $\mathcal{D}^{\text{MAY}}_{Act}$ and $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$. Then:

Theorem 4.3.12 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ and $\mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}$ are closed under prefixing, restriction, renaming, hiding, sequencing, internal choice, CCS choice, non-communicating parallel composition, CSP-style parallel composition, CCS-style parallel composition, split refinements, and choice refinements. Furthermore, all of these operations are continuous functions on the respective domains.

Let Act_1 be a finite alphabet containing $\surd$ and let $Act_1{}' = \{a_0, a_1, a_2 : a \in Act_1 - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then *grow* Act_1 and *shrink* Act_1 are continuous functions from $\langle \mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}, \sqsubseteq^{\text{MUST-split-}\gamma}_{Act'}\rangle$ and $\langle \mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'}, \sqsubseteq^{\text{TEST-split-}\gamma}_{Act,Act'}\rangle$ to $\langle \mathcal{D}^{\text{MUST-split-}\gamma}_{Act_1{}'}, \sqsubseteq^{\text{MUST-split-}\gamma}_{Act_1{}'}\rangle$ and $\langle \mathcal{D}^{\text{TEST-split-}\gamma}_{Act_1,Act_1{}'}, \sqsubseteq^{\text{TEST-split-}\gamma}_{Act_1,Act_1{}'}\rangle$, respectively.

Proof. It is straightforward but tedious to show that $\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$ is closed under all of the operations except alphabet growing and shrinking, and that the domain and range of *grow* Act_1 and *shrink* Act_1 are as specified. It is easy to show that $+_M$, *augment*, *extend 1-2-respect*, *0-split*, 0-1-*choice*, and 0-1-*split* are continuous functions. The theorem then follows easily from Theorem 4.2.17. ■

4.4 The Interval Semantics

This section gives abstract characterizations of the $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$, $[\![\cdot]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$, and $[\![\cdot]\!]^{\text{TEST}}_{\text{intvl-}\gamma}$ semantics on WT Nets, shows that they form algebraic cpo's, and proves that all the corresponding process operations from Chapter 3 are continuous functions on these cpo's.

Definition 4.4.1 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then $\mathcal{D}^{\text{MAY-intvl}}_{Act}$ is defined to be the set $\textit{intervals}(\mathcal{D}^{\text{MAY}}_{Act})$. Furthermore, $\sqsubseteq^{\text{MAY-intvl}}_{Act}$ is the binary relation on $\mathcal{D}^{\text{MAY-intvl}}_{Act}$ such that for every $\langle PT_1, Act\rangle$ and $\langle PT_2, Act\rangle$ in $\mathcal{D}^{\text{MAY-intvl}}_{Act}$, $\langle PT_1, Act_1\rangle \sqsubseteq^{\text{MAY-intvl}}_{Act} \langle PT_2, Act_2\rangle$ iff $PT_1 \subseteq PT_2$.

$\mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}$ is defined to be the set $\textit{intervals}(\mathcal{D}^{\text{MUST-split-}\gamma}_{Act'})$. Furthermore, $\sqsubseteq^{\text{MUST-intvl-}\gamma}_{Act'}$ is the binary relation on $\mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}$ such that for every $\langle PF_1, PD_1, Act'\rangle$ and $\langle PF_2, PD_2, Act'\rangle$ in $\mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}$, $\langle PF_1, PD_1, Act'\rangle \sqsubseteq^{\text{MUST-intvl-}\gamma}_{Act'} \langle PF_2, PD_2, Act'\rangle$ iff $PF_1 \supseteq PF_2$ and $PD_1 \supseteq PD_2$.

$\mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}$ is defined to be the set $intervals(\mathcal{D}^{\text{TEST-split-}\gamma}_{Act,Act'})$. Furthermore, $\sqsubseteq^{\text{TEST-intvl-}\gamma}_{Act,Act'}$ is the binary relation on $\mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}$ such that for every $\langle\alpha_1,\beta_1\rangle$ and $\langle\alpha_2,\beta_2\rangle$ in $\mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}$, $\langle\alpha_1,\beta_1\rangle \sqsubseteq^{\text{TEST-intvl-}\gamma}_{Act,Act'} \langle\alpha_2,\beta_2\rangle$ iff $\alpha_1 \sqsubseteq^{\text{MAY-intvl}}_{Act} \alpha_2$ and $\beta_1 \sqsubseteq^{\text{MUST-intvl-}\gamma}_{Act'} \beta_2$.

We first show that the interval semantics of Chapter 3 map WT Nets to elements of the above partial orders:

Theorem 4.4.2 Let $\langle N, Act\rangle$ be a WT Net. Then $[\![\langle N, Act\rangle]\!]^{\text{MAY}}_{\text{intvl}} \in \mathcal{D}^{\text{MAY-intvl}}_{Act}$, $[\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} \in \mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}$, and $[\![\langle N, Act\rangle]\!]^{\text{TEST}}_{\text{intvl-}\gamma} \in \mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}$.

The theorem is a simple consequence of the definitions of $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$, $[\![\cdot]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$, $[\![\cdot]\!]^{\text{TEST}}_{\text{intvl-}\gamma}$, Theorem 4.2.8 and Theorem 4.3.5.

The following propositions will be useful in the technical development in this section:

Proposition 4.4.3 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. The *intervals* and *augment* functions on $\mathcal{D}^{\text{MAY-intvl}}_{Act}$ are continuous and the *intervals*, *augment*, *extend*, and *0-split* functions on $\mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}$ are continuous.

Proposition 4.4.4 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Let $\langle PT, Act\rangle \in \mathcal{D}^{\text{MAY}}_{Act}$ and let $\langle PF, PD, Act'\rangle \in \mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$. Then

- $augment(intervals(\langle PT, Act\rangle)) \in \mathcal{D}^{\text{MAY}}_{Act}$
- $intervals(augment(intervals(\langle PT, Act\rangle))) = intervals(\langle PT, Act\rangle)$
- $augment(extend_{Act'}(0\text{-}split(intervals(\langle PF, PD, Act'\rangle)))) \in \mathcal{D}^{\text{MUST-split-}\gamma}_{Act'}$
- $intervals(augment(extend_{Act'}(0\text{-}split(intervals(\langle PF, PD, Act'\rangle))))) =$ $intervals(\langle PF, PD, Act'\rangle)$
- $\langle PF, PD, Act'\rangle \sqsubseteq^{\text{MUST-split-}\gamma}_{Act'} augment(extend_{Act'}(0\text{-}split(intervals(\langle PF, PD, Act'\rangle))))$

The proof of Proposition 4.4.3 is routine. The first two items of Proposition 4.4.4 follows easily from Proposition 4.2.1 and Definition 4.2.5. The remaining items of Proposition 4.4.4 follows easily from Proposition 3.2.15, Proposition 3.3.9, Proposition 4.2.1, and Definition 4.3.3. The details are left to the reader.

Theorem 4.4.5 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then $\langle \mathcal{D}^{\text{MAY-intvl}}_{Act}, \sqsubseteq^{\text{MAY-intvl}}_{Act}\rangle$, $\langle \mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}, \sqsubseteq^{\text{MUST-intvl-}\gamma}_{Act'}\rangle$, and $\langle \mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}, \sqsubseteq^{\text{TEST-intvl-}\gamma}_{Act,Act'}\rangle$, are complete partial orders.

Proof. It is easy to see that $\langle \mathcal{D}^{\text{MAY-intvl}}_{Act}, \sqsubseteq^{\text{MAY-intvl}}_{Act}\rangle$, $\langle \mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}, \sqsubseteq^{\text{MUST-intvl-}\gamma}_{Act'}\rangle$, and $\langle \mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}, \sqsubseteq^{\text{TEST-intvl-}\gamma}_{Act,Act'}\rangle$ are partial orders.

We give the proof of completeness for $\mathcal{D}^{\text{MAY-intvl}}_{Act}$. It is easy to see that $\langle \emptyset, Act\rangle$ is the least element of $\mathcal{D}^{\text{MAY-intvl}}_{Act}$. Let A be a directed set in $\mathcal{D}^{\text{MAY-intvl}}_{Act}$; then by definition of $\mathcal{D}^{\text{MAY-intvl}}_{Act}$, $A = intervals(B)$ for some set $B \subseteq \mathcal{D}^{\text{MAY}}_{Act}$. Proposition 4.4.3 and Proposition 4.4.4 imply that $augment(intervals(B))$ is a directed set in $\mathcal{D}^{\text{MAY}}_{Act}$, and Theorem 4.2.9 implies that

$\bigcup augment(intervals(B)) \in \mathcal{D}_{Act}^{\text{MAY}}$. By definition, $intervals(\bigcup augment(intervals(B))) \in \mathcal{D}_{Act}^{\text{MAY-intvl}}$, and by Proposition 4.4.3 and Proposition 4.4.4,

$$intervals(\bigcup augment(intervals(B))) = \bigcup intervals(augment(intervals(B))) = \bigcup(intervals(B)),$$

proving this case.

The proofs for $\mathcal{D}_{Act'}^{\text{MUST-intvl-}\gamma}$ and $\mathcal{D}_{Act,Act'}^{\text{TEST-intvl-}\gamma}$ are completely analogous, except that they use Theorem 4.3.6. The details are simple and are left to the reader. ∎

We now give a finite characterization of the compact elements of these domains.

Definition 4.4.6 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. A pair $\langle PT, Act\rangle$ is a *finite candidate* of $\mathcal{D}_{Act}^{\text{MAY-intvl}}$ iff $\langle PT, Act\rangle = intervals(\langle PT', Act\rangle)$ for some finite candidate $\langle PT', Act\rangle$ of $\mathcal{D}_{Act}^{\text{MAY}}$. A triple $\langle PF, PD, Act'\rangle$ is a *finite candidate* of $\mathcal{D}_{Act'}^{\text{MUST-intvl-}\gamma}$ iff $\langle PF, PD, Act'\rangle = intervals(\langle PF', PD' Act'\rangle)$ for some finite candidate $\langle PF', PD', Act'\rangle$ of $\mathcal{D}_{Act'}^{\text{MUST-split-}\gamma}$. A tuple $\langle \alpha, \beta\rangle$ is a *finite candidate* of $\mathcal{D}_{Act,Act'}^{\text{TEST-intvl-}\gamma}$ iff $\langle \alpha, \beta\rangle = intervals(\langle \alpha', \beta'\rangle)$ for some finite candidate $\langle \alpha', \beta'\rangle$ of $\mathcal{D}_{Act,Act'}^{\text{TEST-split-}\gamma}$.

Lemma 4.4.7 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then all finite candidates of $\mathcal{D}_{Act}^{\text{MAY-intvl}}$, $\mathcal{D}_{Act'}^{\text{MUST-intvl-}\gamma}$, and $\mathcal{D}_{Act,Act'}^{\text{TEST-intvl-}\gamma}$ are compact elements of $\mathcal{D}_{Act}^{\text{MAY-intvl}}$, $\mathcal{D}_{Act'}^{\text{MUST-intvl-}\gamma}$, and $\mathcal{D}_{Act,Act'}^{\text{TEST-intvl-}\gamma}$, respectively.

Proof. The proof for $\mathcal{D}_{Act}^{\text{MAY-intvl}}$ is identical to that for $\mathcal{D}_{Act}^{\text{MAY}}$ in Theorem 4.2.11.

Let $intervals(\langle PF, PD, Act'\rangle)$ be a finite candidate of $\mathcal{D}_{Act'}^{\text{MUST-intvl-}\gamma}$ and let A be a subset of $\mathcal{D}_{Act'}^{\text{MUST-split-}\gamma}$ such that $intervals(A)$ is a directed set in $\mathcal{D}_{Act'}^{\text{MUST-intvl-}\gamma}$ and $intervals(\langle PF, PD, Act'\rangle) \sqsubseteq_{Act'}^{\text{MUST-intvl-}\gamma} \bigsqcup intervals(A)$. By Propositions 4.4.3 and 4.4.4,

$$\begin{array}{lll} \langle PF, PD, Act'\rangle & \sqsubseteq_{Act'}^{\text{MUST-split-}\gamma} & augment(extend_{Act'}(\textit{0-split}(intervals(\langle PF, PD, Act'\rangle)))) \\ & \sqsubseteq_{Act'}^{\text{MUST-split-}\gamma} & augment(extend_{Act'}(\textit{0-split}(\bigsqcup intervals(A)))) \\ & \sqsubseteq_{Act'}^{\text{MUST-split-}\gamma} & \bigsqcup augment(extend_{Act'}(\textit{0-split}(intervals(A)))) \end{array}$$

Since by Theorem 4.3.8, $\langle PF, PD, Act'\rangle$ is a compact element of $\mathcal{D}_{Act'}^{\text{MUST-split-}\gamma}$, there is some $\langle PF_n, PD_n, Act'\rangle \in A$ with

$$\langle PF, PD, Act'\rangle \sqsubseteq_{Act'}^{\text{MUST-split-}\gamma} augment(extend_{Act'}(\textit{0-split}(intervals(\langle PF_n, PD_n, Act'\rangle)))).$$

Then by Proposition 4.4.4,

$$\begin{array}{l} intervals(\langle PF, PD, Act'\rangle) \sqsubseteq_{Act'}^{\text{MUST-intvl-}\gamma} \\ \quad intervals(augment(extend_{Act'}(\textit{0-split}(intervals(\langle PF_n, PD_n, Act'\rangle))))) \end{array}$$

which by Proposition 4.4.4 is equal to $intervals(\langle PF_n, PD_n, Act'\rangle)$, proving this case.

The proof for $\mathcal{D}_{Act,Act'}^{\text{TEST-intvl-}\gamma}$ is a simple consequence of the previous two cases and is left to the reader. ∎

Lemma 4.4.8 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. For every $\langle PT, Act\rangle \in \mathcal{D}_{Act}^{\text{MAY-intvl}}$, there is a directed set $A_1 \subseteq \mathcal{D}_{Act}^{\text{MAY-intvl}}$

of finite candidates with $\bigsqcup A_1 = \langle PT, Act\rangle$. For every $\langle PT, Act'\rangle \in \mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}$, there is a directed set $A_2 \subseteq \mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}$ of finite candidates with $\bigsqcup A_2 = \langle PT, Act'\rangle$. For every $\langle\alpha,\beta\rangle \in \mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}$, there is a directed set $A_3 \subseteq \mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}$ of finite candidates with $\bigsqcup A_3 = \langle\alpha,\beta\rangle$.

Proof. We prove the case for $\mathcal{D}^{\text{MAY-intvl}}_{Act}$. By definition, $\langle PT, Act\rangle = \mathit{intervals}(\langle PT', Act\rangle)$ for some $\langle PT', Act\rangle \in \mathcal{D}^{\text{MAY}}_{Act}$. Lemma 4.2.12 gives a directed set B of finite candidates of $\mathcal{D}^{\text{MAY}}_{Act}$ whose least upper bound in $\mathcal{D}^{\text{MAY}}_{Act}$ is $\langle PT', Act\rangle$. Thus, $\langle PT, Act\rangle = \mathit{intervals}(\bigsqcup B)$. By Proposition 4.4.3 and Definition 4.4.6, $\mathit{intervals}(B)$ is a directed set of finite candidates of $\mathcal{D}^{\text{MAY-intvl}}_{Act}$ and

$$\langle PT, Act\rangle = \mathit{intervals}(\bigsqcup B) = \bigsqcup \mathit{intervals}(B).$$

The proofs of the other cases are analogous and are omitted. ∎

Theorem 4.4.9 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then $\mathcal{D}^{\text{MAY-intvl}}_{Act}$, $\mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}$, and $\mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}$ are algebraic cpo's.

The theorem is a simple consequence of the definition of compact elements and Lemma 4.4.7 and Lemma 4.4.8 (*cf.* [18]).

Theorem 4.4.10 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. For every compact element $\langle PT, Act\rangle \in \mathcal{D}^{\text{MAY-intvl}}_{Act}$, there is some WT Net $\langle N_1, Act\rangle$ with $[\![\langle N_1, Act\rangle]\!]^{\text{MAY}}_{\text{intvl}} = \langle PT, Act\rangle$. For every compact element $\langle PF, PD, Act'\rangle \in \mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}$, there is some WT Net $\langle N_2, Act\rangle$ with $[\![\langle N_2, Act\rangle]\!]^{\text{MUST}}_{\text{intvl-}\gamma} = \langle PF, PD, Act'\rangle$. For every compact element $\langle\alpha,\beta\rangle \in \mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}$, there is some WT Net $\langle N_3, Act\rangle$ with $[\![\langle N_3, Act\rangle]\!]^{\text{TEST}}_{\text{intvl-}\gamma} = \langle\alpha,\beta\rangle$.

The theorem is a simple consequence of the definition of $[\![\cdot]\!]^{\text{MAY}}_{\text{intvl}}$, $[\![\cdot]\!]^{\text{MUST}}_{\text{intvl-}\gamma}$, $[\![\cdot]\!]^{\text{TEST}}_{\text{intvl-}\gamma}$, Lemma 4.4.8, and Theorems 4.2.14 and 4.3.11.

The operations on $\mathcal{D}^{\text{MAY-intvl}}_{Act}$, $\mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}$, and $\mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}$ are given in the proof of Theorem 3.3.10. We do not restate the definitions here. We have that:

Theorem 4.4.11 Let Act be a finite alphabet containing $\surd$ and let $Act' = \{a_0, a_1, a_2 : a \in Act - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then $\mathcal{D}^{\text{MAY-intvl}}_{Act}$, $\mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}$, and $\mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}$ are closed under prefixing, restriction, renaming, hiding, sequencing, internal choice, CCS choice, non-communicating parallel composition, CSP-style parallel composition, CCS-style parallel composition, split refinements, and choice refinements. Furthermore, all of these operations are continuous functions on the respective domains.

Let Act_1 be a finite alphabet containing $\surd$ and let $Act_1{}' = \{a_0, a_1, a_2 : a \in Act_1 - \{\gamma, \surd\}\} \cup \{\surd, \gamma\}$. Then *grow* Act_1 and *shrink* Act_1 are continuous functions from $\langle\mathcal{D}^{\text{MAY-intvl}}_{Act}, \sqsubseteq^{\text{MAY-intvl}}_{Act}\rangle$, $\langle\mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act'}, \sqsubseteq^{\text{MUST-intvl-}\gamma}_{Act'}\rangle$, and $\langle\mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act,Act'}, \sqsubseteq^{\text{TEST-intvl-}\gamma}_{Act,Act'}\rangle$ to $\langle\mathcal{D}^{\text{MAY-intvl}}_{Act'}, \sqsubseteq^{\text{MAY-intvl}}_{Act'}\rangle$, $\langle\mathcal{D}^{\text{MUST-intvl-}\gamma}_{Act_1{}'}, \sqsubseteq^{\text{MUST-intvl-}\gamma}_{Act_1{}'}\rangle$, and $\langle\mathcal{D}^{\text{TEST-intvl-}\gamma}_{Act_1,Act_1{}'}, \sqsubseteq^{\text{TEST-intvl-}\gamma}_{Act_1,Act_1{}'}\rangle$, respectively.

The theorem follows easily from Theorems 4.2.17 and 4.3.12, and Proposition 4.4.3.

Chapter 5

Action Refinement

5.1 An Action Refinement Operator

This section presents our action refinement operator on a restricted class of WT Nets. In order to preserve the finite marking condition on Well-Terminating Nets, we will define our action refinement on a suitably restricted subclass of WT Nets.

Definition 5.1.1 The class of *Refinable Well-Terminating (RWT) Nets* consists of WT Nets $\langle N, Act\rangle$ in which every place has only a finite number of transitions with labels from $Act - \{\surd\}$ emanating from it, *i.e.*, for all $s \in S_N$, the set $\{t \in post_N(s) : l_N(t) \in Act - \{\surd\}\}$ is finite.

It is easy to show that:

Theorem 5.1.2 The class of RWT Nets is closed under prefixing ($\alpha.$), restriction ($\backslash a$), hiding ($-a$), renaming ($[f]$), CSP-style sequencing ($;$), non-communicating parallel composition ($||$), CCS-style parallel-composition-with-hiding ($|$), internal choice ($\oplus$), start-unwinding, CCS-style choice ($+_M$), *split*, and *choice*, but *not* under CSP-style parallel composition ($||_A$).

Proof. CSP-style parallel composition, $||_{\{a\}}$, applied to two RWT Nets that each contain an infinite number of a-labeled transitions will result in a net in which every a-labeled transition of one net is allowed to synchronize with every a-labeled transition of the other net. We recall that by definition, all transitions in WT Nets have non-empty presets. Thus, it is easy to see that some place in this resulting net must have an infinite number of a-labeled transitions emanating from it, violating the defining condition on RWT Nets.

The proof that RWT Nets are closed under all the other net operators follows easily from Theorem 2.2.16 and is omitted. ∎

Our action refinement operator $\langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]$ "replaces" each a-labeled transition in the *target net* $\langle N, Act\rangle$ by a separate but identical copy of the *refinement net* $\langle N_a, Act\rangle$; these copies are distinguished by "tagging" the names of the places and transitions of N_a with the name of the corresponding a-labeled transition. We want our action refinement operator to satisfy some intuitively simple distributivity properties, and so we need to be careful in how we hook up the copies of N_a to the places of N.

