# HOL-CSP_Proc-Omata: A Bridge between CSP Processes and Functional Automata

Benoît Ballenghien Burkhart Wolff

February 4, 2026

# Abstract

This entry develops the Proc-Omata framework on top of `HOL-CSP` and its extensions. Proc-Omata are defined from functional automata and come in four variants: deterministic, terminating deterministic, non-deterministic, and terminating non-deterministic. This subclass of processes combines the expressiveness of CSP with automata-like structure (reachability, enableness), making it particularly amenable to invariant-based reasoning.

We lift sequential composition and synchronization product to the automata level through combination functions and prove compactification theorems that enable reasoning over large process architectures. An essential ingredient is the use of restriction spaces, which guarantees well-defined fixed points even in the non-deterministic setting. Finally, we illustrate the applicability of the framework with the Dining Philosophers, where compactification yields proofs that scale to an arbitrary finite but unbounded number of participants in this parameterized process architecture.

# Contents


**1 Introduction** **9**

**2 An Excursion into Determinism** **11**
2.1 Accepts initials . . . 11
2.1.1 Definition . . . 11
2.1.2 First properties . . . 12
2.1.3 Monotonicity . . . 12
2.1.4 Behaviour on Operators . . . 12
2.1.5 Characterizations with After . . . 14
2.2 Deterministic process . . . 16
2.2.1 Definition . . . 16
2.2.2 Monotonicity . . . 16
2.2.3 Characterization as Maximal . . . 16
2.2.4 Characterization with After . . . 19
2.2.5 Operators preserving Determinism . . . 19
2.2.6 Operators not (always) preserving Determinism . . . 20
2.3 Application to Operational Semantics . . . 20

**3 ProcOmata: Functional Automata embedded into CSP Processes** **23**
3.1 Definitions . . . 24
3.1.1 Non-deterministic and deterministic Automata . . . 24
3.1.2 Enableness . . . 24
3.1.3 States allowing Termination . . . 25
3.1.4 Reachability . . . 25
3.1.5 Morphisms . . . 25
3.1.6 Generic update Functions . . . 27
3.1.7 Assumptions on Automata . . . 28
3.2 First Properties . . . 31
3.2.1 $\varepsilon$, $\varrho$ and $\omega$ first equalities . . . 31
3.2.2 Properties of our morphisms . . . 32
3.2.3 Reachability results (for $\mathcal{R}_d$ and $\mathcal{R}_{nd}$) . . . 36
3.3 Normalization . . . 38

3.3.1 Non-deterministic Case . . . . . 38
3.3.2 Deterministic Case . . . . . 41
3.3.3 Link between deterministic and non-deterministic ProcOmata . . . . . 43
3.3.4 Prove Equality between ProcOmata . . . . . 43

**4 Advanced Properties of ProcOmata 47**

**5 Combining Automata for Synchronization Product 49**
5.1 Definitions . . . . . 49
5.1.1 General Patterns . . . . . 49
5.1.2 Specializations . . . . . 50
5.2 First Properties . . . . . 52
5.3 Reachability . . . . . 53
5.4 Normalization . . . . . 54

**6 Compactification of Synchronization Product 57**
6.1 Iterated Combine . . . . . 57
6.1.1 Definitions . . . . . 57
6.1.2 First Results . . . . . 58
6.1.3 Reachability . . . . . 59
6.1.4 Transmission of Properties . . . . . 59
6.1.5 Normalization . . . . . 60
6.2 Compactification Theorems . . . . . 60
6.2.1 Binary . . . . . 60
6.2.2 Rlist . . . . . 62
6.2.3 ListslenL . . . . . 62
6.2.4 Multiple . . . . . 63
6.3 Derived Versions . . . . . 64
6.4 More on Iterated Combine . . . . . 65
6.5 More on Events . . . . . 66

**7 Combining Automata for Generalized Synchronization Product 69**
7.1 Definitions . . . . . 69
7.1.1 Specializations . . . . . 69
7.2 First Properties . . . . . 70
7.3 Transitions are unchanged in the Generalization . . . . . 72
7.4 Reachability . . . . . 72
7.5 Normalization . . . . . 73

**8 Compactification of Synchronization Product Generalized 77**
8.1 Iterated Combine . . . . . 77
8.1.1 Definitions . . . . . 77