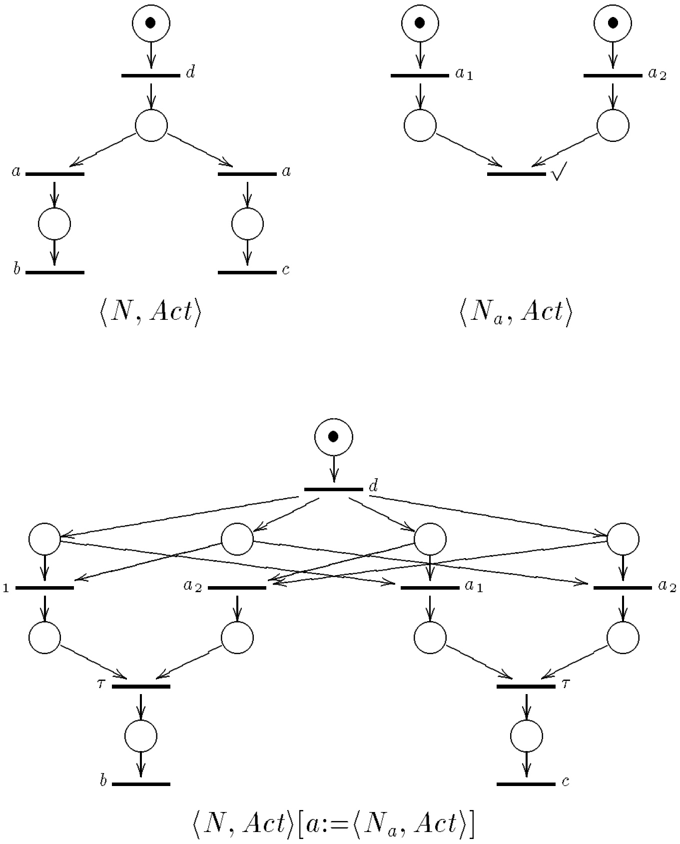


Figure 5-1: An Example of Action Refinement

In the same spirit as the definition of the $+_M$ operator, we take cross products of the start places of appropriate copies of N_a; in particular, for every place s in N, we take a cross product v of the start places of the copies of N_a corresponding to the a-labeled transitions emanating from s. Furthermore, in the same spirit as the definition of sequencing, we relabel with τ all of the $\surd$-labeled transitions of the copies of N_a and connect them all up to the post-set of the corresponding a-labeled transition. The other transitions of the copies of N_a and the non-a-labeled transitions of N are then hooked up to all of these places in the expected manner.

Not surprisingly, we encounter the same difficulties as the $+_M$ operator when our refinement nets have initially marked places that have incoming transitions, and we thus start-unwind the refinement net before performing our replacements.

The action refinement operator is illustrated in Figure 5-1.

We now define the action refinement operator. For simplicity we assume that the refinement net N_a is already start-unwound; otherwise, we first start-unwind N_a and then carry out this construction using the start-unwound version of N_a rather than N_a itself.

Definition 5.1.3 Let $\langle N, Act\rangle$ and $\langle N_a, Act\rangle$ be RWT Nets over a common alphabet Act, and let a be a label in $Act - \{\surd\}$. Then $\langle P, Act\rangle = \langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]$ is defined as:

$$\begin{aligned} S_P = \{(s,v) \mid s \in S_N \text{ and } v\colon T \to \mathit{Start}_{N_a},\ \text{where } T = \{t \in \mathit{post}_N(s) \mid l_N(t) = a\}\ \} \\ \uplus\ \ \{(t,s') \mid t \in T_N,\ l_N(t) = a \text{ and } s' \in S_{N_a} - \mathit{Start}_{N_a}\} \end{aligned}$$

$$T_P = \{(t,*) \mid t \in T_N \text{ and } l_N(t) \neq a\} \uplus \{(t,t') \mid t \in T_N,\ l_N(t) = a \text{ and } t' \in T_{N_a}\}$$

$$\mathit{pre}_P((t,*)) = \{(s,v) \in S_P \mid s \in \mathit{pre}_N(t)\}$$

$$\mathit{post}_P((t,*)) = \{(s,v) \in S_P \mid s \in \mathit{post}_N(t)\}$$

$$l_P((t,*)) = l_N(t)$$

$$\begin{aligned} \mathit{pre}_P((t,t')) = \{(s,v) \in S_P \mid s \in \mathit{pre}_N(t) \text{ and } v(t) \in \mathit{pre}_{N_a}(t')\} \\ \uplus\ \ \{(t,s') \in S_P \mid s' \in \mathit{pre}_{N_a}(t')\} \end{aligned}$$

$$\mathit{post}_P((t,t')) = \begin{cases} \{(t,s') \in S_P \mid s' \in \mathit{post}_{N_a}(t')\} & \text{if } l_{N_a}(t') \neq \surd \\ \{(s,v) \in S_P \mid s \in \mathit{post}_N(t)\} & \text{otherwise} \end{cases}$$

$$l_P((t,t')) = \begin{cases} l_{N_a}(t') & \text{if } l_{N_a}(t') \neq \surd \\ \tau & \text{otherwise} \end{cases}$$

$$\mathit{Start}_P = \{(s,v) \in S_P \mid s \in \mathit{Start}_N\}$$

We refer to the net $\langle N, Act\rangle$ as the *target* of action refinement, or the *target net*, and we refer to the net $\langle N_a, Act\rangle$ as the *operator* of action refinement, or the *refinement net*.

The following facts will be useful in proving that the class of RWT Nets is closed under action refinement.

Definition 5.1.4 Let $\langle N, Act\rangle$ and $\langle N', Act\rangle$ be RWT Nets over a common alphabet Act, let a be a label in $Act - \{\surd\}$, and let r be a run of N of length n. A *$\langle N, r, a, N'\rangle$-respecting substitution* is a function α from the set $\{i : 1 \leq i \leq n \text{ and } l_N(r[i]) = a\}$ to non-empty runs of N' such that for all $i \in \mathit{dom}(\alpha)$, if i is a non-maximal event in the pomset-run of r, then some $\surd$-labeled transition of N' occurs in $\alpha(i)$. Let $\mathit{dom}(\beta) = \mathit{dom}(\alpha)$, and let each $\beta(i) = (r[i], t_1) \ldots (r[i], t_k)$, where $\alpha(i) = t_1 \ldots t_k$. Then we define $r\alpha = r_1 \ldots r_n$, where each $r_i = \beta(i)$ if $i \in \mathit{dom}(\beta)$, and $r_i = (r[i], *)$ otherwise.

Lemma 5.1.5 Let $\langle N, Act\rangle$ and $\langle N_a, Act\rangle$ be RWT Nets over a common alphabet, Act, such that $\langle N_a, Act\rangle$ is start-unwound, and let a be a label in $Act - \{\surd\}$. Also, let $\langle P, Act\rangle = \langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]$, and let r' be a run of P. Then there is some run r of N and some

$\langle N, r, a, N_a\rangle$-respecting substitution α such that $r\alpha$ is a run of P whose pomset-run is isomorphic to that of r'.

Furthermore, let M' be the marking of P reached after firing r', let M be the marking of N reached after firing r, and for all $i \in dom(\alpha)$, let M_i be the marking of N_a reached after firing the run $\alpha(i)$. Then for all places of P of the form $(t, s') \in T_N \times S_{N_a}$,

$$M'(t,s') = \begin{cases} 0 & \text{if } t \text{ does not occur in } r \\ M_i(s') & \text{if } r[i] \text{ is the last occurrence of } t \text{ in } r \end{cases}$$

For all places of P of the form (s, v),

$$M'(s,v) = \begin{cases} 0 & \text{if for some } i \in dom(\alpha) \text{ with } r[i] \in pre_N(s), \\ & \quad \alpha(i) \text{ does not contain a } \surd\text{-labeled transition} \\ 1 & \text{if there is some } i \in dom(\alpha) \text{ with } r[i] \in post_N(s) \text{ such that } M_i(v(r[i])) = 1 \\ M(s) & \text{otherwise} \end{cases}$$

Proof. The proof is by induction on the length of r'. It is easy to see that lemma holds for the base case of $|r'| = 0$. For the other base case, suppose $|r'| = 1$. The proof is obvious if $r' = (t, *)$ for some transition t of N that is not a-labeled. Otherwise, $r' = (t, t')$ for some a-labeled transition t of N and some transition t' of N_a. Let $r = t$, let $dom(\alpha) = \{1\}$, and let $\alpha(1) = t'$. Then it is easy to see that $r\alpha = r'$ and that markings of places of the form $T_N \times S_{N_a}$ satisfy the above equation. The remaining property about markings is easily verified.

For one induction step, let $r' = r''.(t, *)$ for some non-empty run r'' and some transition t of N that is not a-labeled. By induction, there is some run r of N and some $\langle N, r, a, N_a\rangle$-respecting substitution α such that $r\alpha$ is a run of P whose pomset-run is isomorphic to that of r'' and the properties of the corresponding markings hold. By Proposition 3.2.3, the marking of P reached after r'' is identical to that reached after $r\alpha$. Thus, all places $(s, v) \in S_P$ with $s \in pre_N(t)$ must be marked in P. Suppose for the sake of contradiction that for some place $s \in pre_N(t)$ and every corresponding v, there is some $i \in dom(\alpha)$ with $s \in pre_N(r[i])$ such that $M_i(v(r[i])) = 1$. Since N is 1-safe, $\langle N_a, Act\rangle$ is start-unwound, and the firing of $\surd$-labeled transitions cleans out N and N_a, it would follow from Definition 5.1.3 that there is unique such i and that $\alpha(i)$ is empty, contradicting the definition of α. Thus, it follows from the inductive hypothesis about markings that all places $s \in pre_N(t)$ are marked in N. It is then easy to see from Definition 5.1.3 that $r.t$ is a run of N, α is a $\langle N, (r.t), a, N_a\rangle$-respecting substitution, and $(r.t)\alpha$ is a run of P whose pomset-run is isomorphic to that of r'. Furthermore, since N is 1-safe, it is easy to see that the desired property of the markings holds.

For the other induction step, let $r' = r''.(t, t')$ for some non-empty run r'', some a-labeled transition t of N, and some transition t' of N_a. By induction, there is some run r of N and some $\langle N, r, a, N_a\rangle$-respecting substitution α such that $r\alpha$ is a run of P whose pomset-run is isomorphic to that of r'' and the properties of the corresponding markings hold. For one case, suppose that for every occurrence $r[i]$ of t in r, $\alpha(i)$ contains a $\surd$-labeled transition. Since the firing of $\surd$-labeled transitions cleans out N_a, it is easy to see that all such markings M_i are empty. It then follows easily from the 1-safeness of N that all places $s \in pre_N(t)$ must be marked in N, and hence that $r.t$ is a run of N. Let α' be the extension of α with $\alpha(|r.t|) = t'$. It is easy to show that α' is a $\langle N, (r.t), a, N_a\rangle$-respecting substitution, and that $(r.t)\alpha'$ is a run of P whose pomset-run is isomorphic to that of r'. Furthermore, since N is 1-safe, it is easy to see that the desired property of the markings holds.

For the last case, suppose that for there is some occurrence $r[i]$ of t in r such that $\alpha(i)$ does not contain a $\surd$-labeled event. The 1-safeness of N and Definition 5.1.4 immediately imply that i is unique and is the last occurrence of t in r. Thus, it follows from the inductive hypothesis about markings that t' is enabled in N_a under marking M_i. Let $\alpha'(k) = \alpha(k).t'$, let $dom(\alpha') = dom(\alpha)$, and let α' agree with α on the rest of $dom(\alpha)$. It is easy to see from the above fact about M_i that α' is a $\langle N, r, a, N_a\rangle$-respecting substitution. Furthermore, it follows easily from the 1-safeness of N and the inductive hypothesis about markings that $r\alpha'$ is a run of P whose pomset-run is isomorphic to that of r'. Finally, since N is 1-safe, it is easy to see that the desired property of the markings holds, proving the lemma. ∎

The following related fact will be useful in proving properties about our semantics:

Lemma 5.1.6 Let $\langle N, Act\rangle$ and $\langle N_a, Act\rangle$ be RWT Nets over a common alphabet, Act, such that $\langle N_a, Act\rangle$ is start-unwound, and let a be a label in $Act - \{\surd\}$. Also, let r be a run of N and let α be a $\langle N, r, a, N_a\rangle$-respecting substitution. Then $r\alpha$ is a run of $\langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]$.

Proof. Using Lemma 5.1.5, a straightforward induction on the length of r gives the proof. The details are left to the reader. ∎

We now have:

Theorem 5.1.7 The class of RWT Nets is closed under action refinement.

Proof. Let $\langle N, Act\rangle, \langle N_a, Act\rangle$ be RWT Nets over a common alphabet, Act, let $a \in Act - \{\surd\}$, and let $\langle P, Act\rangle = \langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]$. For simplicity, we assume that $\langle N_a, Act\rangle$ is itself start-unwound; however, since by Proposition 5.1.2, *start-unwind*$(\langle N_a, Act\rangle)$ is a RWT Net, the proof is identical for the general case except that we use *start-unwind*$(\langle N_a, Act\rangle)$ instead of $\langle N_a, Act\rangle$. It is easy to see that the initial marking of P is non-empty and that all transitions in P have non-empty presets. Definition 2.1.2 and Definition 5.1.1 then imply that only a finite number of places of N_a are initially marked and that only a finite number of a-labeled transitions emanate from any given place in N. Thus, it is easy to see that for every place $s \in S_N$, there are only a finite number of functions v: $T \rightarrow Start_{N_a}$, where $T = \{t \in post_N(s) \mid l_N(t) = a\}$. Using the fact that $\langle N, Act\rangle$ and $\langle N_a, Act\rangle$ are RWT Nets, it is then easy to show that the initial marking of P is finite, the preset and post-set of every transition in P is finite, and that every place in P has only a finite number of transitions with labels in $Act - \{\surd\}$ emanating from it.

Lemma 5.1.5 together with Proposition 3.2.3 and the 1-safeness of N and N_a immediately implies that P is 1-safe. Similarly, Lemma 5.1.5 together with Proposition 3.2.3 and the fact that all places of N and N_a are unmarked immediately after the firing of any $\surd$-labeled transition of the respective net immediately implies that the same property about $\surd$-labeled transitions holds for P. Finally, the finite-enabling property for P follows easily from Lemma 5.1.5 together with the definition of pomset-runs, Proposition 3.2.3, the fact that $\langle N_a, Act\rangle$ is start-unwound, and the fact that only a finite number of transitions are enabled under any reachable marking of N or N_a. ∎

Our action refinement operator has a rich algebraic theory. For example, the following simple identities hold up to $[\![\cdot]\!]^{\mathrm{TEST}}_{\mathrm{split}\text{-}\gamma}$ equality. We write *Succ* to denote the WT net which must immediately successfully terminate, *i.e.*, exactly one transition is enabled under its initial marking and this transition is $\surd$-labeled. For notational convenience, we simply write a to refer to the net $a.\mathit{Succ}$ and write $a_+.a_-$ to refer to the net $a_+.a_-.\mathit{Succ}$. Furthermore, we write *Dead* to denote the deadlocked process consisting of a single initially marked place and no transitions, and $a.\mathit{Dead}$ to refer to the net that does an a and then deadlocks in the sense that no place is marked.

Our action refinement operator satisfies the following simple identities:

Proposition 5.1.8 Let $\langle N, Act\rangle, \langle N_1, Act\rangle, \langle N_2, Act\rangle$ be RWT Nets over a common alphabet Act, and let $a, a_+, a_-, a_L, a_R, b \in Act - \{\surd\}$. Then the following identities hold up to $[\![\cdot]\!]^{\mathrm{TEST}}_{\mathrm{split}\text{-}\gamma}$ equality:

$$\begin{array}{rcl}
a[a{:=}\langle N;\surd, Act\rangle] & = & \langle N;\surd, Act\rangle \\
\langle N, Act\rangle[a{:=}\langle a, Act\rangle] & = & \langle N, Act\rangle \\
\langle N, Act\rangle[a{:=}\langle \mathit{Dead}, Act\rangle] & = & \langle N, Act\rangle\backslash a \\
\langle N, Act\rangle[a{:=}\langle \tau, Act\rangle] & = & \langle N, Act\rangle[a{:=}\langle \mathit{Succ}, Act\rangle] = \langle N, Act\rangle - a \\
\mathit{split}_{(a,a_+,a_-)}(\langle N, Act\rangle) & = & \langle N, Act\rangle[a{:=}\langle a_+.a_-, Act\rangle] \\
\mathit{choice}_{(a,a_L,a_R)}(\langle N, Act\rangle) & = & \langle N, Act\rangle[a{:=}\langle a_L +_M a_R, Act\rangle]
\end{array}$$

Assuming that a and b are "fresh" labels, (*i.e.*, N_1 and N_2 contain no a-labeled or b-labeled transitions), we also have up to $[\![\cdot]\!]^{\mathrm{TEST}}_{\mathrm{split}\text{-}\gamma}$ equality:

$$\begin{array}{rcl}
((a +_M b)[a{:=}\langle N_1;\surd, Act\rangle])[b{:=}\langle N_2;\surd, Act\rangle] & = & \langle N_1;\surd, Act\rangle +_M \langle N_2;\surd, Act\rangle \\
((a.b)[a{:=}\langle N_1;\surd, Act\rangle])[b{:=}\langle N_2;\surd, Act\rangle] & = & \langle N_1;\surd, Act\rangle;\langle N_2;\surd, Act\rangle \\
((a \mid\mid b)[a{:=}\langle N_1;\surd, Act\rangle])[b{:=}\langle N_2;\surd, Act\rangle] & = & \langle N_1;\surd, Act\rangle \mid\mid \langle N_2;\surd, Act\rangle
\end{array}$$

For all refinements ρ, the following distributivity properties hold up to $[\![\cdot]\!]^{\mathrm{TEST}}_{\mathrm{split}\text{-}\gamma}$ equality:

$$\begin{array}{rcl}
(\langle N_1, Act\rangle +_M \langle N_2, Act\rangle)\rho & = & \langle N_1, Act\rangle\rho +_M \langle N_2, Act\rangle\rho \\
(\langle N_1, Act\rangle;\langle N_2, Act\rangle)\rho & = & \langle N_1, Act\rangle\rho;\langle N_2, Act\rangle\rho \\
(\langle N_1, Act\rangle \mid\mid \langle N_2, Act\rangle)\rho & = & \langle N_1, Act\rangle\rho \mid\mid \langle N_2, Act\rangle\rho
\end{array}$$

Proof. We give a sketch of the proofs of these identities. It is easy to see that that

$$\langle N, Act\rangle[a{:=}\langle \mathit{Dead}, Act\rangle] = \langle N, Act\rangle\backslash a$$

holds up to net isomorphism. For the remaining identities in the first set, we first note that all of the refinement nets satisfy the property that for every reachable marking under which a $\surd$-labeled transition is enabled, no non-$\surd$-labeled transition is enabled under that marking. It is straightforward to see that the τ-labeled transitions resulting from hiding these $\surd$-labeled transitions during action refinement thus do not create any extra failure sets. The identities then follow easily.

For the second set of identities, we note for all nets of the form $\langle N;\surd, Act\rangle$, the τ-labeled transitions resulting from hiding the $\surd$-labeled transitions during action refinement do not create any extra failure sets. The identity for $+_M$ then follows easily. It is straightforward to show that start-unwinding preserves $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$-meanings, from which the identity for sequencing follows easily. Finally, it is easy to see that the different cross-products of $\surd$-labeled transitions in the identity for parallel composition do not affect $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$ equality.

To prove the distributive properties, it is easy to see that the identities for sequencing and non-communicating parallel composition hold up to net isomorphism. The identity for $+_M$ follows easily from the definitions of start-unwinding and action refinement. ∎

Our definition of refinement generalizes the definitions of refinement given by Vogler [47] and van Glabbeek/Goltz [41] in the sense that our refined net is $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$-equivalent to their nets. In fact, there is an even tighter relationship between them, namely, these nets are equivalent up to a weaker form of history-preserving bisimulation [39] which treats τ-moves as hidden and respects concurrent divergences. We omit the definition here since it is not necessary in our development. Since Vogler and van Glabbeek/Goltz use a cross-product construction on the "accept" places of their refinement nets rather than using τ-moves to transfer control back to the target net, our refined net is not quite *strongly* history-preserving bisimilar to their nets. However, if we attach a single $\surd$-labeled transition to their set of accept places, we obtain nets which satisfy the property that for every reachable marking under which a $\surd$-labeled transition is enabled, no non-$\surd$-labeled transition is enabled under that marking. Thus, the τ-labeled transitions resulting from hiding these $\surd$-labeled transitions during action refinement do not create any extra failure sets. We note, however, that van Glabbeek/Goltz do not impose finiteness conditions on their nets since they do not have a hiding operation. Vogler imposes a more liberal finiteness condition than ours since his action refinement operator does not allow refinement nets to have "initial concurrency".

We note that our definition of action refinement preserves finiteness of nets, and thus, in the same spirit as our full class of RWT nets, we can allow arbitrary finite RWT nets to function as both target nets and refinement nets. The class of finite RWT nets is also closed under all of the net operations presented in Chapter 2, including CSP-style parallel composition.

5.2 Semantics for Action Refinement

Since RWT Nets by definition are a subclass of WT Nets, all of the net semantics developed in Chapter 3 are well-defined on RWT Nets. This section shows that all of these semantics are compositional for action refinement, *except* for $[\![\cdot]\!]^{\text{MUST}}$ and $[\![\cdot]\!]^{\text{TEST}}$.

Proposition 5.2.1 $[\![\cdot]\!]^{\text{MUST}}$ and $[\![\cdot]\!]^{\text{TEST}}$ are *not* compositional for RWT Nets as either targets or operators of action refinement.

By Theorem 3.2.47, $[\![\cdot]\!]^{\text{TEST}}_{\text{split-}\gamma}$-equality implies $[\![\cdot]\!]^{\text{TEST}}$-equality. The proposition is then a simple consequence of Proposition 3.2.33 and Proposition 5.1.8.

The following definitions will be useful in proving the compositionality of the other semantics.

Definition 5.2.2 Let p be a pomset over an alphabet Act, let $A \subseteq Act - \{\surd\}$, and let f map every event e in p whose label is in A to some (possibly empty) pomset p_e over Act. The pomset $q = p[A{:=}f]$ is defined as:

- $\text{Events}_q = \{(e,*) : e \in \text{Events}_p \text{ and } l_p(e) \notin A\}$
  $\cup \{(e,e') : e \in \text{Events}_p,\ l_p(e) \in A \text{ and } e' \in \text{Events}_{p_e}\}$.
- $l_q((e,*)) = l_p(e)$ and $l_q((e,e')) = l_{p_e}(e')$.
- For all $(e_1,\alpha_1),(e_2,\alpha_2) \in \text{Events}_q$, $(e_1,\alpha_1) <_q (e_2,\alpha_2)$ iff either $e_1 <_p e_2$ or ($e_1 =_p e_2$, $l_p(e_1) \in A$, and $\alpha_1 <_{p_{e_1}} \alpha_2$).

If A is a singleton set $\{a\}$, we write $p[a{:=}f]$ to denote $p[\{a\}{:=}f]$.

Since non-maximal events in a pomset-trace of a target net represent "fully fired" transitions, we must be careful to replace them only with "successfully terminated" pomset-traces of the refinement nets. The following definition reflects this fact:

Definition 5.2.3 Let PT and PT_a be sets of pomsets over a common alphabet, Act, and let $a \in Act - \{\surd\}$. Then:

$$\begin{aligned}\langle PT, Act\rangle[a{:=}\langle PT_a, Act\rangle] \stackrel{\text{def}}{=} augment(\{p[a{:=}f] : p \in PT \text{ and } & f \text{ maps every } a\text{-labeled event } e \text{ in } p\\ & \text{to some pomset } p_e \text{ in } PT_a \text{ such that}\\ & \text{if } e \notin max(p) \text{ then } p_e;\surd \in PT_a\})\end{aligned}$$

Definition 5.2.4 Let p be a pomset over an alphabet Act, let D_p be a (possibly empty) set of downward-closed subsets of Events_p, and let $A \subseteq Act$. Let g map every event e in p whose label is in A to some pair $\langle p_e, D_e\rangle$, where p_e is a (possibly empty) pomset over Act and D_e is a (possibly empty) set of downward-closed subsets of Events_{p_e}. Then $\langle q, D_q\rangle = \langle p, D_p\rangle[A{:=}g]$ is defined as:

- $q = p[A{:=}f]$, where $dom(f) = dom(g)$ and $f(e) = p_e (= fst(g(e)))$ for every event $e \in dom(g)$.
- $D_q = \{down_q(\{(e,\alpha) \in \text{Events}_q : e \in d\}) : d \in D_p\}$
  $\cup \{down_q(\{e\} \times d) : e \in \text{Events}_p,\ l_p(e) \in A,\ d \in D_e, \text{ and } d \neq \emptyset\}$
  $\cup \{down_q(\{(e',\alpha) \in \text{Events}_q : e' <_p e\}) : e \in \text{Events}_p,\ l_p(e) \in A, \text{ and } \emptyset \in D_e\}$

Since a_0-labeled events of *dupl-split* nets represent "fully fired" transitions, we must be careful to replace them only with "successfully terminated" pomset-traces of the refinement net. Similarly, since a_1-labeled events of *dupl-split* nets represent "half fired" transitions, we must be careful to replace them only with "non-terminated" pomset-traces of the refinement net. Furthermore, in order to be sure that the failure sets corresponding to these non-terminated pomset-traces remain valid after refinement, we require that these failure sets contain $\surd$; this ensures that new actions do not become ready by "looking through" the τ-transition corresponding to successful termination. As in the semantic definition of CSP-style parallel composition, we only refine *1-2-respect*ing pomsets to avoid confusion between "non-matching" a_1 and a_2-labeled actions.

As evidenced by Proposition 5.1.8, hiding is definable from action refinement. More generally, refining a-labeled transitions with any net that can successfully terminate after firing some finite sequence of τ-transitions will have the possible effect of hiding the a-labeled transitions, and hence may create additional divergences. In order to simplify our definition of action refinement on sets of pomset-failures and pomset-divergences, we first define a *replace* operator that ignores the effects of hiding on pomset-divergences (but does account for the independent effects on pomset-failures). Using this *replace* operator, we then define a semantic action refinement operator that properly accounts for all the effects of hiding.

Definition 5.2.5 Let Act be a finite alphabet containing the distinguished symbol $\surd$, let $Act' = \{a_i : a \in Act - \{\surd\} \text{ and } 0 \leq i \leq 2\} \cup \{\gamma, \surd\}$, and let $a \in Act - \{\surd\}$. Let PF, PF_a be sets of pomset-failures over Act', and let PD, PD_a be sets of pomset-divergences over Act'. Then:

$$\langle PF, PD, Act'\rangle[a \textit{ replace } \langle PF_a, PD_a, Act'\rangle] \stackrel{\text{def}}{=} \langle PF', PD', Act'\rangle, \text{ where}$$

$$\begin{array}{rcl} PF'' & = & \{\langle p[\{a_0,a_1\}{:=}f], F'\rangle : \langle p, F_p\rangle \in \textit{1-2-respect}(PF) \text{ for some } F_p, \\ & & \quad \text{and } f \text{ maps every event } e \text{ in } p \text{ with } l_p(e) \in \{a_0, a_1\} \\ & & \quad \text{to some pomset-failure } \langle p_e, F_e\rangle \text{ in } PF_a \text{ such that} \\ & & \quad \text{if } \langle \surd, \emptyset\rangle \in PF_a \text{ then } a_0 \in F_p, \\ & & \quad \text{if } l_p(e) = a_0 \text{ then } \langle p_e; \surd, \emptyset\rangle \in PF_a, \\ & & \quad \text{if } l_p(e) = a_1 \text{ then } \surd \in F_e \text{ and } p_e \text{ contains no } \surd\text{-labeled events}, \\ & & \quad \text{and } F' \subseteq (F_p \cup X) \cap \bigcap\{F_e : l_p(e) = a_1\}, \\ & & \quad \text{where } X = \{a_0, a_1, a_2\} - \textit{init}(PF_a)\} \end{array}$$

$$\begin{array}{rcl} PD'' & = & \{\langle p, D_p\rangle[\{a_0,a_1\}{:=}g] : \langle p, D_p\rangle \in \textit{1-2-respect}(PF) \cup \textit{1-2-respect}(PD), \\ & & \quad D_p \text{ is a (possibly empty) set of downward-closed subsets of } \text{Events}_p, \\ & & \quad g \text{ maps every event } e \text{ in } p \text{ with } l_p(e) \in \{a_0, a_1\} \\ & & \quad \text{to some } \langle p_e, D_e\rangle \text{ in } PF_a \cup PD_a \text{ such that} \\ & & \quad D_e \text{ is a (possibly empty) set of downward-closed subsets of } \text{Events}_{p_e}, \\ & & \quad D_p \cup \bigcup\{D_e : e \in \textit{dom}(g)\} \text{ is non-empty}, \\ & & \quad \text{and if } l_p(e) = a_0 \text{ then } \langle p_e; \surd, \emptyset\rangle \in PF_a\} \end{array}$$

$$\begin{array}{rcl} PF' & = & \textit{augment}(\textit{0-split}(PF'')) \cup \textit{implied-failures}_{Act'}(PD') \\ PD' & = & \textit{augment}(\textit{extend}_{Act'}(\textit{0-split}(PD''))) \end{array}$$

We now define the semantic action refinement to reflect the hiding behavior of action refinement:

Definition 5.2.6 Let Act be a finite alphabet containing the distinguished symbol $\surd$, let $Act' = \{a_i : a \in Act - \{\surd\} \text{ and } 0 \leq i \leq 2\} \cup \{\gamma, \surd\}$, and let $a \in Act - \{\surd\}$. Let PF, PF_a be sets of pomset-failures over Act', and let PD, PD_a be sets of pomset-divergences over Act'. Furthermore, let a' be an action not in $Act \cup Act'$. The following definitions use the operations presented in Definition 3.2.45 and Definition 5.2.5.

If $\langle \surd, \emptyset\rangle \notin PF_a$, then

$$\langle PF, PD, Act'\rangle[a{:=}\langle PF_a, PD_a, Act'\rangle] \stackrel{\text{def}}{=} \langle PF, PD, Act'\rangle[a \textit{ replace } \langle PF_a, PD_a, Act'\rangle]$$

Otherwise, if $\langle\surd,\emptyset\rangle \in PF_a$, then

$$\langle PF, PD, Act'\rangle[a{:=}\langle PF_a, PD_a, Act'\rangle]$$
$$\stackrel{\mathrm{def}}{=}$$
$$(((\textbf{choice}_{(a,a,a')}(\langle PF, PD, Act'\rangle \textbf{ grow } \{a'\})) \textbf{ hide } a') \textbf{ shrink } Act')[a \textit{ replace } \langle PF_a, PD_a, Act'\rangle]$$

We now show that our $[\![\cdot]\!]^{\mathrm{MAY}}$, $[\![\cdot]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$, and $[\![\cdot]\!]^{\mathrm{TEST}}_{\text{split-}\gamma}$ semantics are compositional for nets as *targets* and *operators* of action refinement. As discussed in the Introduction, this is in contrast to the semantics of [47], which is not compositional for nets as action refinement operators.

Theorem 5.2.7 $[\![\cdot]\!]^{\mathrm{MAY}}$, $[\![\cdot]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$, and $[\![\cdot]\!]^{\mathrm{TEST}}_{\text{split-}\gamma}$ are compositional for RWT Nets as targets and operators of action refinement.

Proof. Let Act be a finite alphabet containing $\surd$, let $a \in Act - \{\surd\}$, and let $\langle N, Act\rangle$, $\langle N_a, Act\rangle$ be RWT Nets.

To prove compositionality of the $[\![\cdot]\!]^{\mathrm{MAY}}$ semantics, we first observe that as a simple consequence of Lemma 5.1.5, Lemma 5.1.6, and the definition of pomset-traces and pomset-runs,

$$\begin{array}{lll} \multicolumn{3}{l}{\textit{pomset-traces}(\langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]) =} \\[1ex] \{p[a{:=}f] & : & p \in \textit{pomset-traces}(\langle N, Act\rangle) \text{ and } f \text{ maps every } a\text{-labeled event } e \text{ in } p \\ & & \text{to some pomset } p_e \text{ in } \textit{pomset-traces}(\langle N_a, Act\rangle) \text{ such that} \\ & & \text{if } e \notin \textit{max}(p) \text{ then } p_e;\surd \in \textit{pomset-traces}(\langle N_a, Act\rangle)\} \end{array}$$

The details are straightforward and are left to the reader.

It is now easy to see that

$$[\![\langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]]\!]^{\mathrm{MAY}} = [\![\langle N, Act\rangle]\!]^{\mathrm{MAY}}\ [a{:=}[\![\langle N_a, Act\rangle]\!]^{\mathrm{MAY}}],$$

where the action refinement operation on the right-hand side of the equation is that given in Definition 5.2.3.

We now prove compositionality of the $[\![\cdot]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$ semantics. For the first case, suppose that $\langle\surd,\emptyset\rangle \notin \textit{fst}([\![\langle N_a, Act\rangle]\!]^{\mathrm{MUST}}_{\text{split-}\gamma})$. As a consequence of Lemma 5.1.5, Lemma 5.1.6, and the definition of pomset-runs, pomset-traces, pomset-failures, and pomset-divergences, we have

$$\begin{array}{lll} \multicolumn{3}{l}{\textit{pomset-failures}(\langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]) =} \\[1ex] \textit{0-split}(\{\langle p[\{a_0,a_1\}{:=}f], F'\rangle & : & \langle p, F_p\rangle \in \textit{1-2-respect}(\textit{pomset-failures}(\langle N, Act\rangle)) \text{ for some } F_p, \\ & & \text{and } f \text{ maps every event } e \text{ in } p \text{ with } l_p(e) \in \{a_0, a_1\} \\ & & \text{to some pomset-failure } \langle p_e, F_e\rangle \text{ of } \langle N_a, Act\rangle \text{ such that} \\ & & \text{if } l_p(e) = a_0 \text{ then } \langle p_e;\surd,\emptyset\rangle \in \textit{pomset-failures}(\langle N_a, Act\rangle), \\ & & \text{if } l_p(e) = a_1 \text{ then } \surd \in F_e \text{ and } p_e \text{ contains no } \surd\text{-labeled events,} \\ & & \text{and } F' \subseteq (F_p \cup X) \cap \bigcap\{F_e : l_p(e) = a_1\}, \\ & & \text{where } X = \{a_0, a_1, a_2\} - \textit{init}(\textit{pomset-failures}(\langle N_a, Act\rangle))\}) \end{array}$$

pomset-divergences$(\langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]) =$

0-split$(\{\langle p, D_p\rangle[\{a_0, a_1\}{:=}g] : D_p$ is a (possibly empty) set of downward-closed subsets of Events_p,
$\langle p, D_p\rangle \in$ *1-2-respect*(*pomset-failures*$(\langle N, Act\rangle)$) $\cup$ *1-2-respect*(*pomset-divergences*$(\langle N, Act\rangle)$),
g maps every event e in p with $l_p(e) \in \{a_0, a_1\}$
to some $\langle p_e, D_e\rangle$ in *pomset-failures*$(\langle N_a, Act\rangle) \cup$ *pomset-divergences*$(\langle N_a, Act\rangle)$
such that D_e is a (possibly empty) set of downward-closed subsets of Events_{p_e},
$D_p \cup \bigcup\{D_e : e \in dom(g)\}$ is non-empty,
and if $l_p(e) = a_0$ then $\langle p_e; \surd, \emptyset\rangle \in$ *pomset-failures*$(\langle N_a, Act\rangle)\})$

The details are straightforward but tedious and are left to the reader.

We now show that for the case when $\langle \surd, \emptyset\rangle \notin fst([\![\langle N_a, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma})$,

$$[\![\langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]]\!]^{\text{MUST}}_{\text{split-}\gamma} = [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}\ [a{:=}[\![\langle N_a, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}],$$

where the action refinement operation on the right-hand side of the equation is that given in Definition 5.2.6.

One direction is a simple consequence of the definition of $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$, Definition 5.2.6, and the highlighted equality above for the pomset-failures and pomset-divergences of the refined net. For the other direction, let $\langle r, D_r\rangle \in snd([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}[a{:=}[\![\langle N_a, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma})$; then $\langle r, D_r\rangle \in augment(extend_{Act}(\langle q, D_q\rangle))$ for some pomset-divergence $\langle q, D_q\rangle$ such that $\langle q, D_q\rangle = \langle q_1, D_{q_1}\rangle[\{a_0, a_1\}{:=}g]$ for some $\langle q_1, D_{q_1}\rangle \in [\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}$ and some g mapping a_0-labeled and a_1-labeled events e of q_1 to $[\![\langle N_a, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}$. In turn, $\langle q_1, D_{q_1}\rangle \in augment(extend_{Act}(\langle p_1, D_{p_1}\rangle))$ for some $\langle p_1, D_{p_1}\rangle$ that is a pomset-divergence/pomset-failure of N, and each $g(e) \in augment(extend_{Act}(\langle p_e, D_{p_e}\rangle))$ for some $\langle p_e, D_{p_e}\rangle$ that is a pomset-divergence/pomset-failure of N_a. It is easy to show that $\langle q, D_q\rangle \in augment(extend_{Act}(\langle p_1, D_{p_1}\rangle[\{a_0, a_1\}{:=}g']))$, where g' is the restriction of g to a_0-labeled and a_1-labeled events e of p_1. Hence, the highlighted fact above together with the definition of $[\![\cdot]\!]^{\text{MUST}}_{\text{split-}\gamma}$ implies that $\langle q, D_q\rangle \in snd([\![\langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]]\!]^{\text{MUST}}_{\text{split-}\gamma})$. It now follows from Proposition 3.2.18 that $\langle r, D_r\rangle \in snd([\![\langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]]\!]^{\text{MUST}}_{\text{split-}\gamma})$. The proof for pomset-failures in $\langle r, D_r\rangle \in fst([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}[a{:=}[\![\langle N_a, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}])$ is similar and is left to the reader.

The other case, when $\langle \surd, \emptyset\rangle \in fst([\![\langle N_a, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma})$, then follows from the above proof, Definition 5.2.6, Theorem 3.2.46, and the following easily proved fact: if $\langle \surd, \emptyset\rangle \in fst([\![\langle N_a, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma})$, then

$$[\![\langle N, Act\rangle[a{:=}\langle N_a, Act\rangle]]\!]^{\text{MUST}}_{\text{split-}\gamma}$$
$$=$$
$$(((\mathbf{choice}_{(a,a,a')}([\![\langle N, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}\ \mathbf{grow}\ \{a'\}))\ \mathbf{hide}\ a')\ \mathbf{shrink}\ Act')[a\ \textit{replace}\ [\![\langle N_a, Act\rangle]\!]^{\text{MUST}}_{\text{split-}\gamma}]$$

The details are left to the reader. ∎

In order to prove compositionality of the corresponding interval semantics, we will need the following facts about refining interval pomsets and interval pomset-divergences:

Proposition 5.2.8 Let q be an interval pomset such that $q \in \mathit{augment}(p[a{:=}f])$ for some pomset p and function f mapping a-labeled events in p to pomsets. Then $q \in \mathit{augment}(p'[a{:=}f'])$ for some interval pomset $p' \succeq p$ and some function f' mapping a-labeled events in p' to interval pomsets such that $f'(e) \succeq f(e)$ for all $e \in \mathit{dom}(f)$.

Proposition 5.2.9 Let $\langle q, D_q\rangle$ be an interval pomset-divergence such that $\langle q, D_q\rangle \in \mathit{augment}(\langle p, D_p\rangle[A{:=}g])$ for some pomset-divergence $\langle p, D_p\rangle$ and function g mapping events in p with labels in A to pomset-divergences. Then $\langle q, D_q\rangle \in \mathit{augment}(\langle p', D_{p'}\rangle[A{:=}g'])$ for some interval pomset-divergence $\langle p', D_{p'}\rangle \succeq \langle p, D_p\rangle$ and some function g' mapping events in p' with labels in A to interval pomset-divergences such that $g'(e) \succeq g(e)$ for all $e \in \mathit{dom}(g)$.

Using Lemmas 3.3.7 and 3.3.8 to account for the possible hiding effect of action refinement, the proofs of the propositions are straightforward and left to the reader.

We now have:

Theorem 5.2.10 The $[\![\cdot]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}$, $[\![\cdot]\!]^{\mathrm{MUST}}_{\mathrm{intvl}\text{-}\gamma}$, and $[\![\cdot]\!]^{\mathrm{TEST}}_{\mathrm{intvl}\text{-}\gamma}$ are compositional for RWT Nets as targets and operators of action refinement.

Proof. Let Act be a finite alphabet containing $\surd$, let $a \in \mathit{Act} - \{\surd\}$, and let $\langle N, \mathit{Act}\rangle$, $\langle N_a, \mathit{Act}\rangle$ be RWT Nets.

We show that the following identities hold, where operations on the right-hand side of the equations are those given in Definition 5.2.3 and Definition 5.2.6.

$$[\![\langle N, \mathit{Act}\rangle[a{:=}\langle N_a, \mathit{Act}\rangle]]\!]^{\mathrm{MAY}}_{\mathrm{intvl}} = \mathit{intervals}([\![\langle N, \mathit{Act}\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}\ [a{:=}[\![\langle N_a, \mathit{Act}\rangle]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}])$$

$$[\![\langle N, \mathit{Act}\rangle[a{:=}\langle N_a, \mathit{Act}\rangle]]\!]^{\mathrm{MUST}}_{\mathrm{intvl}\text{-}\gamma} = \mathit{intervals}([\![\langle N, \mathit{Act}\rangle]\!]^{\mathrm{MUST}}_{\mathrm{intvl}\text{-}\gamma}\ [a{:=}[\![\langle N_a, \mathit{Act}\rangle]\!]^{\mathrm{MUST}}_{\mathrm{intvl}\text{-}\gamma}])$$

The identity for $[\![\cdot]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}$ is a simple consequence of the augmentation-closure of the $[\![\cdot]\!]^{\mathrm{MAY}}$ semantics, Theorem 5.2.7, and Proposition 5.2.8.

It is easy to see that one direction of the equation for $[\![\cdot]\!]^{\mathrm{MUST}}_{\mathrm{intvl}\text{-}\gamma}$ follows easily from Theorem 5.2.7 and the monotonicity of the action refinement operation. For the other direction, let $\langle r, D_r\rangle \in \mathit{snd}([\![\langle N, \mathit{Act}\rangle[a{:=}\langle N_a, \mathit{Act}\rangle]]\!]^{\mathrm{MUST}}_{\mathrm{intvl}\text{-}\gamma})$; then $\langle r, D_r\rangle \in \mathit{intervals}(\mathit{augment}(\mathit{extend}_{\mathit{Act}}(\langle p, D_p\rangle)))$ for some pomset-divergence $\langle p, D_p\rangle$ of $\langle N, \mathit{Act}\rangle[a{:=}\langle N_a, \mathit{Act}\rangle]$. By Lemma 3.2.15 and Lemma 3.3.9, there is some *interval* pomset-divergence $\langle q, \{d'\}\rangle$ such that $\langle r, D_r\rangle \in \mathit{augment}(\mathit{extend}_{\mathit{Act}}(\langle q, \{d'\}\rangle))$, q is an augmentation of a prefix of p and $d' \supseteq d$ for some $d \in D_p$. By Proposition 3.2.14,

$$\langle q, \{d'\}\rangle \in \mathit{extend}_{\mathit{Act}}(\mathit{augment}(\mathit{pomset\text{-}divergences}(\langle N, \mathit{Act}\rangle[a{:=}\langle N_a, \mathit{Act}\rangle]))).$$

It then follows easily from the highlighted fact in the proof of Theorem 5.2.7 that $\langle q, \{d'\}\rangle \in \mathit{extend}_{\mathit{Act}}(\mathit{augment}(\mathit{0\text{-}split}(\langle p', D_{p'}\rangle)))$ for some $\langle p', D_{p'}\rangle \in \langle p_1, D_1\rangle[\{a_0, a_1\}{:=}g]$, where $\langle p_1, D_1\rangle$ is an appropriate pomset-divergence or pomset-failure and g is a suitable function.