8.1.2 First Results . . . 77
8.1.3 Transmission of Properties . . . 78
8.1.4 Normalization . . . 79
8.2 Compactification Theorems . . . 80
8.2.1 Binary . . . 80
8.2.2 Rlist . . . 81
8.2.3 ListslenL . . . 81
8.2.4 Multiple . . . 82
8.3 Derived Versions . . . 83
8.4 More on Iterated Combine and Events . . . 83

**9 Combining Automata for Sequential Composition Generalized 85**
9.1 Definitions . . . 85
9.2 General Patterns . . . 85
9.3 Specializations . . . 86
9.4 First Properties . . . 87
9.4.1 Reachability . . . 88
9.5 Normalization . . . 89

**10 Compactification of Sequential Composition Generalized 91**
10.1 Iterated Combine . . . 91
10.1.1 Definitions . . . 91
10.1.2 First Results . . . 91
10.1.3 Reachability . . . 92
10.1.4 Transmission of Properties . . . 92
10.1.5 Normalization . . . 93
10.2 Compactification Theorems . . . 93
10.2.1 Binary . . . 93
10.2.2 ListslenL . . . 94
10.2.3 Multiple . . . 95

**11 Application : May Philosophers dine ? 97**
11.1 Preliminaries . . . 97
11.1.1 Preliminary lemmas for proof automation . . . 97
11.2 The dining processes definition . . . 97
11.2.1 Unfolding rules . . . 98
11.3 Translation into normal form . . . 98
11.3.1 *FORK*, *LPHIL0* and *RPHIL* are normalizable . . . 98
11.3.2 *FORKS* is normalizable . . . 100
11.3.3 *PHILS* is normalizable . . . 100
11.3.4 The complete process *DINING* is normalizable . . . 101
11.4 And finally: Philosophers may dine ! Always ! . . . 101
11.4.1 Construction of an invariant for the dining automaton 101

11.4.2 The invariant *inv-dining* implies that *DINING* is *deadlock-free* . . . 102
11.4.3 Conclusion . . . 102
11.5 Alternative version with only right-handed philosophers (in order to show that it's not *deadlock-free*) . . . 102
11.5.1 Setup . . . 102
11.5.2 Normalization . . . 103
11.5.3 Correspondance between our normalized processes and the previous definitions . . . 103
11.5.4 Proof that we have a deadlock in the state (*replicate N 1*, *replicate N 1*) . . . 103
11.5.5 Proof that this state is reachable from our initial state, i.e. (*replicate N 1*, *replicate N 1*) $\in \mathcal{R}_d$ $A_{RD}$ (*replicate N 0*, *replicate N 0*) . . . 103

**12 Other Results similar to Compactification** **105**
12.1 Some preliminary Results . . . 105
12.2 Results for *Det* . . . 106
12.3 Results for *Ndet* . . . 106
12.4 Other Operators . . . 106
12.4.1 *initials* . . . 106
12.4.2 *Throw* . . . 107
12.4.3 ($\triangle$) . . . 107
12.4.4 After . . . 108
12.5 OpSem . . . 109

**13 Conclusion** **111**
13.1 Entry Point . . . 111
13.2 Conclusion . . . 111

# Chapter 1

# Introduction

Communicating Sequential Processes (CSP) offers a rich and expressive framework for modeling and reasoning about concurrent systems. Its denotational, operational, and algebraic facets are covered by the sessions `HOL-CSP` [3], `HOL-CSPM` [2], `HOL-CSP_OpSem` [4], `HOL-CSP_RS` [6], and `HOL-CSP_PTick`. These developments, initially following Roscoes presentation [7], have since evolved considerably to admit arbitrary types, infinite sets, parameterized termination, and more.

However, this expressiveness comes with a cost: proofs about complex or parametric process architectures often become intricate and hard to scale. Proc-Omata address this issue by slightly constraining the class of processes in order to benefit from more powerful proof techniques. First sketched in [8] and properly conceptualized in [5], the Proc-Omata framework consists in embedding functional automata into CSP. The resulting subclass of processes combines the expressive and compositional features of CSP with automata-like properties (reachability, enableness, absence of divergences), making it particularly amenable to invariant reasoning.