It follows from Lemma 3.3.9 and Proposition 5.2.9 that there are some *interval* pomset-divergences $\langle q', D_{q'}\rangle \succeq \langle p', D_{p'}\rangle$, $\langle q_1, D_{q_1}\rangle \succeq \langle p_1, D_1\rangle$ and $g'(e) \succeq g(e)$ for all $e \in dom(g)$, such that $\langle q, \{d'\}\rangle \in extend_{Act}(augment(\mathit{0\text{-}split}(\langle q', D_{q'}\rangle)))$ and $\langle q', D_{q'}\rangle \in augment(\langle q_1, D_{q_1}\rangle[\{a_0, a_1\}{:=}g'])$. From the definition of *0-split* and *augment* and Lemma 3.2.16, it is easy to see that

$$\langle q, \{d'\}\rangle \in augment(extend_{Act}(\mathit{0\text{-}split}(\langle q_1, D_{q_1}\rangle[\{a_0, a_1\}{:=}g']))).$$

The desired equality then follows easily. The proof for pomset-failures is similar, except that it uses Proposition 5.2.8 as well. ∎

We then have:

Theorem 5.2.11 The $[\![\cdot]\!]^{\mathrm{MAY}}_{\mathrm{intvl}}$, $[\![\cdot]\!]^{\mathrm{MUST}}_{\mathrm{intvl}\text{-}\gamma}$, and $[\![\cdot]\!]^{\mathrm{TEST}}_{\mathrm{intvl}\text{-}\gamma}$ semantics are respectively fully abstract for MAY-equivalence, MUST-equivalence, and Testing-equivalence with respect to alphabet expansion and action refinement.

Proof. It is easy to see from the definitions of the $[\![\cdot]\!]^{\mathrm{TEST}}_{\mathrm{split}\text{-}\gamma}$ and $[\![\cdot]\!]^{\mathrm{TEST}}_{\mathrm{intvl}\text{-}\gamma}$ that $[\![\cdot]\!]^{\mathrm{TEST}}_{\mathrm{split}\text{-}\gamma}$-equality implies $[\![\cdot]\!]^{\mathrm{TEST}}_{\mathrm{intvl}\text{-}\gamma}$-equality. Thus, Proposition 5.1.8 shows that split refinements, choice refinements, and CCS choice can be defined from action refinement up to $[\![\cdot]\!]^{\mathrm{TEST}}_{\mathrm{intvl}\text{-}\gamma}$-equality. The theorem is then a simple consequence of Theorem 3.3.11 and Theorem 5.2.10. ∎

5.3 The Semantic Domains Revisited

All of the semantic domains, *except* for $\mathcal{D}^{\mathrm{MUST}}_{Act}$ and $\mathcal{D}^{\mathrm{TEST}}_{Act}$, developed in Chapter 4 are closed under the appropriate action refinement operators given Definition 5.2.3 and Definition 5.2.6. Furthermore, these action refinements operators are continuous functions on the corresponding domains.

Theorem 5.3.1 The $\mathcal{D}^{\mathrm{MAY}}_{Act}$, $\mathcal{D}^{\mathrm{MUST\text{-}split\text{-}}\gamma}_{Act'}$, $\mathcal{D}^{\mathrm{TEST\text{-}split\text{-}}\gamma}_{Act,Act'}$, $\mathcal{D}^{\mathrm{MAY\text{-}intvl}}_{Act}$, $\mathcal{D}^{\mathrm{MUST\text{-}intvl\text{-}}\gamma}_{Act'}$, and $\mathcal{D}^{\mathrm{TEST\text{-}intvl\text{-}}\gamma}_{Act,Act'}$ domains are closed under action refinement. Furthermore, action refinement is a continuous function on all of these domains.

Proof. The proof that the domains are closed is straightforward but tedious; the details are left to the reader.

The continuity of action refinement on $\mathcal{D}^{\mathrm{MAY}}_{Act}$ and and $\mathcal{D}^{\mathrm{MAY\text{-}intvl}}_{Act}$ is completely routine to verify, as is the continuity for the other domains when $\langle \surd, \emptyset\rangle$ is not in the failure set of the refinement operator. The general case is then a simple consequence of Definition 5.2.6 and the continuity of alphabet expansion and shrinking, choice refinements, and hiding, which were proved in Chapter 4. ∎

Chapter 6

Deciding True Concurrency Equivalences

6.1 Introduction

The computational complexity of the equivalence problem for nondeterministic finite-state automata under a variety of standard process semantics has been tightly characterized. In particular, trace equivalence and failure equivalence [8] are PSPACE-complete [26], while bisimulation [30] is PTIME-complete [4, 26]. It has been shown recently that these equivalence problems are exponentially harder for automata presented as finite "Mazurkiewicz nets" of synchronized state-machines [35]: namely, trace equivalence and failure equivalence of these nets are EXPSPACE-complete [29, 34] and bisimulation of these nets is DEXPTIME-complete [36].

The known results for "true" concurrency equivalences are much more limited. Vogler [46, 48] has shown the decidability of history-preserving bisimulation [5, 35, 39, 50, 46] and maximality-preserving bisimulation [13, 50] for finite 1-safe Petri nets; however, their complexity remained open. Decidability of such a basic true concurrency property as *pomset-trace* equivalence [39] appears not to have been known. (An ordinary trace is a linear sequence of visible actions; pomset-traces generalize these to multi-sets of actions partially ordered to reflect causality and concurrency.)

In contrast to trace equivalence, the decidability of pomset-trace equivalence for finite nets does not obviously reduce to equivalence of finite automata. The difficulty is that if a run of a net has a pomset-trace isomorphic to the pomset-trace of a run of another net, then whether a transition firable after one run yields the "same" pomset extension as a transition firable after the other run depends *a priori* on the entire pomset trace, which may be unboundedly large. Hence instead of searching for a suitable equivalence relation on the finite set of net markings, one has to consider equivalence relations on a potentially infinite set of pomset traces and final markings.

A similar difficulty appears in deciding whether finite nets are history-preserving bisimilar, which Vogler [46, 48] overcomes by maintaining, instead of an entire pomset history, a partial order on the fixed set of places of the nets that reflects "most-recent" firings. We use a similar partial order, but instead of places, we find it technically smoother to keep track of the partial ordering between the most-recent firings of transitions. This idea leads to a decision procedure

Class	*Equivalence*	*Complexity*
Traces	Traces	EXPSPACE-complete
	Step-traces [25, 37, 39]	
	ST-traces [39, 42]	
	Interval-pomset-traces [23, 47]	
	Pomset-traces [23, 39, 47]	
Failures/ divergences	Failures [9]	EXPSPACE-complete
	Step-failures [25, 37, 39]	
	ST-failures [39, 42]	
	Interval-pomset-failures [23, 47]	
Bisimulation	Bisimulation [30]	DEXPTIME-complete
	Delay bisimulation [43]	
	Branching bisimulation [43]	
	Step-bisimulation [25, 39]	
	ST-bisimulation [39, 42]	
	History-preserving Bisimulation [5, 35, 39, 50, 46]	
	Maximality-preserving-bisimulation [13]	
	Pomset-bisimulation [6]	DEXPTIME-hard and in EXPSPACE
	Pomset-ST-bisimulation [50]	DEXPTIME-hard and in EXPSPACE

Table 6.1: Complexity results for finite 1-safe Petri Nets

for pomset-trace equivalence, and a simple analysis of this procedure yields an EXPSPACE upper bound.[1] The same approach also gives a DEXPTIME decision procedure for history-preserving bisimulation.

Our lower bounds for these true concurrency equivalences follow easily from reductions from the corresponding interleaving equivalences, whose lower bounds in turn essentially follow from the results of [29, 34, 36]. We thus obtain a tight bound of EXPSPACE-completeness for pomset-trace equivalence. Likewise, we obtain DEXPTIME-completeness for history-preserving bisimulation and maximality-preserving bisimulation, settling questions left open by Vogler [46, 48].

Our methods also yield tight complexity bounds for several other true concurrency equivalences, summarized in Table 6.1. In particular, our EXPSPACE-completeness results for ST-traces and ST-failures solve problems left open by Vogler [49], who had earlier proved the decidability of these equivalences. Furthermore, our decidability results for pomset-bisimulation and pomset-ST-bisimulation settle questions alluded to by Vogler [45].

This chapter is organized as follows. Section 6.2 describes our alternate characterization of pomset-trace equivalence, together with an EXPSPACE decision procedure. Similar analyses

[1] For expository purposes, we refer to bounds of the form $2^{O(n^k)}$ for fixed k as *exponential* in n. In the results presented here, k is at most 4.

of history-preserving bisimulation and pomset bisimulation are given in Section 6.3, while Section 6.4 describes decision procedures for the other equivalences. Section 6.5 gives lower bounds for all these equivalences. A discussion of some open problems appears in Section 6.6.

6.2 Deciding Pomset-Trace Equivalence

Throughout this chapter, we use the term *nets* to refer to marked, *1-safe* Petri Nets [46] whose transitions have labels from a fixed set $Act \cup \{\tau\}$, where Act is a set of "visible actions" and $\tau \notin Act$ is the "hidden action." A transition is *visible* (hidden) iff its label is visible (hidden). The *runs* of a net are its finite firing sequences [46]. A net is *finite* iff it has a finite number of places and transitions; the *size* of a net is the total number of its places and transitions.

Definition 6.2.1 A *pomset* is a labeled partial order. Formally, a pomset, p, consists of a set Events_p whose elements are called *events*, a set Labels_p whose elements are called *labels*, a function $\mathrm{label}_p\colon \mathrm{Events}_p \to \mathrm{Labels}_p$, and a partial order relation $<_p$ on Events_p. A function f is an *isomorphism* between pomset p and pomset q iff it is a label-preserving order-isomorphism, namely,

- $f\colon \mathrm{Events}_p \to \mathrm{Events}_q$ is a bijection,
- $\mathrm{label}_p = \mathrm{label}_q \circ f$,
- $e <_p e'$ iff $f(e) <_q f(e')$ for all $e, e' \in \mathrm{Events}_p$.

The *places* of a transition t of a net N are the places directly connected to it, *i.e.*, the union of the preset and postset of t. Let t_1, t_2 be transitions of a net N. We say that t_1 and t_2 are *statically concurrent* in N iff the places of t_1 are disjoint from the places of t_2.

A *transition-sequence*, r, is a sequence of transitions of a net N. The *transition-pomset* of r has as events the integers from 1 to n, where the label of event i is t_i and the partial ordering is the transitive closure of the following "proximate cause" relation: event i *proximately causes* event j iff $i < j$ and t_i and t_j are *not* statically concurrent in N, *cf.* Figure 6-1.

The *visible-pomset* of r is the transition-pomset of r, restricted to events labeled with visible transitions; moreover, in the visible-pomset, the label of event i is the *label* of t_i (rather than t_i itself), *cf.* Figure 6-1. The *pomset-traces* of N are the visible-pomsets of *runs* of N.

For transition-pomsets and visible-pomsets, it is traditional to say that event e *causes* event e' iff $e < e'$ in the partial order.

Definition 6.2.2 Let N and N' be nets. Then N *pomset-trace approximates* N', written $N \sqsubseteq_{\mathrm{pt}} N'$, iff every pomset-trace of N is isomorphic to some pomset-trace of N'. N and N' are *pomset-trace equivalent* iff each is $\sqsubseteq_{\mathrm{pt}}$ the other.

The runs of a finite net are clearly recognizable by a finite state automaton, namely, the "global state" automaton of the net itself. We represent an ordered pair $r = t_1 \ldots t_n$, $r'' = t''_1 \ldots t''_n$, of transition-sequences of the same length as an input string $(t_1, t''_1) \ldots (t_n, t''_n)$ for an automaton whose alphabet is ordered pairs of transitions. So an "obvious" solution to the pomset-trace equivalence problem would be to define an effective procedure that, given any two finite nets as input, computes a finite-state automaton whose language consists of all the

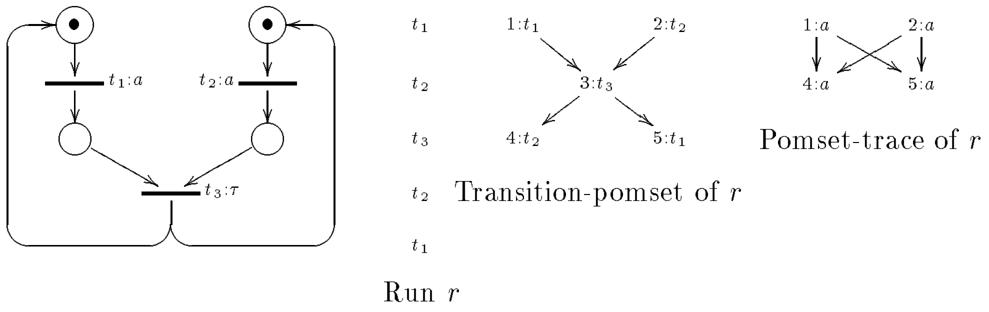


Figure 6-1: An Example of a Transition-pomset and Pomset-trace

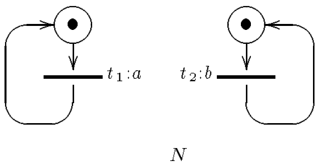


Figure 6-2: An Example

pairs of runs of the respective nets that have isomorphic pomset-traces. Such an automaton would easily yield a decision procedure for pomset-trace equivalence, since we could project the language it accepts onto the components of the pairs and check that the resulting languages include the set of runs of the respective nets.

However, such a finite-state automaton does not exist; the difficulty is that pairs of runs with isomorphic pomset-traces may generate the pomset-traces in different order, one getting unboundedly behind the other before catching up at the end. For example, let N be the net pictured in Figure 6-2. Then two runs of N have the same pomset-trace iff they have the same number of occurrences of a- and b-labeled transitions, and the set of such pairs of runs is obviously not finite-state recognizable.

We will show in this section that it suffices to consider pairs of runs that are "synchronous" in the sense that their behavior corresponds at each pair of transitions. We say that two runs r' and r'' are *equivalent up to concurrency* iff they have isomorphic transition-pomsets. We will show that:

- For all pairs of runs r and r' with isomorphic pomset-traces, there is a run r'' that is equivalent to r' up to concurrency, and r and r'' are "synchronous."

- The set of pairs of synchronous runs is recognizable by a finite automaton with size bounded by an exponential in the sizes of the nets.

Our decision procedure for pomset-trace equivalence is based on constructing such a finite-state automaton. To simplify the exposition, we consider first the case without hidden transitions. Our proofs will use the following fact about transition-pomsets:

Definition 6.2.3 A pomset p' is a *linearization* of a pomset p iff it has the same events and labels as p and $<_{p'}$ is a total ordering that contains $<_p$. Let q be a pomset such that $<_q$ is a total ordering. Then for any $1 \leq i \leq |\text{Events}_q|$, the i^{th} *largest event* of q is the (necessarily unique) event $e \in \text{Events}_q$ such that the longest chain $e_1 <_q \ldots <_q e_k <_q e$ in q is of length i.

Let $r = t_1 t_2 \ldots$ be a transition-sequence of a net N; we write $|r|$ for the length of r, and for any $1 \leq i \leq |r|$, we write $r[i]$ to denote the i^{th} element, t_i, of r.

Proposition 6.2.4 Let r be a run of a net N, let p' be a linearization of the transition-pomset of r, and let r' be the transition-sequence corresponding to p', *i.e.*, $r' = t_1 \ldots t_{|r|}$, where each t_i is the label of the i^{th} largest event of p'. Then r' is a run of N reaching the same final marking as r.

The proposition is easily proved by induction on the number of pairs (i, j) such that $i < j$ but the i^{th} event of p' is larger (in the standard integer ordering) than the j^{th} event of p'. The details are left to the reader.

6.2.1 Nets without Hidden Transitions

In this section, we assume that nets *do not contain hidden transitions.*

Definition 6.2.5 Let r and r' be transition-sequences of nets N and N', respectively. We say that r and r' are *synchronous* iff the identity function on $\{1, 2, \ldots, |r|\}$ is an isomorphism between the visible-pomset of r and the visible-pomset of r'.

In particular, if r and r' are synchronous, then they are of the same length.

We then have:

Lemma 6.2.6 Let r and r' be runs of nets N and N', respectively. If the pomset-traces of r and r' are isomorphic, then there is some run r'' of N' such that

- r' and r'' are equivalent up to concurrency, and
- r and r'' are synchronous.

Proof. Let I be the isomorphism between the pomset-trace of r and the pomset-trace of r'. Since in this section we assume that nets do not contain hidden transitions, clearly r and r' are of the same length. Let r'' be the transition-sequence obtained from r' by applying I element-wise to r; that is, $r''[i] = r'[I(i)]$ for all $1 \leq i \leq |r'|$.

It follows easily from the definition of r'' that I is a label-preserving bijection between the transition-pomsets of r'' and r'. To show that I is an order-isomorphism, it clearly suffices to show that I and I^{-1} preserve proximate causes. Let event i be a proximate cause of event j in the transition-pomset of r''. Then $i < j$, and transition $r''[i]$ and transition $r''[j]$ are not statically concurrent in N'; hence transition $r'[I(i)]$ and transition $r'[I(j)]$ are not statically concurrent in N'. $I(j) < I(i)$ would imply that event $I(j)$ is a proximate cause of event $I(i)$ in the pomset-trace of r'; since I is an isomorphism between the pomset-trace of r and the pomset-trace of r', it would follow that event j causes event i in the pomset-trace of r, and therefore that $j < i$, a contradiction. Hence $I(i) < I(j)$, and so event $I(i)$ is a proximate cause of $I(j)$ in the transition-pomset of r', proving this direction. The proof of the other direction is similar and omitted. This completes the proof that r' and r'' are equivalent up to concurrency; that is, they have isomorphic transition-pomsets.

Every transition-sequence corresponds to a linearization of its transition-pomset, by definition. Since r' is a run, and r' and r'' have isomorphic transition-pomsets, Proposition 6.2.4 immediately implies that r'' is a run of N'.

Clearly, I^{-1} is an isomorphism between the pomset-trace of r' and the pomset-trace of r''. Pomset isomorphisms are closed under function composition; thus $I^{-1} \circ I$, *i.e.*, the identity function on $\{1, \ldots, |r|\}$, is an isomorphism between the pomset-trace of r and the pomset-trace of r''. This implies that r and r'' are synchronous, completing the proof of the lemma. ∎

An important property of synchronous transition-sequences is that their equal-length prefixes are also synchronous.

Definition 6.2.7 Let p be a pomset and $e, e' \in \text{Events}_p$. Event e' is a *maximal cause* of event e in p providing $e' <_p e$ and there is no event $e'' \in \text{Events}_p$ such that $e' <_p e'' <_p e$.

Proposition 6.2.8 Let r and r' be transition-sequences of length $n \geq 0$ and let t and t' be transitions of nets N and N', respectively. Then $r.t$ and $r'.t'$ are synchronous iff

- r and r' are synchronous,
- t and t' have the same label, and
- the maximal causes of event $n + 1$ are the same in the transition-pomsets of $r.t$ and $r'.t'$.

The proof is completely straightforward and is left to the reader.

Thus, in determining whether two pomset-traces "grow" synchronously, it suffices to keep track of the correspondence between maximal causes. We now observe that all maximal causes will necessarily be the most-recent firings of the corresponding transitions.

Definition 6.2.9 Let $r = t_1 \ldots t_n$ be a transition-sequence of a net N. Event i is a *most recent firing of transition* t in r iff $t_i = t$ and $t_j \neq t$ for $i < j \leq n$. Let *growth-sites*(r) be the transition-pomset of r, restricted to the most-recent firings of the transitions in r, *cf.* Figure 6-3.

Proposition 6.2.10 Let $r = t_1 \ldots t_n$ be a transition-sequence and t be a transition of a net N. Then the maximal causes of event $n + 1$ in the visible-pomset of $r.t$ are a subset of the events of *growth-sites*(r).

Proof. Suppose event i of the visible-pomset of $r.t$ is a maximal cause of event $n + 1$. Then by the definition of the causal partial ordering, event i must be a proximate cause of event $n + 1$, and hence transition t_i must not be statically concurrent with t. Therefore any later firing of t_i, that is, any event j with $i < j \leq n$ and $t_j = t_i$, would also be a proximate cause of t. But since event i proximately causes any such event j, this would contradict event i being a maximal cause of event $n + 1$. ∎

We also make the simple observation that the growth-sites of transition-sequence $r.t$ are fully determined by t and the growth-sites of r:

Proposition 6.2.11 Let r be a transition-sequence and t a transition of a net N. Then $\mathit{growth\text{-}sites}(r.t) = \{i \in \mathit{growth\text{-}sites}(r) : r[i] \neq t\} \cup \{|r.t|\}$.

Proof. Clearly, event $|r.t|$ is the most-recent firing of transition t in $r.t$. Furthermore, the most recent firing of any other transition t' is the same in r and $r.t$. ∎

It now follows that whether two synchronous runs remain synchronous after firing another pair of transitions depends solely on the labels of these transitions, and on whether the causes of these transitions are the same in the growth-sites of the respective runs. It will be helpful to define a more general *growth-site correspondence* (*gsc*) between causes in growth-sites. To avoid confusion, we introduce the following terminology:

Definition 6.2.12 Let p and q be pomsets and let $f : p \rightarrow q$ be a partial function from Events_p to Events_q. Then p is the *source* of f, written $\mathit{source}(f)$, and q is the *target* of f, written $\mathit{target}(f)$. Furthermore, the *domain-of-definition* of f is the subset of Events_p given by $\{e \in \mathrm{Events}_p : f(e) \text{ is defined}\}$, and the *image* of f is the subset of Events_q given by $\{e' \in \mathrm{Events}_q : f(e) = e' \text{ for some } e \in \mathrm{Events}_p\}$.