In this entry we start by formalizing the basic notions of functional automata such as reachability and enableness, before introducing the definitions of Proc-Omata themselves. For synchronization product and sequential composition, we then provide combination functions that realize the effect of CSP operators at the level of the underlying automata. These translations are formally proved correct, and culminate in compactification theorems, which generalize the constructions inductively to architectural operators.

# Chapter 2

# An Excursion into Determinism

This chapter is a preliminary work. Indeed, later in the construction, we will define the notion of Procomata which comes in different flavours, in particular deterministic ones. We will establish then that such ProcOmata produce deterministic processes, a classical notion in CSP that we formalize below.

In a word, a deterministic process cannot refuse an event in which it can engage. More formally, if $s$ @ $[e] \in \mathcal{T}\ P$, then $(s, \{e\}) \notin \mathcal{F}\ P$. In this theory, we follow the proof sketch given in [7] for characterizing deterministic processes as maximal elements for the failure-divergence refinement ($\sqsubseteq_{FD}$). Other lemmas are proved with respect to CSP operators.

## 2.1 Accepts initials

This notion is a weak version of determinism. It captures the idea of being deterministic for one step.

### 2.1.1 Definition

**unbundle** *option-type-syntax*

**definition** *accepts-initials* :: ‹$('a, 'r)\ process_{ptick} \Rightarrow bool$› (‹$determ^0$›)
**where** ‹$determ^0\ P \equiv \forall\, e \in P^0.\ \{e\} \notin \mathcal{R}\ P$›

**lemma** *accepts-initialsI* : ‹$(\bigwedge e.\ e \in P^0 \Longrightarrow \{e\} \notin \mathcal{R}\ P) \Longrightarrow determ^0\ P$›
**and** *accepts-initialsD* : ‹$determ^0\ P \Longrightarrow e \in P^0 \Longrightarrow \{e\} \notin \mathcal{R}\ P$›
⟨*proof*⟩

**lemma** *accepts-initials-def-bis*:
‹$determ^0\ P \longleftrightarrow (\forall\, e \in P^0.\ \forall\, X \in \mathcal{R}\ P.\ e \notin X)$›
⟨*proof*⟩

**lemma** *accepts-initialsI-bis* : ‹$(\bigwedge e\ X.\ e \in P^0 \Longrightarrow X \in \mathcal{R}\ P \Longrightarrow e \notin X) \Longrightarrow determ^0\ P$›
**and** *accepts-initialsD-bis* : ‹$determ^0\ P \Longrightarrow e \in P^0 \Longrightarrow X \in \mathcal{R}\ P \Longrightarrow e \notin X$›
⟨*proof*⟩

### 2.1.2 First properties

**lemma** *accepts-initials-STOP* [*simp*] : ‹$determ^0\ STOP$›
⟨*proof*⟩

**lemma** *accepts-initials-SKIP* [*simp*] : ‹$determ^0\ (SKIP\ r)$›
⟨*proof*⟩

**lemma** *not-accepts-initials-BOT* [*simp*] : ‹$\neg\ determ^0\ \bot$›
⟨*proof*⟩

**lemma** *accepts-initials-imp-initial-tick-iff-is-SKIP*:
‹$determ^0\ P \Longrightarrow \checkmark(r) \in P^0 \longleftrightarrow P = SKIP\ r$›
⟨*proof*⟩

**lemma** *accepts-initials-imp-not-initial-tick-iff-is-STOP-or-some-initial-ev*:
‹$determ^0\ P \Longrightarrow (range\ tick \cap P^0 = \{\}) \longleftrightarrow P = STOP \vee (\exists\, e.\ ev\ e \in P^0)$›
⟨*proof*⟩

### 2.1.3 Monotonicity

**lemma** *mono-accepts-initials-F*: ‹$P \sqsubseteq_F Q \Longrightarrow determ^0\ P \Longrightarrow determ^0\ Q$›
⟨*proof*⟩

**lemma** *mono-accepts-initials-FD*: ‹$P \sqsubseteq_{FD} Q \Longrightarrow determ^0\ P \Longrightarrow determ^0\ Q$›
⟨*proof*⟩

**lemma** *mono-accepts-initials*: ‹$P \sqsubseteq Q \Longrightarrow determ^0\ P \Longrightarrow determ^0\ Q$›
⟨*proof*⟩

**lemma** *restriction-adm-accepts-initials* [*restriction-adm-process*$_{ptick}$*-simpset*, *simp*]
:
‹$adm_{\downarrow}\ (\lambda x.\ determ^0\ (f\ x))$› **if** ‹$cont_{\downarrow}\ f$›
**for** $f$ :: ‹$'b$ :: $restriction \Rightarrow ('a,\ 'r)\ process_{ptick}$›
⟨*proof*⟩

### 2.1.4 Behaviour on Operators

**lemma** *accepts-initials-Mprefix* [*simp*] : ‹$determ^0\ (\Box a \in A \rightarrow P\ a)$›

⟨*proof*⟩

**lemma** *accepts-initials-write0* [*simp*] : ‹$determ^0\ (a \to P)$›
⟨*proof*⟩

**lemma** *accepts-initials-write* [*simp*] : ‹$determ^0\ (c!a \to P)$›
⟨*proof*⟩

**lemma** *accepts-initials-read* [*simp*] : ‹$determ^0\ (c?a{\in}A \to P\ a)$›
⟨*proof*⟩

**lemma** *accepts-initials-Ndet-iff*:
‹$determ^0\ (P \sqcap Q) \longleftrightarrow determ^0\ P \wedge determ^0\ Q \wedge P^0 = Q^0$›
⟨*proof*⟩

**lemma** *accepts-initials-GlobalNdet-iff*:
‹$determ^0\ (\sqcap a \in A.\ P\ a) \longleftrightarrow$
$(\forall\, a \in A.\ determ^0\ (P\ a) \wedge (\forall\, b \in A.\ (P\ a)^0 = (P\ b)^0))$›
⟨*proof*⟩

**lemma** *accepts-initials-Mndetprefix-iff*:
‹$determ^0\ (\sqcap a \in A \to P\ a) \longleftrightarrow (\exists\, a.\ A \subseteq \{a\})$›
⟨*proof*⟩

**lemma** *accepts-initials-ndet-write-iff*:
‹$determ^0\ (c!!a \in A \to P\ a) \longleftrightarrow (\exists\, b.\ c\ `\ A \subseteq \{b\})$›
⟨*proof*⟩

**lemma** *accepts-initials-SKIPS-iff* :
‹$determ^0\ (SKIPS\ R) \longleftrightarrow R = \{\} \vee (\exists\, r.\ R = \{r\})$›
⟨*proof*⟩

**lemma** *accepts-initials-Det* :
‹$determ^0\ (P \sqcup Q) \longleftrightarrow P = STOP \vee Q = STOP \vee range\ tick \cap P^0 \cap Q^0 \neq \{\}$
$\vee$
$range\ tick \cap (P^0 \cup Q^0) = \{\}$›
(**is** ‹- $\longleftrightarrow$ *?rhs*›) **if** *accepts-initials* : ‹$determ^0\ P$› ‹$determ^0\ Q$›
⟨*proof*⟩

**lemma** *accepts-initials-GlobalDet* :
‹$determ^0\ (\sqcup a \in A.\ P\ a)$› **if** ‹$\bigwedge a.\ a \in A \Longrightarrow determ^0\ (P\ a)$›
‹$range\ tick \cap (\bigcap a \in A.\ (P\ a)^0) \neq \{\} \vee range\ tick \cap (\bigcup a \in A.\ (P\ a)^0) = \{\}$›
⟨*proof*⟩

**lemma** *accepts-initials-Seq*$_{ptick}$ :
‹$determ^0$ $(P \;;_{\checkmark}\; Q) \longleftrightarrow (\forall r.\ \checkmark(r) \in P^0 \longrightarrow determ^0\ (Q\ r))$› **if** ‹$determ^0$ $P$›
⟨*proof*⟩

**corollary** *accepts-initials-Seq* :
‹$determ^0$ $(P \;;\; Q) \longleftrightarrow (P^0 \cap range\ tick = \{\} \vee determ^0\ Q)$› **if** ‹$determ^0$ $P$›
⟨*proof*⟩

**lemma** (**in** *Sync*$_{ptick}$*-locale*) *accepts-initials-Sync*$_{ptick}$ :
‹$determ^0$ $(P\ [\![S]\!]_{\checkmark}\ Q)$› **if** ‹$determ^0$ $P$› ‹$determ^0$ $Q$›
⟨*proof*⟩