Definition 6.2.13 Let $r = t_1 \ldots t_n$ and $r' = t'_1 \ldots t'_m$ be transition-sequences of nets N and N', respectively. Then $\mathit{gsc}(r, r')$ is defined iff r and r' are synchronous. Furthermore, if r and r' are synchronous, then $\mathit{gsc}(r, r')$ is the partial identity function $\beta : \mathit{growth\text{-}sites}(r) \rightarrow \mathit{growth\text{-}sites}(r')$ such that $\beta(i) = j$ iff $i = j$ and $i \in \mathrm{Events}_{\mathit{growth\text{-}sites}(r)} \cap \mathrm{Events}_{\mathit{growth\text{-}sites}(r')}$, *cf.* Figure 6-3. In particular, $\mathit{growth\text{-}sites}(r)$ is the source of $\mathit{gsc}(r, r')$, and $\mathit{growth\text{-}sites}(r')$ is the target of $\mathit{gsc}(r, r')$.

We now state the key observation underlying our decision procedure: the growth-site correspondence of a pair of runs $r.t$ and $r'.t'$ is determined up to isomorphism by the isomorphism class of the growth-site correspondence between r and r'.

Definition 6.2.14 Let β and γ be partial functions whose source and target are pomsets. We say that β and γ are *isomorphic*, written $\beta \approx \gamma$, iff there is a pair of functions (I, J) such that

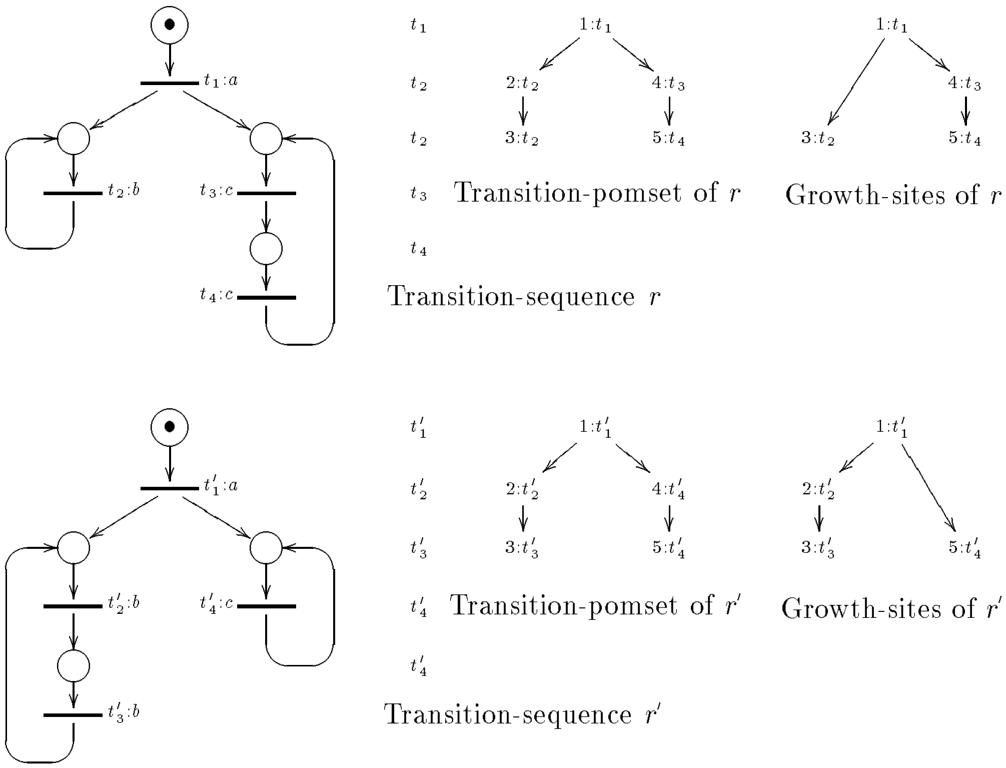
- I is an isomorphism between $\mathit{source}(\beta)$ and $\mathit{source}(\gamma)$,
- J is an isomorphism between $\mathit{target}(\beta)$ and $\mathit{target}(\gamma)$, and
- $\gamma \circ I = J \circ \beta$.

Lemma 6.2.15 Let r_1, r_2 be transition-sequences and t a transition of net N; likewise for r'_1, r'_2, t' of net N'. If $\mathit{gsc}(r_1, r'_1) \approx \mathit{gsc}(r_2, r'_2)$, then $\mathit{gsc}(r_1.t, r'_1.t') \approx \mathit{gsc}(r_2.t, r'_2.t')$.

Proof. Let (I, J) be the isomorphism between $\mathit{gsc}(r_1, r'_1)$ and $\mathit{gsc}(r_2, r'_2)$, noting that both $\mathit{gsc}(r_1, r'_1)$ and $\mathit{gsc}(r_2, r'_2)$ are defined.

We define the function I' to be

$$I'(i) = \begin{cases} |r_2.t| & \text{if } i = |r_1.t| \\ I(i) & \text{if } i \in \mathrm{Events}_{\mathit{growth\text{-}sites}(r_1.t)} \text{ and } i \neq |r_1.t| \end{cases}$$



$$gsc(r, r') \stackrel{\text{def}}{=} \{(1,1), (3,3), (5,5)\}$$

Figure 6-3: An Example of Growth-Sites and Growth-Site Correspondence and define the function J' to be

$$J'(j) = \begin{cases} |r'_2.t'| & \text{if } j = |r'_1.t'| \\ J(j) & \text{if } j \in \text{Events}_{\textit{growth-sites}(r'_1.t')} \text{ and } j \neq |r'_1.t'| \end{cases}$$

By Proposition 6.2.11, I' and J' are total functions on $\text{Events}_{\textit{growth-sites}(r_1.t)}$ and $\text{Events}_{\textit{growth-sites}(r'_1.t')}$, respectively. Definition 6.2.9, Proposition 6.2.11, and the properties of I and J imply that I' is an isomorphism between $\textit{growth-sites}(r_1.t)$ and $\textit{growth-sites}(r_2.t)$, and J' is an isomorphism between $\textit{growth-sites}(r'_1.t')$ and $\textit{growth-sites}(r'_2.t')$. The details are left to the reader.

In order to prove that $gsc(r_2.t, r'_2.t') \circ I' = J' \circ gsc(r_1.t, r'_1.t')$, we first show that $gsc(r_1.t, r'_1.t')$ is defined iff $gsc(r_2.t, r'_2.t')$ is defined. For one direction, suppose that $gsc(r_1.t, r'_1.t')$ is defined; thus, $r_1.t$ and $r'_1.t'$ are synchronous and t and t' have the same label, Furthermore, since $gsc(r_2, r'_2)$ is defined, we have that r_2 and r'_2 are synchronous and $|r_2| = |r'_2|$. By Proposition 6.2.8, it remains to show that the maximal causes of event $|r_2.t|$ are the same in the transition-pomsets of $r_2.t$ and $r'_2.t'$. For one direction, let event k be a maximal cause of event $|r_2.t|$ in the transition-pomset of $r_2.t$; Proposition 6.2.10 implies that $k \in \textit{growth-sites}(r_2)$. Since I is an isomorphism between $\textit{growth-sites}(r_1)$ and $\textit{growth-sites}(r_2)$, it follows that $I^{-1}(k) \in \textit{growth-sites}(r_1)$ and that event $I^{-1}(k)$ is a maximal cause of event $|r_1.t|$ in the transition-pomset of $r_1.t$; the details are straightforward but slightly tedious and are left to the reader. Since $r_1.t$ and $r'_1.t'$ are synchronous, Proposition 6.2.8 implies that event $I^{-1}(k)$ is also a maximal cause of event $|r'_1.t'|$ in the transition-pomset of $r'_1.t'$, $r'_1[I^{-1}(k)]$ and t' are not statically concurrent, and $I^{-1}(k) \in \textit{growth-sites}(r'_1)$. Definitions 6.2.9, 6.2.13, and 6.2.14 and our definition of (I, J) then imply that $r'_1[I^{-1}(k)] = r'_2[J(I^{-1}(k))] = r'_2[k]$, and so $r'_2[k]$ and t' are not statically concurrent; hence, event k must cause event $|r'_2.t'|$ in the transition-pomset of $r'_2.t'$. The other direction is analogous, and so the maximal causes of event $|r_2.t|$ are the same in the transition-pomsets of $r_2.t$ and $r'_2.t'$. Thus, by Proposition 6.2.8, $r_2.t$ and $r'_2.t'$ are synchronous, proving that $gsc(r_2.t, r'_2.t')$ is defined. The proof of the other direction, namely that $gsc(r_1.t, r'_1.t')$ is defined whenever $gsc(r_2.t, r'_2.t')$ is defined, is analogous and omitted.

We now show that $gsc(r_2.t, r'_2.t') \circ I' = J' \circ gsc(r_1.t, r'_1.t')$. For one direction, let i be some event on which $gsc(r_2.t, r'_2.t') \circ I'$ is defined. It then follows by Definition 6.2.13 and the definition of I' that $i \in \textit{growth-sites}(r_1.t)$, $I'(i) \in \textit{growth-sites}(r_2.t) \cap \textit{growth-sites}(r'_2.t')$, and $gsc(r_2.t, r'_2.t')$ is defined; thus, by the above proof, $gsc(r_1.t, r'_1.t')$ is defined, $|r_1.t| = |r'_1.t'|$, and $|r_2.t| = |r'_2.t'|$. For one case, suppose that $i \neq |r_1.t|$; then $I'(i) = I(i) \neq |r'_2.t'|$ and thus by Proposition 6.2.11, $i \in \textit{growth-sites}(r_1)$, $I'(i) \in \textit{growth-sites}(r_2) \cap \textit{growth-sites}(r'_2)$, and $r'_2[I'(i)] \neq t'$. Since by assumption, $gsc(r_1, r'_1)$ and $gsc(r_2, r'_2)$ are defined and $gsc(r_2, r'_2) \circ I = J \circ gsc(r_1, r'_1)$, it follows that $(J \circ gsc(r_1, r'_1))(i) = I'(i)$. Thus, $i \in \textit{growth-sites}(r'_1)$, $I'(i) = J(i)$, and $r'_1[i] = r'_2[J(i)] = r'_2[I'(i)]$, and so $r'_1[i] \neq t'$. Proposition 6.2.11 then implies that $i \in \textit{growth-sites}(r'_1.t')$, and so $J' \circ gsc(r_1.t, r'_1.t')$ is defined on i. Furthermore,

$$(gsc(r_2.t, r'_2.t') \circ I')(i) = (gsc(r_2, r'_2) \circ I)(i) = (J \circ gsc(r_1, r'_1))(i) = (J' \circ gsc(r_1.t, r'_1.t'))(i),$$

proving this case. The other case is similar and is left to the reader. The proof of the other direction is analogous. ∎

The size of the growth-sites of any transition-sequence of a net is obviously bounded by the number of transitions in that net. We can thus easily conclude that the number of isomorphism classes of growth-site correspondences between transition-sequences of nets N and N' is bounded by an exponential in the maximum of the number of transitions in N and N'.

We thus have:

Theorem 6.2.16 For any finite nets N and N', there is a deterministic finite-state automaton recognizing the set of pairs of *synchronous* transition-sequences of N and N'. If m and m' are the number of transitions in N and N', respectively, then the number of states in the automaton is bounded by $c^{\max\{m,m'\}^2}$ for some fixed constant $c > 1$.

Proof. The states of the automaton are the isomorphism classes of growth-site correspondences between transition-sequences of N and N'. A state β moves to a state γ via a pair (t, t') of transitions iff β is the isomorphism class of $gsc(r, r')$ and γ is the isomorphism class of $gsc(r.t, r'.t')$ for some transition-sequences r and r' of N and N', respectively. The start state is the isomorphism class of the empty function, and all states are accepting. By Lemma 6.2.15, this automaton is deterministic.

If $(t_1, t'_1) \ldots (t_k, t'_k)$ is in the language of the automaton, then by Lemma 6.2.15, the final state reached must be the isomorphism class of $gsc(t_1 \ldots t_k, t'_1 \ldots t'_k)$. Hence, this growth-site correspondence is defined, and so $t_1 \ldots t_k$ and $t'_1 \ldots t'_k$ are synchronous. Conversely, if $t_1 \ldots t_k$ and $t'_1 \ldots t'_k$ are synchronous, then all their equal-length prefixes are synchronous, and so $gsc(t_1 \ldots t_i, t'_1 \ldots t'_i)$ is defined for all $0 \leq i \leq k$. Hence, by Lemma 6.2.15 and the definition of the automaton, $(t_1, t'_1) \ldots (t_k, t'_k)$ is in its language. ∎

Since the runs of a finite net are finite-state recognizable by the (necessarily deterministic) transition system of the net itself, and since finite-state recognizable sets are closed under intersection and renaming input symbols, we conclude:

Corollary 6.2.17 For any finite nets N and N', there is a finite-state automaton whose language is the set of runs r of N for which there is some run r' of N' such that r and r' are synchronous. If m and m' are the number of transitions in N and N', respectively, and n and n' are the number of places in N and N', respectively, then the number of states in the automaton is bounded by $d^{\max\{m,m'\}^2 + \max\{n,n'\}}$ for some fixed constant $d > 1$.

Proof. The number of states in the deterministic automaton that recognizes the set of pairs of runs of N and N' is $b^{\max\{n,n'\}}$ for some fixed constant $b > 1$. The intersection of this automaton with that of Theorem 6.2.16 has number of states bounded by $d^{\max\{m,m'\}^2 + \max\{n,n'\}}$ for some fixed constant $d > 1$. Then renaming each input symbol (t, t') by symbol t does not change the number of states and yields the desired automaton. ∎

It is fairly straightforward to show that such an automaton can in fact be constructed in space proportional to the size of its transition table. The desired decidability result then follows as a corollary:

Theorem 6.2.18 The pomset-trace equivalence problem for finite nets without hidden transitions can be decided in space exponential in the number of places and transitions in the nets.

Proof. By Lemma 6.2.6 and Corollary 6.2.17, $N \sqsubseteq_{\mathrm{pt}} N'$ iff the language of the finite-state automaton given in Corollary 6.2.17 is the set of all runs of N. It is easy to construct another finite-state automaton, of essentially the same size, recognizing the runs of N. So $N \sqsubseteq_{\mathrm{pt}} N'$ iff these automata recognize the same language. But language equivalence is checkable in space proportional to the size of the automata [22]. ∎

6.2.2 Nets with Hidden Transitions

We now show how the results above extend to nets which *may contain hidden transitions*. We begin by modifying our definition of "synchronous" to take account of hidden transitions. This new definition will coincide with Definition 6.2.5 for nets without hidden transitions.

Definition 6.2.19 Let $r = t_1 \ldots t_n$ and and $r' = t'_1 \ldots t'_m$ be transition-sequences of nets N and N', respectively.

Let $\alpha_{r,r'}$ be the partial function on the integers such that $\alpha_{r,r'}(i) = j$ iff t_i is the k^{th} transition of r with a visible label and t'_j is the k^{th} transition of r' with a visible label, for some (necessarily unique) k.

Then r and r' are *synchronous* iff $\alpha_{r,r'}$ is an isomorphism between the visible-pomset of r and the visible-pomset of r'.

In particular, if r and r' are synchronous, then they have the same number of occurrences of visible transitions.

Lemma 6.2.6 continues to hold for this generalized notion of synchronous:

Lemma 6.2.20 Let r and r' be runs of nets N and N', respectively. If the pomset-traces of r and r' are isomorphic, then there is some run r'' of N' such that

- the transition-pomsets of r' and r'' are isomorphic, and
- r and r'' are synchronous.

Proof. The proof extends that of Lemma 6.2.6. Let I be the isomorphism between the pomset-trace of r and the pomset-trace of r', and let q and q' respectively be the transition-pomsets of r and r'. Clearly, r and r' must contain the same number, k, of occurrences of transitions with visible labels. For $1 \leq i \leq k$, we define $vis_r(i)$ to be the index of the i^{th} visible transition-occurrence in r; that is, $vis_r(i) = m$, where $r[m]$ is the (necessarily unique) i^{th} transition of r with a visible label. We let v be the sequence of visible transition-occurrences obtained from r' by applying I element-wise to visible transitions of r; that is, $v[i] = r'[I(vis_r(i))]$ for all $1 \leq i \leq k$. We then obtain r'' by "padding" v with sequences w_i of hidden transition-occurrences of r'; each composite sequence $w_1 \ldots w_i$ will contain exactly the hidden transition-occurrences of r' that are necessary for the $v[1], \ldots, v[i]$ to fire. In order to define the w_i, we first define z_i, for $1 \leq i \leq k$, to be the ascending sequence of *indices* of the "remaining" hidden transition-occurrences that causally precede $r'[I(vis_r(i))]$. Furthermore, we define z_{k+1} to be the sequence of indices of "left-over" hidden transition-occurrences of r'.

$$z_i = \text{the ascending sequence over the set}$$
$$\{j <_{q'} I(vis_r(i)) : r'[j] \text{ is a hidden transition and } j \not<_{q'} I(vis_r(n)) \text{ for all } n < i\}$$

$$z_{k+1} = \text{the ascending sequence over the set}$$
$$\{j \leq |r'| : r'[j] \text{ is a hidden transition and } j \not<_{q'} I(vis_r(n)) \text{ for all } n \leq k\}$$

We then define r'' to be the sequence $w_1 v[1] w_2 v[2] \dots v[k] w_{k+1}$, where each w_i is the sequence of transition-occurrences of r' corresponding to z_i; that is, $|w_i| = |z_i|$ and $w_i[n] = r'[z_i[n]]$ for all $1 \leq n \leq |z_i|$. Hence, for all $1 \leq i \leq k$, $r''[vis_{r''}(i)] = v[i] = r'[I(vis_r(i))]$.

Let

$$C(i) = \begin{cases} I(vis_r(vis_{r''}^{-1}(i))) & \text{if } r''[i] \text{ is a visible transition} \\ & \\ m & \begin{array}{l}\text{if for some (necessarily unique) } n \text{ and hidden transition } t \\ \quad r''[i] \text{ is the } n^{\text{th}} \text{ occurrence of } t \text{ in } r'' \text{, and} \\ \quad r'[m] \text{ is the } n^{\text{th}} \text{ occurrence of } t \text{ in } r'\end{array} \end{cases}$$

It is straightforward but tedious to show that C is a label-preserving bijection between the transition-pomsets of r'' and r'; the details are left to the reader.

To show that C is an order-isomorphism, it clearly suffices to show that C and C^{-1} preserve proximate causes. Suppose that event i is a proximate cause of event j in the transition-pomset of r''; then $i < j$ and transition $r''[i]$ and transition $r''[j]$ are not statically concurrent in N'. Then by definition of r'' and C, transition $r'[C(i)]$ and transition $r'[C(j)]$ are not statically concurrent in N'. For one case, suppose that both $r''[i]$ and $r''[j]$ are visible transitions. $C(j) < C(i)$ would imply that event $C(j)$ is a proximate cause of event $C(i)$ in the pomset-trace of r'; since I is an isomorphism between the pomset-trace of r and the pomset-trace of r', it would follow that event $I^{-1}(C(j))$ causes event $I^{-1}(C(i))$ in the pomset-trace of r, and so $I^{-1}(C(j)) < I^{-1}(C(i))$. Clearly, vis_r and $vis_{r''}^{-1}$ are monotone functions, implying that $j < i$, a contradiction. Hence $C(i) < C(j)$, and so event $C(i)$ is a proximate cause of event $C(j)$ in the transition-pomset of r', proving this case.

For another case, suppose that $r''[i]$ is a hidden transition t, and $r''[j]$ is a visible transition. Then for some n, $r''[i]$ is the n^{th} occurrence of t in r'' and $r'[C(i)]$ is the n^{th} occurrence of t in r'. Let n' be the number of occurrences of t preceding $r''[j]$ in r''; clearly, $n' \geq n$ since $i < j$. By definition of r'', $r''[j] = v[vis_{r''}^{-1}(j)]$; hence by definition of the z_i, there are distinct $l_1, \dots, l_{n'}$ in $z_0 \dots z_{vis_{r''}^{-1}(j)}$ such that $r'[l_1], \dots, r'[l_{n'}]$ is each an occurrence of t. Let l be the maximum of $l_1, \dots, l_{n'}$; from the definition of C and the z_i, $l > C(j)$ would imply that there is some $j' < j$ such that $r'[C(j')]$ is a visible transition and $l <_{q'} C(j')$. Then, clearly, $C(j) <_{q'} l <_{q'} C(j')$, and so $I(vis_r(vis_{r''}^{-1}(j))) <_{q'} I(vis_r(vis_{r''}^{-1}(j')))$. Since I is an isomorphism between the pomset-traces of r and r', it would follow that $vis_r(vis_{r''}^{-1}(j)) <_q vis_r(vis_{r''}^{-1}(j'))$, and so $vis_r(vis_{r''}^{-1}(j)) < vis_r(vis_{r''}^{-1}(j'))$. The monotonicity of vis_r and $vis_{r''}^{-1}$ would then imply that $j < j'$, a contradiction. Thus, $l < C(j)$ after all; now, $C(j) < C(i)$ would imply that there are $n' \geq n$ occurrences of t preceding $r'[C(i)]$ in r', contradicting the fact that $r'[C(i)]$ is the n^{th} occurrence of t in r'. Hence $C(i) < C(j)$, and so event $C(i)$ is a proximate cause of event $C(j)$

in the transition-pomset of r', proving this case.

The proofs of the other cases and the other direction are similar, and are left to the reader.

The proof that r'' is a run of N' is identical to that for Lemma 6.2.6.