**corollary** *accepts-initials-Sync*:
‹$determ^0\ P \Longrightarrow determ^0\ Q \Longrightarrow determ^0\ (P\ [\![S]\!]\ Q)$›
⟨*proof*⟩

**lemma** *accepts-initials-Renaming* : ‹$determ^0$ $(Renaming\ P\ f\ g)$› **if** ‹$determ^0$ $P$›
⟨*proof*⟩

**lemma** *accepts-initials-Throw-iff* : ‹$determ^0$ $(P\ \Theta\ a \in A.\ Q\ a) \longleftrightarrow determ^0\ P$›
⟨*proof*⟩

**lemma** *accepts-initials-Sliding*:
‹$determ^0\ P \Longrightarrow determ^0\ Q \Longrightarrow determ^0\ (P \rhd Q) \longleftrightarrow$
$P = STOP \vee P^0 \subseteq Q^0 \wedge (range\ tick \cap P^0 \neq \{\} \vee range\ tick \cap Q^0 = \{\})$›
⟨*proof*⟩

### 2.1.5 Characterizations with After

**context** *After*
**begin**

Interesting results about the fact that we can express a process with *Mprefix* and (*after*)

**lemma** *leFD-SKIPS-Det-Mprefix-After*:
‹$P \sqsubseteq_{FD} SKIPS\ \{r.\ \checkmark(r) \in P^0\}\ \Box\ (\Box a \in \{a.\ ev\ a \in P^0\} \rightarrow P\ after\ a)$› (**is** ‹$P \sqsubseteq_{FD}$ *?rhs*›)
⟨*proof*⟩

**lemma** *accepts-initials-imp-eq-Mprefix-After*:
‹$P = (\ \ if\ \exists r.\ \checkmark(r) \in P^0\ then\ SKIP\ (THE\ r.\ \checkmark(r) \in P^0)$
$else\ \Box a \in \{e.\ ev\ e \in P^0\} \rightarrow P\ after\ a)$› (**is** ‹$P =$ *?rhs*›)

**if** ‹$determ^0$ $P$›
⟨*proof*⟩

**theorem** *is-some-Mprefix-iff*:
‹$(\exists\, A\ Q.\ P = \Box a \in A \rightarrow Q\ a) \longleftrightarrow$ *range tick* $\cap\ P^0 = \{\} \wedge$ *accepts-initials* $P$›
**for** $P$ :: ‹$('a,\ 'r)$ $process_{ptick}$›
⟨*proof*⟩

**lemma** *tick-not-initial-imp-STOP-Ndet-Mndetprefix-After-FD*:
‹*range tick* $\cap\ P^0 = \{\} \Longrightarrow STOP \sqcap (\Box a \in \{e.\ ev\ e \in P^0\} \rightarrow P$ *after* $a) \sqsubseteq_{FD} P$›
⟨*proof*⟩

**lemma** ‹*lifelock-free* $P \longleftrightarrow \mathcal{D}\ P = \{\} \wedge (\forall\, t \in \mathcal{T}\ P.\ tF\ t)$›
⟨*proof*⟩

**lemma** *STOP-Ndet-SKIPS-Ndet-Mprefix-After-leF* :
‹$STOP \sqcap SKIPS\ \{r.\ \checkmark(r) \in P^0\} \sqcap (\Box a \in \{e.\ ev\ e \in P^0\} \rightarrow P$ *after* $a) \sqsubseteq_F P$›
(**is** ‹- $\sqcap$ *?lhs1* $\sqcap$ *?lhs2* $\sqsubseteq_F P$›)
⟨*proof*⟩

**lemma** *non-BOT-imp-Mprefix-After-leD* :
‹$\Box a \in \{e.\ ev\ e \in P^0\} \rightarrow P$ *after* $a \sqsubseteq_D P$› (**is** ‹-*?lhs* $\sqsubseteq_D P$›) **if** ‹$P \neq \bot$›
⟨*proof*⟩

**lemma** *non-BOT-imp-STOP-Ndet-SKIPS-Ndet-Mprefix-After-leFD* :
‹$P \neq \bot \Longrightarrow STOP \sqcap SKIPS\ \{r.\ \checkmark(r) \in P^0\} \sqcap (\Box a \in \{e.\ ev\ e \in P^0\} \rightarrow P$ *after* $a) \sqsubseteq_{FD} P$›
⟨*proof*⟩

**theorem** *singl-initial-imp-equals-prefix-After*:
‹$P =$ (*if* $UNIV \notin \mathcal{R}\ P$ *then* $a \rightarrow P$ *after* $a$ *else* $STOP \sqcap (a \rightarrow P$ *after* $a)$)›
**if** *initials-is* : ‹*initials* $P = \{ev\ a\}$›
⟨*proof*⟩

**lemma** ‹$\{ev\ e\} \notin \mathcal{R}\ P \Longrightarrow ev\ e \in P^0$›
⟨*proof*⟩

**end**

## 2.2 Deterministic process

### 2.2.1 Definition

**definition** *deterministic* :: ‹$('a, 'r)$ $process_{ptick}$ $\Rightarrow$ *bool*› (‹*determ*›)
**where** ‹$determ\ P \equiv \forall\ s\ e.\ s\ @\ [e] \in \mathcal{T}\ P \longrightarrow (s, \{e\}) \notin \mathcal{F}\ (P)$›

**lemma** *deterministicI* : ‹$(\bigwedge t\ e.\ t\ @\ [e] \in \mathcal{T}\ P \Longrightarrow (t, \{e\}) \notin \mathcal{F}\ (P)) \Longrightarrow determ\ P$›
**and** *deterministicD* : ‹$determ\ P \Longrightarrow t\ @\ [e] \in \mathcal{T}\ P \Longrightarrow (t, \{e\}) \notin \mathcal{F}\ (P)$›
⟨*proof*⟩

**lemma** *deterministic-STOP* [*simp*] : ‹$determ\ STOP$›
**and** *deterministic-SKIP* [*simp*] : ‹$determ\ (SKIP\ r)$›
⟨*proof*⟩

**lemma** *deterministic-div-free* : ‹$determ\ P \Longrightarrow \mathcal{D}\ P = \{\}$›
⟨*proof*⟩

**lemma** *not-deterministic-BOT* [*simp*] : ‹$\neg\ determ\ \bot$›
⟨*proof*⟩

### 2.2.2 Monotonicity

**lemma** *mono-deterministic-F*: ‹$P \sqsubseteq_F Q \Longrightarrow determ\ P \Longrightarrow determ\ Q$›
⟨*proof*⟩

**lemma** *mono-deterministic-FD*: ‹$P \sqsubseteq_{FD} Q \Longrightarrow determ\ P \Longrightarrow determ\ Q$›
⟨*proof*⟩

**lemma** *mono-deterministic*: ‹$P \sqsubseteq Q \Longrightarrow determ\ P \Longrightarrow determ\ Q$›
⟨*proof*⟩

**lemma** *restriction-adm-deterministic* [$restriction\text{-}adm\text{-}process_{ptick}$-*simpset*, *simp*]
:
‹$adm_{\downarrow}\ (\lambda x.\ determ\ (f\ x))$› **if** ‹$cont_{\downarrow}\ f$›
**for** $f$ :: ‹$'b$ :: *restriction* $\Rightarrow$ $('a, 'r)$ $process_{ptick}$›
⟨*proof*⟩

### 2.2.3 Characterization as Maximal

#### Some preliminary work

**definition** $is\text{-}process_T$ :: ‹$('a, 'r)$ $trace_{ptick}$ *set* $\Rightarrow$ *bool*›
**where** ‹$is\text{-}process_T\ T \equiv$
$[] \in T \wedge (\forall t \in T.\ ftF\ t) \wedge (\forall t\ u.\ t\ @\ u \in T \longrightarrow t \in T) \wedge$
$(\forall t\ r\ e.\ t\ @\ [\checkmark(r)] \in T \longrightarrow e \neq \checkmark(r) \longrightarrow t\ @\ [e] \notin T)$›

**typedef** $('a, 'r)$ $process_T$ = ‹$\{T :: ('a, 'r)\ trace_{ptick}\ set\ .\ is\text{-}process_T\ T\}$›

⟨*proof*⟩

**setup-lifting** *type-definition-process*$_T$

**lift-definition** *Traces*$_T$ ::
‹$('a,\ 'r)$ *process*$_T \Rightarrow ('a,\ 'r)$ *trace*$_{ptick}$ *set*› (‹$\mathcal{T}_T$›)
**is** ‹$\lambda P.$ *Rep-process*$_T$ $P$› ⟨*proof*⟩