Clearly, $vis_{r''} \circ vis_r^{-1} \circ I^{-1}$ is an isomorphism between the pomset-trace of r' and the pomset-trace of r''. Pomset isomorphisms are closed under function composition; thus, $vis_{r''} \circ vis_r^{-1} \circ I^{-1} \circ I$ is an isomorphism between the pomset-trace of r and the pomset-trace of r''. It follows easily from the definitions of $\alpha_{r,r''}$, vis_r, and $vis_{r''}$ that $\alpha_{r,r''} = vis_{r''} \circ vis_r^{-1}$, proving that r and r'' are synchronous, and completing the proof of the lemma. ∎

The notion of maximal cause must now be sharpened to be a maximal *visible* cause.

Definition 6.2.21 Let N be a net, let p be a transition-pomset of N, and let $e, e' \in \text{Events}_p$. Event e' is a *maximal visible cause* of event e in p providing $l_p(e')$ is a visible transition of N, $e' <_p e$ and there is no event $e'' \in \text{Events}_p$ such that $l_p(e'')$ is a visible transition of N and $e' <_p e'' <_p e$.

Then Proposition 6.2.8 generalizes as follows:

Proposition 6.2.22 Let r, r' be transition-sequences and let t, t' be *visible* transitions of nets N, N', respectively. Then $r.t$ and $r'.t'$ are synchronous iff

- r and r' are synchronous,
- t and t' have the same label, and
- $\alpha_{r,r'}$ restricted to the maximal visible causes of event $|r| + 1$ in the transition-pomset of $r.t$ is a bijection onto the maximal visible causes of event $|r'| + 1$ in the transition-pomset of $r'.t'$.

Also, if t is a hidden transition, then $r.t$ and r' are synchronous iff r and r' are synchronous.

The proof is completely straightforward and is left to the reader.

The notion of growth-sites extends to hidden transitions as follows:

Definition 6.2.23 Let r be a transition-sequence of a net N. Let *most-recent*(r) be the set of most recent firings in r of each transition. Let *max-visible-causes*(t, r) be the maximal visible causes (in the transition-pomset of r) of the most recent firing in r of transition t. Then *growth-sites*(r) is the restriction of the transition-pomset of r to

$$\textit{most-recent}(r) \cup \bigcup\{\textit{max-visible-causes}(t, r) : t \text{ is a hidden transition}\}.$$

As before, the maximal causes will necessarily be a subset of the events in the growth-sites:

Proposition 6.2.24 Let $r = t_1 \ldots t_n$ be a transition-sequence and t be a visible transition of a net N. Then the maximal causes of event $n + 1$ in the visible-pomset of $r.t$ are a subset of the events of *growth-sites*(r).

Proof. Suppose event i of the visible-pomset of $r.t$ is a maximal cause of event $n+1$. For one case, suppose that event i is also a maximal cause of $n+1$ in the *transition-pomset* of $r.t$; then $i \in$ *most-recent*(r) by a proof identical to that of Proposition 6.2.10. For the other case, there must be some event k in the transition-pomset of $r.t$ such that t_k is a hidden transition, event i causes event k, and event k is a maximal cause of event $n+1$. It follows by the same reasoning as in the proof of Proposition 6.2.10 that event k must be the most-recent firing of transition t_k in r. Therefore, event i not being in *growth-sites*(r) would imply that event i is not a maximal visible cause of event k. There would thus be some event j in the transition-pomset of r such that t_j is a visible transition, event i causes event j, and event j causes event k. But this would contradict event i being a maximal cause of $n+1$ in the visible-pomset of $r.t$. ∎

We now observe that the growth-sites of transition-sequence $r.t$ are fully determined by t, the growth-sites of r, and the static concurrency relation of N:

Proposition 6.2.25 Let r be a transition-sequence and t a transition of a net N. Then an event i is a visible cause of event $|r.t|$ in the transition-pomset of $r.t$ iff $i \in$ *growth-sites*(r), $r[i]$ is a visible transition, and there is some event $j \in$ *growth-sites*(r) such that transition $r[j]$ and t are not statically concurrent, and either event i causes event j in *growth-sites*(r) or $i = j$. Furthermore, an event i is a maximal visible cause of event $|r.t|$ in the transition-pomset of $r.t$ iff event i is a visible cause of event $|r.t|$ in the transition-pomset of $r.t$ and there is no event $k \in$ *growth-sites*(r) such that event i causes event k in *growth-sites*(r) and event k is a visible cause of event $|r.t|$ in the transition-pomset of $r.t$.

The proposition is a straightforward consequence of Proposition 6.2.10; the details are left to the reader.

Proposition 6.2.26 Let $r = t_1 \ldots t_n$ be a transition-sequence of a net N. Then *most-recent*$(r) =$

$$\{i \in \textit{growth-sites}(r) : \text{there is no event } j \in \textit{growth-sites}(r) \text{ such that } j > i \text{ and } l_{\textit{growth-sites}(r)}(i) = l_{\textit{growth-sites}(r)}(j)\}$$

Furthermore, *max-visible-causes*$(t_k, r) =$

$$\{i \in \textit{growth-sites}(r) : \text{there is some event } j \in \textit{most-recent}(r) \text{ such that } l_{\textit{growth-sites}(r)}(j) = t_k \text{ and event } i \text{ is a maximal visible cause of event } j \text{ in } \textit{growth-sites}(r)\}$$

The proposition is a simple consequence of Definition 6.2.23; the details are left to the reader.

Proposition 6.2.27 Let r be a transition-sequence and t a transition of a net N. Then *growth-sites*$(r.t) =$

$$\begin{aligned}\{|r.t|\} \cup \{i \in \textit{growth-sites}(r) :\ & \text{either } i \in \textit{most-recent}(r) \text{ and } r[i] \neq t \\ & \text{or } i \in \textit{max-visible-causes}(t', r) \\ & \quad \text{for some hidden transition } t' \neq t \\ & \text{or } i \in \textit{max-visible-causes}(t, r.t) \\ & \quad \text{and } t \text{ is a hidden transition}\}\end{aligned}$$

Proof. Clearly, event $|r.t|$ is the most-recent firing of transition t in $r.t$, and the most-recent firing of any other transition is the same in r and $r.t$. Furthermore, the maximal visible causes of the most-recent occurrence of any hidden transition other than t are the same in the transition-pomsets of r and $r.t$, from which the highlighted equality immediately follows. ∎

As an immediate consequence of the preceding three propositions, we have:

Proposition 6.2.28 Let r be a transition-sequence and t a transition of a net N. Then $\mathit{growth\text{-}sites}(r.t)$ is fully determined by t, $\mathit{growth\text{-}sites}(r)$, and the static concurrency relation of N.

Our definition of growth-site correspondences is also modified accordingly; this new definition will coincide with Definition 6.2.13 for nets without hidden transitions.

Definition 6.2.29 Let r and r' be transition-sequences of nets N and N', respectively. Then $\mathit{gsc}(r, r')$ is defined iff r and r' are synchronous. Furthermore, if r and r' are synchronous, then $\mathit{gsc}(r, r')$ is the 1-1 partial function
β: $\mathit{growth\text{-}sites}(r) \rightarrow \mathit{growth\text{-}sites}(r')$ such that

$$\mathit{graph}(\beta) = \mathit{graph}(\alpha_{r,r'}) \cap (\mathrm{Events}_{(\mathit{growth\text{-}sites}(r))} \times \mathrm{Events}_{(\mathit{growth\text{-}sites}(r'))}).$$

In particular, $\mathit{growth\text{-}sites}(r)$ is the source of $\mathit{gsc}(r, r')$, and $\mathit{growth\text{-}sites}(r')$ is the target of $\mathit{gsc}(r, r')$.

Again, the growth-site correspondences are significant only up to isomorphism:

Lemma 6.2.30 Let r_1, r_2 be transition-sequences of net N and let r'_1, r'_2 be transition-sequences of net N'. If $\mathit{gsc}(r_1, r'_1) \approx \mathit{gsc}(r_2, r'_2)$, then

- $\mathit{gsc}(r_1.t, r'_1.t') \approx \mathit{gsc}(r_2.t, r'_2.t')$ for any pair of *visible* transitions t and t' of N and N', respectively.
- $\mathit{gsc}(r_1.t, r'_1) \approx \mathit{gsc}(r_2.t, r'_2)$ for any *hidden* transition t of N.
- $\mathit{gsc}(r_1, r'_1.t') \approx \mathit{gsc}(r_2, r'_2.t')$ for any *hidden* transition t' of N'.

The proof is a straightforward but tedious adaptation of the proof of Lemma 6.2.15 and uses Definitions 6.2.1, 6.2.19, 6.2.23, and 6.2.29, and Propositions 6.2.22, 6.2.28, and 6.2.24, instead of the corresponding definitions and propositions in the previous section. The details are left to the reader.

We note that it follows from Definition 6.2.23 that the size of the growth-sites of any transition-sequence of a net is bounded by the square of the number of transitions in that net.

We remark that, in order to allow hidden transitions to move independently, the alphabet of the automaton of Theorem 6.2.16 is generalized to pairs (u, u'), where either u and u' are both *visible* transitions of the respective nets, or exactly one of u and u' is a *hidden* transition of the respective net and the other is a special symbol $\bullet$. We refer to any sequence w of such pairs as a *$\bullet$-pair-sequence*, and for $i = 1, 2$, we write $\mathit{proj}_i(w)$ to denote the projection of w onto its i^{th} component alphabet, with all occurrences of $\bullet$ omitted.

Theorem 6.2.31 For any finite nets N and N', there is a deterministic finite-state automaton recognizing the set of pairs of *synchronous* transition-sequences of N and N'. If m and m' are the number of transitions in N and N', respectively, then the number of states in the automaton is bounded by $c^{\max\{m,m'\}^4}$ for some fixed constant $c > 1$.

Proof. The states of the automaton are the isomorphism classes of growth-site correspondences between transition-sequences of N and N'. A state β moves to a state γ via a pair (t, t') of transitions iff β is the isomorphism class of $gsc(r, r')$ and γ is the isomorphism class of $gsc(r.t, r'.t')$ for some transition-sequences r and r' of N and N', respectively. A state β moves to a state γ via a pair $(t, \bullet)$ iff β is the isomorphism class of $gsc(r, r')$ and γ is the isomorphism class of $gsc(r.t, r')$ for some transition-sequences r and r' of N and N', respectively; a similar definition applies to pairs $(\bullet, t')$. The start state is the isomorphism class of the empty function, and all states are accepting. By Lemma 6.2.30, this automaton is deterministic.

If $w = (u_1, u'_1) \ldots (u_k, u'_k)$ is in the language of the automaton, then by Lemma 6.2.30, the final state reached must be the isomorphism class of $gsc(proj_1(w), proj_2(w))$. Hence, this growth-site correspondence is defined, and so $proj_1(w)$ and $proj_2(w)$ are synchronous. Conversely, if $proj_1(w)$ and $proj_2(w)$ are synchronous, then $gsc(proj_1(w'), proj_2(w'))$ is defined for all prefixes w' of w. Hence, by Lemma 6.2.30 and the definition of the automaton, w is in its language. ∎

As before, we conclude:

Corollary 6.2.32 For any finite nets N and N', there is a finite-state automaton whose language is the set of runs r of N for which there is some run r' of N' such that r and r' are synchronous. If m and m' are the number of transitions in N and N', respectively, and n and n' are the number of places in N and N', respectively, then the number of states in the automaton is bounded by $d^{\max\{m,m'\}^4 + \max\{n,n'\}}$ for some fixed constant $d > 1$.

Proof. The number of states in the deterministic automaton whose alphabet consists of $\bullet$-pairs and that recognizes the set of pairs of runs of N and N' is $b^{\max\{n,n'\}}$ for some fixed constant $b > 1$. The intersection of this automaton with that of Theorem 6.2.31 has number of states bounded by $d^{\max\{m,m'\}^4 + \max\{n,n'\}}$ for some fixed constant $d > 1$. Then renaming each input symbol (t, t') by symbol t, renaming each input symbol $(t, \bullet)$ by t, and renaming each input symbol $(\bullet, t')$ by ε does not change the number of states and yields the desired automaton. ∎

The earlier argument without hidden transitions now carries over:

Theorem 6.2.33 The pomset-trace equivalence problem for finite nets that *may contain hidden transitions* can be decided in space exponential in the number of places and transitions in the nets.

Proof. Since, language equivalence of automata with ε-moves is decidable in space proportional to the size of the automata [22], the proof of the theorem is identical to that of Theorem 6.2.18, except that it uses Lemma 6.2.20 and Corollary 6.2.32. ∎

6.3 History-Preserving Bisimulation and Pomset-Bisimulation

In this section, we assume that all nets may contain τ-labeled transitions. We begin by defining history-preserving bisimulation on nets. Our definition induces the same equivalence as that of [5, 35, 39, 50, 46].

Definition 6.3.1 A set $\mathcal{H}$ of triples of the form (r, r', f) is a *history-preserving bisimulation* between nets N and N' iff

1. If $(r, r', f) \in \mathcal{H}$, then r and r' are runs of N and N', respectively, and f is an isomorphism between *pomset-trace*(r) and *pomset-trace*(r').

2. $(\varepsilon, \varepsilon, \emptyset) \in \mathcal{H}$, where ε is the empty transition-sequence.

3. If $(r, r', f) \in \mathcal{H}$ and $r.t$ is a run of N, then there is some, possibly empty, sequence of transitions $t'_1 \ldots t'_k$ and some function f' such that $((r.t), (r'.t'_1 \ldots t'_k), f') \in \mathcal{H}$ and f' restricted to *pomset-trace*(r) equals f.

4. If $(r, r', f) \in \mathcal{H}$ and $r'.t'$ is a run of N', then there is some, possibly empty, sequence of transitions $t_1 \ldots t_k$ and some function f' such that $((r.t_1 \ldots t_k), (r'.t'), f') \in \mathcal{H}$ and f' restricted to *pomset-trace*(r) equals f.

Vogler [46, 48] has given an alternate characterization of history-preserving bisimulation based on partially ordered sets of *places*, together with a decidability result. We give an alternate proof based on the approach presented in Section 6.2. We recall that the finite automaton described in Theorem 6.2.31 is deterministic, and we let *update* refer to its state-transition function. Furthermore, for any $\bullet$-pair-sequence w and any gsc β, we write *update*(β, w) to mean the successive application of *update* to each of the pairs in w. For any net N, we write *init*(N) to denote the initial marking of N.

Definition 6.3.2 A set $\mathcal{G}$ of triples of the form (M, M', β) is an *gsc-bisimulation* between nets N and N' iff

1. If $(M, M', \beta) \in \mathcal{G}$, then M and M' are markings of N and N', respectively, and β is an isomorphism class of growth-site correspondences between N and N'.

2. $(\mathit{init}(N), \mathit{init}(N'), \emptyset) \in \mathcal{G}$.

3. If $(M, M', \beta) \in \mathcal{G}$ and $M[t\rangle M_1$ for some transition t and some marking M_1, then there is some marking M'_1 and some $\bullet$-pair-sequence w such that $\mathit{proj}_1(w) = t$, $M'[\mathit{proj}_2(w)\rangle M'_1$ and $(M_1, M'_1, \mathit{update}(\beta, w)) \in \mathcal{G}$.

4. Vice-versa; if $(M, M', \beta) \in \mathcal{G}$ and $M'[t'\rangle M'_1$ for some transition t' and some marking M'_1, then there is some marking M_1 and some $\bullet$-pair-sequence w such that $\mathit{proj}_2(w) = t'$, $M[\mathit{proj}_1(w)\rangle M_1$ and $(M_1, M'_1, \mathit{update}(\beta, w)) \in \mathcal{G}$.

Lemma 6.3.3 Nets are history-preserving bisimilar iff they are *gsc*-bisimilar.

Proof. For one direction, let $\mathcal{H}$ be a history-preserving bisimulation between nets N and N'. Let

$$\mathcal{G} = \{(M, M', gsc(r, r')) : (r, r', gsc(r, r')) \in \mathcal{H},\ init(N)[r\rangle M \text{ and } init(N')[r'\rangle M'\}.$$

Property (1) and (2) of Definition 6.3.2 follow easily from Definition 6.2.29 and Definition 6.3.1; the details are left to the reader. To prove property (3), let $(M, M', \beta) \in \mathcal{G}$ and let transition t and marking M_1 be such that $M[t\rangle M_1$. Clearly, there must be some $(r, r', gsc(r, r')) \in \mathcal{H}$ such that $\beta = gsc(r, r')$, $init(N)[r\rangle M$, and $init(N')[r'\rangle M'$. By Definition 6.3.1, $r.t$ is a run of N, and so property (3) of Definition 6.3.1 implies the existence of some, possibly empty, sequence of transitions $t'_1 \ldots t'_k$ and some function f' such that $((r.t), (r'.t'_1 \ldots t'_k), f') \in \mathcal{H}$ and f' restricted to $pomset\text{-}trace(r)$ equals $gsc(r, r')$. Definition 6.3.1 implies that f' is an isomorphism between the pomset-traces of $r.t$ and $r'.t'_1 \ldots t'_k$, from which it then follows easily from Definition 6.2.29 that $f' = gsc(r.t, r'.t'_1 \ldots t'_k)$. The definition of $\bullet$-sequences, the definition of *update*, and the definition of $\mathcal{G}$ then immediately imply that property (3) of Definition 6.3.2 must hold for $\mathcal{G}$. A similar proof holds for property (4), and hence $\mathcal{G}$ is a gsc-bisimulation.

For the other direction, let $\mathcal{G}$ be a *gsc*-bisimulation between nets N and N'. We define the set of triples $\mathcal{H}$ inductively as follows. For the basis step, let $\mathcal{H} = \{(\varepsilon, \varepsilon, \emptyset)\}$. For one inductive step, if $(r, r', f) \in \mathcal{H}$, and for some t, $t'_1 \ldots t'_k$,

1. $r.t$ is a run of N, $r'.t'_1 \ldots t'_k$ is a run of N', and
2. $(M, M', gsc(r.t, r'.t'_1 \ldots t'_k)) \in \mathcal{G}$, where $init(N)[r.t\rangle M$ and $init(N')[r'.t'_1 \ldots t'_k\rangle M'$,

then $(r.t, r'.t'_1 \ldots t'_k, \alpha_{r.t, r'.t'_1 \ldots t'_k}) \in \mathcal{H}$.

For the other inductive step, if $(r, r', f) \in \mathcal{H}$, and for some $t_1 \ldots t_k$, t',

1. $r.t_1 \ldots t_k$ is a run of N, $r'.t'$ is a run of N', and
2. $(M, M', gsc(r.t_1 \ldots t_k, r'.t')) \in \mathcal{G}$, where $init(N)[r.t_1 \ldots t_k\rangle M$ and $init(N')[r'.t'\rangle M'$,

then $(r.t_1 \ldots t_k, r'.t', \alpha_{r.t_1 \ldots t_k, r'.t'}) \in \mathcal{H}$.

By the definition of *gsc* and the α, it is clear that properties (1) and (2) of Definition 6.3.1 hold for $\mathcal{H}$. To prove (3), suppose that $(r, r', f) \in \mathcal{H}$ and $r.t$ is a run of N. Then $(M, M', gsc(r, r')) \in \mathcal{G}$, where $init(N)[r\rangle M$ and $init(N')[r'\rangle M'$. Let M_1 be the marking such that $init(N)[r.t\rangle M_1$. Then by the definition of *gsc*-bisimulations, there is some marking M'_1 and some $\bullet$-pair-sequence w such that $proj_1(w) = t$, $M'[proj_2(w)\rangle M'_1$ and $(M_1, M'_1, update(gsc(r, r'), w) \in \mathcal{G}$. Let $proj_2(w) = t'_1 \ldots t'_k$; then by definition, $update(gsc(r, r'), w)$ is isomorphic to $gsc(r.t, r'.t'_1 \ldots t'_k)$, so $(r.t, r'.t'_1 \ldots t'_k, \alpha_{r.t, r'.t'_1 \ldots t'_k}) \in \mathcal{H}$. It is easy to see by the definition of α that $\alpha_{r.t, r'.t'_1 \ldots t'_k}$ restricted to the pomset-trace of r is equal to $\alpha_{r,r'}$, which is in turn equal to f, proving this case. The proof of (4) is analogous. ∎

As in Section 6.2.2, it is easy to see that for any finite net, the number of triples (M, M', β) is bounded by an exponential in the sizes of the nets. We use this fact in our decision procedure:

Theorem 6.3.4 For finite nets that may contain hidden transitions, history-preserving bisimulation can be decided in deterministic time exponential in the number of places and transitions in the nets.

Proof. The algorithm to decide history-preserving bisimulation of nets N and N' is similar to the decision procedure for (interleaving) bisimulation by successive refinement. We start with a set $\mathcal{G}_0$ that contains all possible triples, and each step, we shrink this set. Specifically, we define inductively:

$$\begin{array}{lcl} \mathcal{G}_0 & = & \{(M, M', \beta) : M, M' \text{ are markings of } N, N', \\ & & \qquad\qquad\qquad \text{and } \beta \text{ is a } gsc\text{-isomorphism class between } N \text{ and } N'\} \\[1em] \mathcal{G}_{i+1} & = & \{(M, M', \beta) \in \mathcal{G}_i : \text{ for every transition } t \text{ and marking } M_1 \text{ with } M[t\rangle M_1, \\ & & \qquad\qquad\qquad \text{there is some marking } M_1' \text{ and some } \bullet\text{-pair-sequence } w \\ & & \qquad\qquad\qquad \text{such that } proj_1(w) = t,\ M'[proj_2(w)\rangle M_1', \\ & & \qquad\qquad\qquad \text{and } (M_1, M_1', update(\beta, w)) \in \mathcal{G}_i \\ & & \qquad\qquad\quad \text{and vice-versa}\} \end{array}$$

We now show that N and N' are gsc-bisimilar iff

$$(init(N), init(N'), \emptyset) \in \mathcal{G}_k$$

for any k that exceeds the number of possible triples (M, M', β). For one direction, let $\mathcal{G}'$ be a gsc-bisimulation between N and N'. Using Definition 6.3.2, a simple induction on i shows that $\mathcal{G}' \subseteq \mathcal{G}_i$ for all $i \geq 0$. Since Definition 6.3.2 implies that $(init(N), init(N'), \emptyset) \in \mathcal{G}'$, we have that $(init(N), init(N'), \emptyset) \in \mathcal{G}_k$, as desired. For the other direction, we observe that for all i, $\mathcal{G}_{i+1}$ is either a strict subset of $\mathcal{G}_i$ or $\mathcal{G}_i = \mathcal{G}_j$ for all $j > i$. Since k is greater than the number of triples, it immediately follows that $\mathcal{G}_k = \mathcal{G}_{k+1}$. Thus, by Definition 6.3.2 and the definition of the $\mathcal{G}_i$, $\mathcal{G}_k$ is a gsc-bisimulation whenever it contains $(init(N), init(N'), \emptyset)$.