**lemma** *Process*$_T$*-eq-spec* : ‹$T = U \longleftrightarrow \mathcal{T}_T\ T = \mathcal{T}_T\ U$›
⟨*proof*⟩

**lemma** *is-process*$_T$*-1* : ‹$[] \in \mathcal{T}_T\ P$›
**and** *is-process*$_T$*-2* : ‹$s \in \mathcal{T}_T\ P \Longrightarrow$ *ftF* $s$›
**and** *is-process*$_T$*-3* : ‹$s\ @\ t \in \mathcal{T}_T\ P \Longrightarrow s \in \mathcal{T}_T\ P$›
**and** *is-process*$_T$*-4* : ‹$s\ @\ [\checkmark(r)] \in \mathcal{T}_T\ P \Longrightarrow e \neq \checkmark(r) \Longrightarrow s\ @\ [e] \notin \mathcal{T}_T\ P$›
⟨*proof*⟩

**lemmas** *is-process*$_T$*-def-bis* = *is-process*$_T$*-def*[*of* ‹*Rep-process*$_T$ -›, *folded Traces*$_T$*.rep-eq*]

**lift-definition** *process*$_{ptick}$*-of-process*$_T$ ::
‹$('a,\ 'r)$ *process*$_T \Rightarrow ('a,\ 'r)$ *process*$_{ptick}$›
**is** ‹$\lambda T.\ (\{(s,\ X).\ s \in \mathcal{T}_T\ T \wedge X \subseteq -\ \{e.\ s\ @\ [e] \in \mathcal{T}_T\ T\}\},\ \{\})$›
⟨*proof*⟩

**lemma** *F-process*$_{ptick}$*-of-process*$_T$ :
‹$\mathcal{F}$ (*process*$_{ptick}$*-of-process*$_T$ $T$) $= \{(s,\ X).\ s \in \mathcal{T}_T\ T \wedge X \subseteq -\ \{e.\ s\ @\ [e] \in \mathcal{T}_T\ T\}\}$›
**and** *D-process*$_{ptick}$*-of-process*$_T$ :
‹$\mathcal{D}$ (*process*$_{ptick}$*-of-process*$_T$ $T$) $= \{\}$›
**and** *T-process*$_{ptick}$*-of-process*$_T$ :
‹$\mathcal{T}$ (*process*$_{ptick}$*-of-process*$_T$ $T$) $= \mathcal{T}_T\ T$›
⟨*proof*⟩

**lemmas** *process*$_{ptick}$*-of-process*$_T$*-projs* = *F-process*$_{ptick}$*-of-process*$_T$
*D-process*$_{ptick}$*-of-process*$_T$ *T-process*$_{ptick}$*-of-process*$_T$

## Now the big results

**lemma** *bij-betw-det* :
‹*bij-betw process*$_{ptick}$*-of-process*$_T$ *UNIV* $\{P :: ('a,\ 'r)$ *process*$_{ptick}$. *determ* $P\}$›
(**is** ‹*bij-betw process*$_{ptick}$*-of-process*$_T$ *?S1 ?S2*›)
⟨*proof*⟩

**lemma** *SKIPS-is-GlobalDet-SKIP* : ‹$SKIPS\ R = \Box r \in R.\ SKIP\ r$›
⟨*proof*⟩

**lemma** *SKIP-Ndet-SKIP-is-SKIP-Det-SKIP* : ‹$SKIP\ r \sqcap SKIP\ s = SKIP\ r \Box SKIP\ s$›
⟨*proof*⟩

**theorem** *P-FD-some-det* :
— In the generalization, since several terminations may occur after the same trace in the initial process, we have to specify a choice.
**fixes** *termination-choice* :: ‹$('a,\ 'r)\ trace_{ptick} \Rightarrow 'r$›
**assumes** ‹$\bigwedge t.\ \exists r.\ t\ @\ [\checkmark(r)] \in \mathcal{T}\ P \Longrightarrow termination\text{-}choice\ t \in \{r.\ t\ @\ [\checkmark(r)] \in \mathcal{T}\ P\}$›
**defines** ‹$T \equiv \{t \in \mathcal{T}\ P.\ \forall t' < t.\ (\exists r.\ t'\ @\ [\checkmark(r)] \in \mathcal{T}\ P) \longrightarrow t = t'\ @\ [\checkmark(termination\text{-}choice\ t')]\}$›
**shows** ‹$P \sqsubseteq_{FD} process_{ptick}\text{-}of\text{-}process_T\ (Abs\text{-}process_T\ T)$›
⟨*proof*⟩