We observe that k is easily bounded by an exponential in the sizes of N and N'. It is also easy to check that $\mathcal{G}_k$ can be computed in DEXPTIME in the size of N and N' (using a transitive closure technique as in [26] to calculate the existence of a $\bullet$-pair-sequence w). Thus, it can be checked in deterministic time exponential in the number of places and transitions in N and N' whether $(init(N), init(N'), \emptyset) \in \mathcal{G}_k$, and hence the theorem follows easily from Lemma 6.3.3. ∎

We now define pomset-bisimulation. Our definition induces the same equivalence as that of [6, 39, 50].

Definition 6.3.5 A set $\mathcal{P}$ of pairs of the form (M, M') is a *pomset-bisimulation* between nets N and N' iff

1. If $(M, M') \in \mathcal{P}$, then M and M' are markings of N and N', respectively.

2. $(init(N), init(N')) \in \mathcal{P}$.

3. If $(M, M') \in \mathcal{P}$ and $M[r\rangle M_1$ for some transition-sequence r and some marking M_1, then there is some transition-sequence r' and some marking M_1' such that the pomset-traces of r and r' are isomorphic, $M'[r'\rangle M_1'$, and $(M_1, M_1') \in \mathcal{P}$.

4. Vice-versa; if $(M, M') \in \mathcal{P}$ and $M'[r'\rangle M_1'$ for some transition-sequence r' and some marking M_1', then there is some transition-sequence r and some marking M_1 such that the pomset-traces of r and r' are isomorphic, $M[r\rangle M_1$, and $(M_1, M_1') \in \mathcal{P}$.

Theorem 6.3.6 For finite nets that may contain hidden transitions, pomset-bisimulation can be decided in space exponential in the number of places and transitions in the nets.

Proof. The algorithm to decide pomset-bisimulation of nets N and N' is also by successive refinement. We start with a set $\mathcal{P}_0$ that contains all possible pairs, and each step, we shrink this set. Specifically, we define inductively:

$$\mathcal{P}_0 = \{(M, M') : M, M' \text{ are markings of } N, N'\}$$

$$\begin{aligned}\mathcal{P}_{i+1} = \{(M, M') \in \mathcal{P}_i : \ & \text{for every transition-sequence } r \text{ and marking } M_1 \text{ with } M[r\rangle M_1, \\ & \quad \text{there is some transition-sequence } r' \text{ and some marking } M_1' \\ & \quad \text{such that the pomset-traces of } r \text{ and } r' \text{ are isomorphic,} \\ & \quad M'[r'\rangle M_1', \text{ and } (M_1, M_1') \in \mathcal{P}_i \\ & \text{and vice-versa}\}\end{aligned}$$

It is straightforward to show that N and N' are pomset-bisimilar iff

$$(init(N), init(N')) \in \mathcal{P}_k$$

for any k that exceeds the number of pairs, and this number is easily bounded by an exponential in the sizes of N and N'. To compute each $\mathcal{P}_{i+1}$, we use the following straightforward modification of the decision procedure for pomset-trace equivalence. For each pair $(M, M') \in \mathcal{P}_i$, let N_M be N, except that the initial marking of N_M is M (rather than $init(N)$); net $N'_{M'}$ is defined similarly. As in the proof of Corollary 6.2.32, we intersect the automaton that recognizes the set of pairs of runs of N_M and $N'_{M'}$ with the automaton of Theorem 6.2.31 constructed for N_M and $N'_{M'}$. Each state of the resulting automaton is a pair of the form $(\beta, (M_1, M_1'))$, where M_1 is a state of N_M and M_1' is a state of $N'_{M'}$. For each state $(\beta, (M_1, M_1'))$, we now add a new M_1-labeled transition iff $(M_1, M_1') \in \mathcal{P}_i$; all such transitions lead to a single new, accepting state. All other states of the automaton are defined to be non-accepting. We then relabel the other transitions (u, u') as in the proof of Corollary 6.2.32. Thus, the language of this automaton is all pairs (r, M_r) of runs r and corresponding final marking M_r of N_M for which there is some run r' and corresponding final marking $M'_{r'}$ of $N'_{M'}$ such that r and r' are synchronous and $(M_r, M'_{r'}) \in \mathcal{P}_i$. It is easy to see that the transition table of this modified automaton remains exponential in the sizes of N and N'. (An similar automaton is also constructed whose language is all pairs $(r', M'_{r'})$ of runs r' and corresponding final marking $M'_{r'}$ of $N'_{M'}$ for which there is some run r and corresponding final marking M_r of N_M such that r and r' are synchronous and $(M_r, M'_{r'}) \in \mathcal{P}_i$.)

By Proposition 6.2.4, Definition 6.2.19, and Lemma 6.2.20, it is then straightforward to show that $(M, M') \in \mathcal{P}_{i+1}$ iff (1) the language of the finite-state automaton given above is the set of all pairs (r, M_r) such that r is a run of N_M and $M[r\rangle M_r$, and (2) the language of the similar automaton constructed for $N'_{M'}$ is the set of all pairs $(r', M_{r'})$ such that r' is a run of $N'_{M'}$ and $M'[r'\rangle M_{r'}$. It is easy to construct other finite-state automata of essentially the same size, recognizing the set of such pairs (r, M_r) or the set of such pairs $(r', M_{r'})$. So $(M, M') \in \mathcal{P}_{i+1}$ iff each of the two appropriate pairs of automata recognize the same language. Since language equivalence is checkable in space proportional to the size of the automata [22], each P_i can be computed in space exponential in the size of N and N', and hence so can P_k. ∎

6.4 Deciding Other True Concurrency Equivalences

Since the transition system of a net is a finite-state automaton, the decision procedures for the interleaving trace, failure and bisimulation equivalences for nets follow directly from the results of Kanellakis&Smolka [26] for finite-state automata.

Theorem 6.4.1 For finite nets that may contain hidden transitions, the trace equivalence problem and the failure equivalence problem can be decided in *space* which is a product of an *exponential* in the number of *places* in the nets and a *polynomial* in the number of *transitions* in the nets. Furthermore, the bisimulation problem, the delay bisimulation problem, and the branching bisimulation problem can be decided in *deterministic time* which is a product of an *exponential* in the number of *places* in the nets and a *polynomial* in the number of *transitions* in the nets.

Proof. The transition system of a finite net is a deterministic finite-state automaton whose states correspond to the reachable markings of the net and whose transitions correspond to transitions of the net. Let m and m' be the number of transitions in N and N', respectively, and let n and n' be the number of places in N and N', respectively. Then the maximum of the number of transitions in these automata is bounded by $m \cdot 2^{\max\{n,n'\}}$, and the maximum of the number of states in these automata is bounded by $2^{\max\{n,n'\}}$. Clearly, relabeling each visible transition t with the *label* of t and relabeling each hidden transition t' with ε does not change the sizes of the automata.

By definition, the finite nets are trace, failures, or bisimulation equivalent iff these finite-state automata with ε-moves are respectively trace, failures, or bisimulation equivalent. Trace equivalence of finite-state automata is checkable in space proportional to the size of the automata [26], while bisimulation equivalence is checkable in PTIME [26], as are delay bisimulation and branching bisimulation [17]. The decision procedure for divergence-respecting failures equivalence [9] of finite-state automata is a straightforward generalization of Kannelakis&Smolka's PSPACE decision procedure for divergence-blind failures equivalence. ∎

The decision procedures for most of the other true concurrency equivalences in Table 6.1 then follow from reductions to the corresponding interleaving equivalences, which are part of known full abstraction proofs [23, 25, 47, 49].

Theorem 6.4.2 For finite nets that may contain hidden transitions, the step-trace equivalence problem and the step-failure equivalence problem can be decided in space exponential in the number of places and transitions in the nets. Furthermore, the step-bisimulation problem can be decided in deterministic time exponential in the number of places and transitions in the nets.

Proof. By a known full abstraction result [25], there is a context $C[\cdot]$ involving only a self-synchronization operator [25] such that nets N and N' are step-trace, step-failures, or step-bisimulation equivalent iff the nets $C[N]$ and $C[N']$ are respectively trace equivalent, failures equivalent, or bisimulation equivalent. In particular, $C[\cdot]$ adds a new transition for every set of pairwise statically concurrent transitions, and does not add any new places.

Let m and m' be the number of transitions in N and N', respectively, and let n and n' be the number of places in N and N', respectively. Then the maximum of the number of transitions in $C[N]$ and $C[N']$ is bounded by $2^{\max\{m,m'\}}$, and the maximum of the number of places in $C[N]$ and $C[N']$ is bounded by $\max\{n, n'\}$. The proof then follows easily by Theorem 6.4.1. ■

The decision procedure for interval-pomset-trace equivalence and interval-pomset-failure equivalence relies on a full abstraction result involving action refinement:

Theorem 6.4.3 For finite nets that may contain hidden transitions, the interval-pomset-trace equivalence problem and the interval-pomset-failures equivalence problem can be decided in space exponential in the number of places and transitions in the nets.

Proof. By known full abstraction results [23, 47], there is a context $C[\cdot]$ built from split and choice refinements such that nets N and N' are interval-pomset-trace equivalent or interval-pomset-failures equivalent iff the nets $C[N]$ and $C[N']$ are respectively trace equivalent or failures equivalent. In particular, $C[\cdot]$ refines every visible transition by the net $a_1^+.a_1^- + \ldots + a_k^+.a_k^-$, where a is the label of the visible transition and k is bounded by the maximum of the number of transitions in N and N'.

Let m and m' be the number of transitions in N and N', respectively, and let n and n' be the number of places in N and N', respectively. Then the maximum of the number of transitions in $C[N]$ and $C[N']$ is bounded by $2 \cdot \max\{m, m'\}^2 + 1$, and the maximum of the number of places in $C[N]$ and $C[N']$ is bounded by $\max\{n, n'\} + \max\{m, m'\}^2$. The proof then follows easily by Theorem 6.4.1. ■

Vogler [49] has shown that the interval-pomset equivalences coincide with the ST-equivalences [39, 42]. We have as an immediate consequence:

Theorem 6.4.4 For finite nets that may contain hidden transitions, the ST-trace equivalence problem and the ST-failure equivalence problem can be decided in space exponential in the number of places and transitions in the nets. Furthermore, the ST-bisimulation problem can be decided in deterministic time exponential in the number of places and transitions in the nets.

Proof. The proofs for ST-traces and ST-failures is identical to that of Theorem 6.4.3, while the proof for ST-bisimulation uses the same context $C[\cdot]$ to yield a reduction to bisimulation. The desired upper bound then follows by Theorem 6.4.1. ■

Using the decision procedure for history-preserving bisimulation, a similar result holds for maximality-preserving bisimulation [13]:

Theorem 6.4.5 For finite nets that may contain hidden transitions, the maximality-preserving bisimulation problem can be decided in deterministic time exponential in the number of places and transitions in the nets.

Proof. Let $C[\cdot]$ be the net context involving split and choice refinements given in the proof of Theorem 6.4.3. Then by a proof similar to that of [45], nets N and N' are maximality-preserving bisimilar iff the nets $C[N]$ and $C[N']$ are history-preserving bisimilar. The theorem is then a simple consequence of Theorem 6.3.4. ∎

Lastly, our decision procedure for pomset-bisimulation yields one for pomset-ST-bisimulation [50]:

Theorem 6.4.6 For finite nets that may contain hidden transitions, the pomset-ST-bisimulation problem can be decided in space exponential in the number of places and transitions in the nets.

Proof. Let $C[\cdot]$ be the net context involving split and choice refinements given in the proof of Theorem 6.4.3. Then by a proof similar to that of [45], nets N and N' are pomset-ST-bisimilar iff the nets $C[N]$ and $C[N']$ are pomset-bisimilar. The theorem is then a simple consequence of Theorem 6.3.6. ∎

6.5 Lower Bounds

The lower bounds for trace equivalence and bisimulation essentially follow from previous results of Mayer&Stockmeyer on Mazurkiewicz nets and regular expressions with interleaving. In particular, Mayer&Stockmeyer [29] have shown the EXPSPACE-hardness of deciding whether the language of a regular expression with interleaving is Σ^*. Our EXPSPACE lower bound for trace equivalence of finite 1-safe Petri nets follows by a polynomial-time reduction. For expository simplicity, we first give the proof for nets that may contain hidden transitions.

Theorem 6.5.1 The problem of deciding whether the language of a regular expression with interleaving is Σ^* is polynomial-time reducible to trace equivalence of finite nets *that may contain* hidden transitions.

Proof. Let Σ be a finite alphabet consisting only of visible labels, and let $\surd \notin \Sigma$ be a visible label. For any regular expression r over Σ built from $\{\cup, *, \cdot, ||\}$, we give an inductive translation to finite 1-safe nets with labels from $\Sigma \cup \{\tau, \surd\}$. Each of these nets will have exactly one $\surd$-labeled transition, and the post-set of this transition will be empty.

The translation, *net*, uses net operators defined in [23]; we do not repeat the definitions here. However, we slightly modify the internal choice operator presented there to ensure that the resulting nets always have exactly one $\surd$-labeled transition. This in turn guarantees that the translation *net* can be performed in polynomial-time; that is, for any regular expression r with interleaving, the net $net(r)$ can be constructed in deterministic time polynomial in the number of symbols in r.

For every $a \in \Sigma$, a is the net corresponding to $a.\surd$. The $\cdot$ operator is modeled by the sequencing operator on nets. The $*$ operator applied to a net N adds the initially marked places of N to the post-set of its $\surd$-labeled transition, relabels the $\surd$-transition with τ, and hooks up a single new $\surd$-labeled transition to the set of initially marked places of N. The union operator applied to nets N and N' is modeled by the internal choice operator on nets except that in addition, the $\surd$-labeled transitions of N and N' are relabeled by τ, one common new place is added to the postset of both of these relabeled transitions, and this new place feeds into a new $\surd$-labeled transition. The interleaving operator applied to nets N and N' is modeled by the non-communicating parallel composition operator on nets, in which N and N' are simply placed side by side but required to synchronize on $\surd$-labeled transitions. We note that since all nets in the target of *net* have exactly one $\surd$-labeled transition, the non-communicating parallel composition operator takes only a trivial cross-product of the $\surd$-labeled transitions and hence adds no extra transitions (or places). This ensures that *net* is a polynomial-time translation in the length of r.

It is straightforward to show by induction that each of the nets in the target of *net* will immediately reach a deadlocked state whenever its (necessarily unique) $\surd$-labeled transition fires. Furthermore, this $\surd$-labeled can be fired from any reachable marking, after first performing a finite, possibly empty, sequence of other transitions. For any regular expression r with interleaving, it follows by a simple induction that

$$L(r) = \{v \in \Sigma^* \mid v\surd \text{ is a trace of } net(r)\},$$

where $L(r)$ is the language of r.

Let N_{Σ^*} be the finite net with exactly $|\Sigma|+1$ transitions, each uniquely labeled from $\Sigma \cup \{\surd\}$, and exactly one place, which is initially marked and is in the preset of all the transitions and in the post-set of all the transitions not labeled with $\surd$. The set of traces of N_{Σ^*} is the prefix closure of $\Sigma^* \cdot \surd$. We show that for any regular expression r with interleaving, $L(r) = \Sigma^*$ iff $net(r)$ and N_{Σ^*} are trace equivalent. One direction follows immediately from the equality highlighted above. For the other direction, suppose $L(r) = \Sigma^*$. Since firing the $\surd$-labeled transition immediately puts $net(r)$ in a deadlocked state, clearly the traces of $net(r)$ are contained in the traces of N_{Σ^*}. For the reverse containment, it follows immediately from the highlighted equality that the set $\Sigma^* \cdot \surd$ is contained in the traces of $net(r)$. Since traces are prefix-closed, the set Σ^* is also contained in the traces of $net(r)$, and so $net(r)$ and N_{Σ^*} are trace-equivalent.

This is a polynomial-time reduction from deciding whether the language of a regular expressions with interleaving is Σ^* to trace equivalence of finite nets with hidden transitions. ∎

We then have as a corollary:

Theorem 6.5.2 For finite nets that may contain hidden transitions, trace equivalence is EXPSPACE-hard.

We now modify the proof of Theorem 6.5.1 to yield the lower bound for trace equivalence of finite nets *without* hidden transitions.

Theorem 6.5.3 The problem of deciding whether the language of a regular expression with interleaving is Σ^* is polynomial-time reducible to trace equivalence of finite nets *without* hidden transitions.

Proof. Let *net* be the translation defined in the proof of Theorem 6.5.1, and let $\mathbf{1} \notin (\Sigma^* \cup \{\surd\})$ be a visible label. For any regular expression r with interleaving, we define a new translation *Net* from $net(r)$ as follows: first, we relabel all τ-labeled transitions in $net(r)$ with the label $\mathbf{1}$, then for every place s in $net(r)$, we add a new $\mathbf{1}$-labeled transition and put it in the preset and postset of the place s (*i.e.*, in a self-loop under s). $Net(r)$ is defined to be the resulting net, and clearly can be constructed in polynomial time in the length of r. Furthermore, $Net(r)$ satisfies all the properties of $net(r)$ specified in the proof of Theorem 6.5.1 concerning markings and $\surd$-labeled transitions. The labeled transition system of $Net(r)$ is identical to that of *net*(r), except that all τ-labeled transitions are replaced by $\mathbf{1}$-labeled transitions, and every state has a $\mathbf{1}$-labeled transition trivially looping back to itself.

It is straightforward to show by induction that for any regular expression r with interleaving, *net*(r) can perform at most $4 \cdot |r|$ consecutive τ-moves, where $|r|$ is the number of symbols in r. By construction of *Net*(r), it then follows that:

$$L(r) = \{a_1 \ldots a_k \in \Sigma^* \mid \mathbf{1}^{4\cdot|r|} a_1 \mathbf{1}^{4\cdot|r|} \ldots a_k \mathbf{1}^{4\cdot|r|} \surd \text{ is a trace of } \mathit{Net}(r)\}.$$

For any regular expression r with interleaving, let N_r be the finite net with $4 \cdot |r| + |\Sigma| + 1$ transitions and $4 \cdot |r| + 1$ places, whose set of traces is the prefix-closure of $(\mathbf{1}^{4\cdot|r|} \cdot \Sigma)^* \cdot \mathbf{1}^{4\cdot|r|} \cdot \surd$; the intended definition of N_r is obvious and omitted. By reasoning similar to that of the proof of Theorem 6.5.1, it follows that $L(r) = \Sigma^*$ iff the set of traces of *Net*(r) *contains* the set of the traces of N_r. The details are left to the reader.

To reduce trace-containment to trace equivalence, we observe that for any nets N_1 and N_2, the set of traces of N_1 contains the set of traces of N_2 iff the net $(N_1 \;||_{\Sigma\cup\{\surd,\mathbf{1}\}}\; N_2)$ and the net N_2 are trace equivalent, where $||_{\Sigma\cup\{\surd,\mathbf{1}\}}$ is a parallel composition operator which requires synchronization on (visible) labels and hence corresponds to trace intersection. Furthermore, the size of $N_1 \;||_{\Sigma\cup\{\surd,\mathbf{1}\}}\; N_2$ is polynomial in the sizes of N_1 and N_2, giving a polynomial-time reduction from trace containment to trace equivalence, and proving the theorem. ∎

We then have as a corollary:

Theorem 6.5.4 For finite nets *without* hidden transitions, trace equivalence is EXPSPACE-hard.

Using these results, we obtain a lower-bound for failures equivalence; the proof is very similar to that of Kanellakis&Smolka [26] for finite-state automata.

Theorem 6.5.5 For finite nets without hidden transitions, trace equivalence is polynomial-time reducible to failures equivalence.

Proof. For any finite nets N_1 and N_2 without hidden transitions, let N_i' be constructed by adding to N_i a single new, initially marked place, s_{new}, which is placed in the preset and post-set of every transition of N_i. The labeled transition system of N_i' is isomorphic to that of N_i. Now, N_i'' is constructed by adding to N_i' a new a-labeled transition t_a, for every visible label a, and hooking up each t_a so that its post-set is empty and its preset contains only the place s_{new}. All of the t_a are enabled under every reachable marking of N_i', and firing any one of them puts N_i'' in a deadlocked state.

N_1 and N_2 are trace equivalent iff N_1'' and N_2'' are failures equivalent; the proof is identical to that of Kanellakis&Smolka [26] and is omitted. This is a polynomial-time reduction from trace equivalence to failures equivalence. ∎

We then have as a corollary:

Theorem 6.5.6 Failures equivalence of finite nets is EXPSPACE-hard.

Our proof of a DEXPTIME lower bound for bisimulation is a simple adaptation of Stockmeyer's result [36] for Mazurkiewicz nets: namely, we reduce the acceptance problem for polynomial-space Alternating Turing Machines to the bisimulation problem for finite 1-safe Petri nets. In particular, we simulate the tape and finite-state control of polynomial-space Alternating Turing Machines by polynomial-time constructible 1-safe Petri Nets, and our reduction to bisimulation is essentially identical to that of Stockmeyer. Since Mazurkiewicz nets are somewhat more succinct than 1-safe Petri Nets, our lower bound for bisimulation is a minor technical improvement of the results of Stockmeyer.

Theorem 6.5.7 The acceptance problem for polynomial-space Alternating Turing Machines is polynomial-time reducible to bisimulation of finite nets.

Proof. Let A be an alternating Turing Machine that, for some polynomial p, uses $p(n)$ space on input of size n. A well-known property of polynomial-space alternating Turing Machines is that every computation halts in deterministic time exponential in the size of the input [11, 27]. Let $p'(n)$ be so large that $2^{p'(n)}$ exceeds the time bound of A on input of size n, and let Σ be the finite tape alphabet of A. We can assume without loss of generality that A begins in an existential state, existential and universal states alternate at every step, and when A enters an accepting state it continues to take steps while staying in accepting states. Furthermore, we can assume that A has exactly two possible moves at every step, every existential state has at least one immediate successor that is a rejecting universal state, every universal state has at least one immediate successor that is an accepting existential state, and the final state of every computation is an existential state.

For any input x, we first construct a polynomial-size Petri Net $net(A_x)$ that "simulates" the computation of A on x. Each tape square i of A is represented as a group of places $\{s_{(i,a_1)}, \ldots, s_{(i,a_k)}\} \cup \{s_{(i,q_0)}, \ldots, s_{(i,q_l)}\}$, where $\Sigma = \{a_1, \ldots, a_k\}$ and $\{q_0, \ldots, q_l\}$ are the control states of A. The idea is that for each tape square i, exactly one of the places in $\{s_{(i,a_1)}, \ldots, s_{(i,a_k)}\}$ will be marked under every reachable marking, indicating which tape symbol is currently written on tape square i. Furthermore, over all $1 \leq i \leq p(n)$ and all $0 \leq j \leq l$, exactly one of $s_{(i,q_j)}$ is marked, indicating which tape square holds the head and which control state A is currently in. Let $x = a_{i_1} \ldots a_{i_n}$; then exactly the places $\{s_{(1,a_{i_1})}, \ldots, s_{(n,a_{i_n})}\} \cup \{s_{(1,q_0)}\}$ are initially marked.

The net $net(A_x)$ is wired up as follows: for every tape square i, every control state q, every symbol $a_j \in \Sigma$, and every control transition $(q', a_{j'}, D) \in \delta(q, a_j)$ in A, where D is either L or R, $net(A)$ contains a transition $t^i_{(q,a_j)\rightarrow(q',a_{j'},D)}$, labeled with some common label $\mathbf{1}$. The idea is that this transition fires iff A is currently in control state q and tape square i holds the head and contains a_j. Firing this transition puts A in control state q', writes $a_{j'}$ on tape square i, and moves the head to tape square $i-1$ if $D = L$ and to tape square $i+1$ if $D = R$. In particular, the preset of transition $t^i_{(q,a_j)\rightarrow(q',a_{j'},D)}$ is $\{s_{(i,q)}, s_{(i,a_j)}\}$ and the post-set is $\{s_{(i-1,q')}, s_{(i,a_{j'})}\}$ or $\{s_{(i+1,q')}, s_{(i,a_{j'})}\}$ depending on whether D is L or R. Finally, for every *accepting existential* control state q and tape square i, we introduce a transition $X_{(i,q)}$ with preset $\{s_{(i,q)}\}$, empty postset, and label **acc**. For every *rejecting existential* control state q and tape square i, we introduce a transition $X_{(i,q)}$ with preset $\{s_{(i,q)}\}$, empty postset, and label **acc**, *and* a transition $Y_{(i,q)}$ with preset $\{s_{(i,q)}\}$, empty postset, and label **rej**. Clearly, $net(A_x)$ contains $(k+l) \cdot p(n)$ places and at most $(2l+m) \cdot p(n)$ transitions, where k is the size of the tape alphabet of A, l is the number of control states of A, and m is the number of control transitions of A.

It is straightforward to show that $net(A_x)$ is 1-safe, sequential (*i.e.*, no transitions can fire concurrently under any reachable marking), and that its labeled transition system is isomorphic

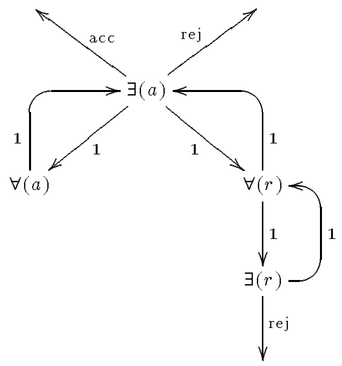


Figure 6-4: Labeled Transition System of N_F

to that of A on input x, ignoring the labels of the control transitions, and ignoring the **acc**-labeled and **rej**-labeled transitions altogether.

Let T be the deterministic Turing machine which, started with a string of 0's on its tape, successively adds 1 to the binary number on its tape until the original string of 0's is changed into a string of 1's (of the same length). Then T enters an accepting state and halts. So, when started on a string on m 0's, it runs for at least 2^m steps and halts. The polynomial-time translation *net* given above for alternating Turing Machines also holds for any deterministic polynomial-space Turing Machine, except that we add both **acc**-labeled and **rej**-labeled transitions for *every* pair (i, q). Hence if "started" on input consisting of a string of $p'(|x|)$ 0's, this net is of size bounded by some polynomial in $|x|$, and has the sole behaviors that it fires at most some fixed $m' > 2^{p'(|x|)}$ number of **1**'s, and each point along the way it non-deterministically chooses between firing either **acc** or **rej** and exiting, or firing a **1**. Furthermore, after firing m' **1**'s followed by a single **acc** or **rej**, it reaches a deadlocked state. We call this net Count(m'). We can assume without loss of generality that m' is odd, and since m' exceeds the time bound of A on input x, we can assume without loss of generality that every computation path of A on input x is exactly of length m'.

To finish the construction, let N_F be a finite 1-safe net of constant size with the labeled transition system pictured in Figure 6-4, and let $N_x \stackrel{\text{def}}{=} N_F \;||_{\mathbf{1},\mathbf{acc},\mathbf{rej}}$ Count(m'), where synchronization is required on the symbols **1**, **acc**, and **rej**. N_x is of size polynomial in $|x|$, and its labeled transition system is bisimilar to the transition system pictured in Figure 6-5.

We now show that *net*(A_x) is bisimilar to the net N_x iff A accepts input x. For one direction, suppose that *net*(A_x) is bisimilar to N_x; then *net*(A_x) must have some m'-length path bisimilar to $\exists(a)\forall(a)\exists(a)\forall(a)\ldots\exists(a)$ after which it fires an **acc**-labeled transition. Thus, all the states of *net*(A_x) that are reached along the way must be accepting. Since the labeled transition system of *net*(A_x) is essentially isomorphic to the labeled transition system of A on x, A must accept x. Recalling our assumptions on A, the other direction follows by a simple induction on $\approx_i$, where $\approx_i$ is an i-step bisimulation (*cf.* [30]). This is a polynomial-time reduction from the acceptance problem for polynomial-space alternating Turing Machines to bisimulation of finite nets. ■

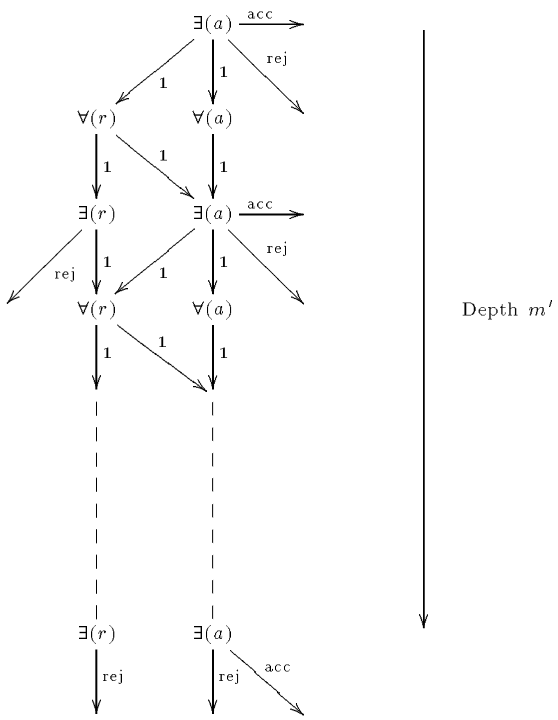


Figure 6-5: Bisimilar to the Labeled Transition System of N_x

It is well-known that the class of problems decidable in polynomial space by alternating Turing Machines is the same as the class of problems decidable in deterministic exponential time by ordinary Turing Machines [11, 27]. We then have as a simple corollary of this fact and Theorem 6.5.7:

Theorem 6.5.8 Bisimulation of finite nets is DEXPTIME-hard.

We now show the lower bounds for the remaining equivalences listed in Table 6.1.

Theorem 6.5.9 For finite nets,

1. trace equivalence is polynomial-time reducible to step-trace equivalence, ST-trace equivalence, interval pomset-trace equivalence, and pomset-trace equivalence,
2. failures equivalence is polynomial-time reducible to step-failures equivalence, ST-failures equivalence, and interval pomset-failures equivalence, and
3. bisimulation is polynomial-time reducible to step-bisimulation, ST-bisimulation, history-preserving bisimulation, maximality-preserving bisimulation, pomset-bisimulation, and pomset-ST-bisimulation.

Proof. We give the proof only for pomset-trace equivalence, as the other cases are completely analogous. For any finite nets N_1, N_2 without hidden transitions, let N_i' be constructed by adding to N_i a single new, initially marked place which is placed in the preset and post-set of every transition of N_i. Clearly, N_i' is trace equivalent to N_i. Since no transitions in N_i' are statically concurrent, it is easy to see that N_1' and N_2' are trace equivalent iff they are pomset-trace equivalent; hence N_1 and N_2 are trace equivalent iff N_1' and N_2' are pomset-trace equivalent. This is a polynomial-time reduction from trace equivalence to pomset-trace equivalence. ∎

We then have as a simple corollary:

Theorem 6.5.10 For finite nets, the decision problems for

1. step-trace equivalence, ST-trace equivalence, interval pomset-trace equivalence, and pomset-trace equivalence are EXPSPACE-hard,
2. step-failures equivalence, ST-failures equivalence, and interval pomset-failures equivalence are EXPSPACE-hard,
3. delay bisimulation, branching bisimulation, step-bisimulation, ST-bisimulation, history-preserving bisimulation, maximality-preserving bisimulation, pomset-bisimulation, and pomset-ST-bisimulation are DEXPTIME-hard.

Proof. Delay bisimulation and branching bisimulation coincide with bisimulation for nets without hidden transitions [43]. The lower bound for delay bisimulation and branching bisimulation is thus a simple consequence of Theorem 6.5.8. All the other lower bounds follow immediately from Theorems 6.5.4, 6.5.6, 6.5.8, and 6.5.9. ∎

We remark that all the lower bound results in this section are independent of the presence of hidden transitions, except as specifically stated in the lower bound proofs for trace equivalence.

6.6 Conclusions

We remark that all these complexity results apply equally to process approximation as well as equivalence. An open problem is the decidability and complexity of augmentation-closed pomset-trace equivalence. Another open problem that we regard as especially significant is the decidability and complexity of our earlier general pomset-failures semantics [23], which keeps track of concurrent divergences. We are currently working to extend our methods to handle these cases.

Chapter 7

Other Results, Open Problems, and Future Work

There is not yet a consensus on what an action refinement operator should be. For example, our action refinement operator and that of [47] are tuned to a CSP-style synchronization-with-restriction, while those of [3, 20] are tuned to a CCS-style synchronization-by-hiding-complementary-actions. In this regard, an action-refinement theory closely related to ours has been proposed by Hennessy [20]. His theory incorporates an interesting, and in certain respects more powerful, action refinement operation, and he has compositionality and full abstraction results similar to ours. Unlike our action refinement operation, Hennessy's definition allows "concurrent" refinement nets to "communicate" with one another in a manner closely related to CCS-style parallel composition, where concurrent, complementary actions (*i.e.*, a and $\bar{a}$) can synchronize and perform a hidden move. However, in order for Hennessy's semantics to remain compositional for this powerful sort of action refinement, this inter-communication must in fact be quite restricted: in particular, "initial" hidden communications between refinement nets must be disallowed. As a result, Hennessy forbids some simple action refinements like $(a \mid b)[a{:=}c,\ b{:=}\bar{c}]$. We have explored the connection between Hennessy's and our theories of action refinement in [25]. In particular, [25] presents a new operator of *self-synchronization*, which allows concurrent transitions within a process to synchronize, and shows that self-synchronization provides a tight connection between our action refinement operator and Hennessy's communicating action refinement operator. Furthermore, self-synchronization can detect "steps" of concurrent actions, and hence non-interleaving semantics are not compositional.

In a related direction, we believe that true concurrency semantics may reveal a distinction between existing synchronization operators for which non-interleaving semantics are compositional. For example, in interleaving theories like CSP and CCS, the choice of operators is immaterial since the different synchronization operators can simulate each other. However, the known simulations do not preserve true concurrency semantics. The relation between the process theories based on these different synchronization mechanisms remains an interesting question, which we are currently exploring.

This thesis has shown that our $[\![\cdot]\!]^{\mathrm{MAY}}$,$[\![\cdot]\!]^{\mathrm{MUST}}_{\text{split-}\gamma}$ and $[\![\cdot]\!]^{\mathrm{TEST}}_{\text{split-}\gamma}$ are compositional for all our operators, including action refinement, and are respectively adequate for MAY-, MUST-, and Testing-equivalence. However, it remains open as to which sorts of observations these semantics are fully abstract. To this end, we are currently working on a theory of *local observers*, which we believe will be able to detect full causality and concurrency through experiments.

In Chapter 4, we showed that all of our semantical spaces form complete partial orders, and that our action refinement and CCS/CSP operators on nets correspond to continuous semantical operations. Consequently, we expect that our theory will routinely support *arbitrary* (not merely guarded) *recursive* definitions of nets, with recursion understood as usual via least fixed points. We hope to formalize these definitions in the near future.

An important direction for further research is development of the algebra of process terms with refinement. One immediate problem to consider is finding a complete axiom system for equations between closed recursion-free CSP/CCS process terms—corresponding to the (non-divergent) isolated elements in our semantical spaces.

Bibliography

[1] L. Aceto and U. Engberg. Failure semantics for a simple process language with refinement. Technical report, INRIA, Sophia-Antipolis, 1991.

[2] L. Aceto and M. Hennessy. Towards action-refinement in process algebras. In *Proceedings, Fourth Annual Symposium on Logic in Computer Science*, pages 138–145. IEEE Computer Society Press, 1989.

[3] L. Aceto and M. Hennessy. Adding action refinement to a finite process algebra. In *Proceedings of ICALP '91, Volume 510 of Lecture Notes in Computer Science*, pages 506–519, 1991.

[4] C. Alvarez, B. J., J. Gabarro, and M. Santa. Parallel complexity in the design and analysis of concurrent systems. In *Proceedings of PARLE '91, Volume 505 of the Lecture Notes in Computer Science*, pages 288–303, 1991.

[5] E. Best, R. Devillers, A. Kiehn, and L. Pomello. Concurrent bisimulations in Petri Nets. *Acta Inf.*, 28:231–264, 1991.

[6] G. Boudol and I. Castellani. On the semantics of concurrency: Partial orders and transition systems. In *Proceedings of TAPSOFT '87, Volume 249 of the Lecture Notes in Computer Science*, pages 123–137, 1987.

[7] S. Brookes. *A Model for Communicating Systems.* PhD thesis, Oxford University, 1984.

[8] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, July 1984.

[9] S. D. Brookes and A. W. Roscoe. An improved failures model for communicating processes. In *Seminar on Concurrency, Volume 197 of Lecture Notes in Computer Science*, pages 281–305, 1984.

[10] L. Castellano, G. De Michelis, and L. Pomello. Concurrency vs. interleaving: an instructive example. *Bulletin of the European Association of Theoretical Computer Science*, 31:12–15, 1987.

[11] A. Chandra and L. Stockmeyer. Alternation. In *Proceedings of the Seventeenth Annual IEEE Symposium on Foundations of Computer Science*, pages 98–108, 1976.

[12] F. Cherief and P. Schnoebelen. τ-bisimulations and full abstraction for the refinement of actions. *Information Processing Letters*, 40:219–222, 1991.

[13] R. Devillers. Maximality preserving bisimulation. *Theor. Comput. Sci.*, 102(1):165–184, Aug. 1992.

[14] P. Fishburn. Intransitive indifference with unequal indifference intervals. *Journal of Mathematical Psychology*, 7:144–149, 1970.

[15] U. Goltz. CCS and Petri nets. Technical report, GMD, July 1990.

[16] R. Gorrieri. *Refinement, Atomicity, and Transactions for Process Description Languages.* PhD thesis, University of Pisa, 1991.

[17] J. Groote and F. Vaandrager. An efficient algorithm for branching bisimulation and stuttering equivalence. In *Proceedings of ICALP '90*, 1990.

[18] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques.* Series on Foundations of Computing. MIT Press, 1992. 419 pp.

[19] M. C. Hennessy. *Algebraic Theory of Processes.* Series on Foundations of Computing. MIT Press, 1988. 272 pp.

[20] M. C. Hennessy. Concurrent testing of processes. In *Proceedings of CONCUR '92, Volume 630 of Lecture Notes in Computer Science*, pages 94–107, 1992.

[21] C. A. R. Hoare. *Communicating Sequential Processes.* Series in Computer Science. Prentice-Hall, Inc., 1985. 256 pp.

[22] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley, 1979.

[23] L. Jategaonkar and A. R. Meyer. Testing equivalence for Petri nets with action refinement. In *Proceedings of CONCUR '92, Volume 630 of the Lecture Notes in Computer Science*, pages 17–31, 1992.

[24] L. Jategaonkar and A. R. Meyer. Deciding true concurrency equivalences on finite safe nets. In *Proceedings of ICALP '93, Volume 700 of the Lecture Notes in Computer Science*, pages 519–531, 1993.

[25] L. Jategaonkar and A. R. Meyer. Self-synchronization of concurrent processes. In *Proceedings of LICS '93*, pages 409–417, 1993.

[26] P. Kannelakis and S. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Inf. Comput.*, 86(1):43–68, 1990.

[27] D. Kozen. On parallelism in turing machines. In *Proceedings of the Seventeenth Annual IEEE Symposium on Foundations of Computer Science*, pages 89–97, 1976.

[28] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, 1991. 427 pp.

[29] A. J. Mayer and L. J. Stockmeyer. The complexity of word problems – this time with interleaving. Technical report, IBM Research Division, Almaden Research Center, San Jose, CA, Sept. 1992.

[30] R. Milner. *Communication and Concurrency.* Series in Computer Science. Prentice-Hall, Inc., 1989.

[31] R. Milner. Action structures. Technical report, University of Edinburgh, UK, 1992.

[32] M. Nielsen, U. Engberg, and K. S. Larsen. Fully abstract models for a process language with refinement. In *Linear Time, Branching Time, and Partial Order in Logics and Models of Concurrency, Volume 354 of Lecture Notes in Computer Science*, pages 523–548, 1988.

[33] L. Pomello. Some equivalence notions for concurrent systems: An overview. In *Advances in Petri Nets, Volume 222 of Lecture Notes in Computer Science*, pages 381–400, 1985.

[34] A. Rabinovich. Checking equivalences between concurrent systems of finite agents. In *Proceedings of ICALP '92, Volume 379 of the Lecture Series in Computer Science*, pages 696–707, 1992.

[35] A. Rabinovich and B. Trakhtenbrot. Behavior structures and nets of processes. *Fundamenta Informaticae*, 11(4):357–404, 1988.

[36] L. Stockmeyer, Jan. 1992. Unpublished notes.

[37] D. Taubner and W. Vogler. Step failures semantics and a complete proof system. *Acta Inf.*, 27(2):125–156, Nov. 1989.

[38] F. Vaandrager, 1991. Private communication.

[39] R. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions. In *Proceedings of MFCS '89, Volume 379 of the Lecture Series in Computer Science*, pages 237–248, 1989.

[40] R. van Glabbeek and U. Goltz. Partial order semantics for refinement of actions – neither necessary nor sufficient but appropriate when used with care. *Bulletin of the European Association of Theoretical Computer Science*, 38:154–163, 1989.

[41] R. van Glabbeek and U. Goltz. Refinement of actions in causality based models. In *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness, Proceedings of REX Workshop '89, Volume 430 of Lecture Notes in Computer Science*, pages 267–300, 1989.

[42] R. van Glabbeek and F. Vaandrager. Petri net models for algebraic theories of concurrency. In *Proceedings of PARLE '87, Volume 259 of the Lecture Notes in Computer Science*, pages 224–242, 1987.

[43] R. van Glabbeek and P. Weijland. Branching time and abstraction in bisimulation semantics. *Information Processing Letters*, 89:613–618, 1989.

[44] W. Vogler. Bisimulation and action refinement. In *Proceedings of STACS '91, Volume 480 of the Lecture Notes in Computer Science*, pages 309–321, 1991.

[45] W. Vogler. Bisimulation and action refinement. Technical report, Technische Universität München, 1991.

[46] W. Vogler. Deciding history preserving bisimulation. In *Proceedings of ICALP '91, Volume 510 of the Lecture Notes in Computer Science*, pages 495–505, 1991.

[47] W. Vogler. Failures semantics based on interval semiwords is a congruence for refinement. *Distributed Computing*, 4:139–162, 1991.

[48] W. Vogler. Generalized om-bisimulation. Technical report, Technische Universität München, 1991.

[49] W. Vogler. Is partial order semantics necessary for action refinement? Technical report, Technische Universität München, 1991.

[50] W. Vogler. Bisimulation and action refinement. *Theor. Comput. Sci.*, 114:173–200, 1993.