**theorem** *deterministic-iff-maximal-for-leFD*:
‹$determ\ P \longleftrightarrow (\forall Q.\ P \sqsubseteq_{FD} Q \longrightarrow P = Q)$› **for** $P$ :: ‹$('a,\ 'r)\ process_{ptick}$›
— see TPC, chapter 9)
⟨*proof*⟩

**lemma** ‹$determ\ P \Longrightarrow X \in \mathcal{R}\ P \Longrightarrow X \subseteq -\ P^0$›
⟨*proof*⟩

We have the immediate powerful corollaries.

**corollary** (**in** *After*) *deterministic-process-eq-SKIPS-Det-Mprefix-After* :
‹$determ\ P \Longrightarrow P = SKIPS\ \{r.\ \checkmark(r) \in P^0\} \Box (\Box a \in \{a.\ ev\ a \in P^0\} \rightarrow P\ after\ a)$›
⟨*proof*⟩

**lemma** *deterministic-imp-initial-tick-iff-eq-SKIP* [*simp*] :
‹$determ\ P \Longrightarrow \checkmark(r) \in P^0 \longleftrightarrow P = SKIP\ r$›
⟨*proof*⟩

**lemma** *deterministic-imp-constraints-on-initials* :
‹$determ\ P \Longrightarrow P^0 = \{\} \vee \{a.\ ev\ a \in P^0\} = \{\} \wedge (\exists r.\ P^0 = \{\checkmark(r)\}) \vee$
$\{a.\ ev\ a \in P^0\} \neq \{\} \wedge \{r.\ \checkmark(r) \in P^0\} = \{\}$›
⟨*proof*⟩

**corollary** (**in** *After*) *deterministic-process-eq-SKIP-or-Mprefix-After* :
‹*determ* $P \Longrightarrow P = (\mathit{if}\ \exists\, r.\ \checkmark(r) \in P^0\ \mathit{then}\ \mathit{SKIP}\ (\mathit{THE}\ r.\ P^0 = \{\checkmark(r)\})$
$\mathit{else}\ \Box a \in \{a.\ \mathit{ev}\ a \in P^0\} \rightarrow P\ \mathit{after}\ a)$›
⟨*proof*⟩

### 2.2.4 Characterization with After

**lemma** (**in** *AfterExt*) *deterministic-iff-accepts-initials-After*$_{trace}$:
‹*determ* $P \longleftrightarrow (\forall\, t \in \mathcal{T}\ P.\ tF\ t \longrightarrow \mathit{determ}^0\ (P\ \mathit{after}_{\mathcal{T}}\ t))$›
⟨*proof*⟩

### 2.2.5 Operators preserving Determinism

**lemma** *deterministic-Mprefix-iff* :
‹*determ* $(\Box a \in A \rightarrow P\ a) \longleftrightarrow (\forall\, a \in A.\ \mathit{determ}\ (P\ a))$›
⟨*proof*⟩

**corollary** *deterministic-write0-iff* : ‹*determ* $(a \rightarrow P) \longleftrightarrow \mathit{determ}\ P$›
⟨*proof*⟩

**corollary** *deterministic-write-iff* : ‹*determ* $(c!a \rightarrow P) \longleftrightarrow \mathit{determ}\ P$›
⟨*proof*⟩

**corollary** *deterministic-inj-on-read-iff* :
‹*inj-on* $c\ A \Longrightarrow \mathit{determ}\ (c?a \in A \rightarrow P\ a) \longleftrightarrow (\forall\, a \in A.\ \mathit{determ}\ (P\ a))$›
⟨*proof*⟩

**lemma** *deterministic-inj-Renaming* :
‹*determ* (*Renaming* $P\ f\ g$)› **if** ‹*inj* $f$› ‹*inj* $g$› ‹*determ* $P$›
⟨*proof*⟩

**lemma** *deterministic-bij-Renaming-iff* :
‹*determ* (*Renaming* $P\ f\ g) \longleftrightarrow \mathit{determ}\ P$› **if** ‹*bij* $f$› **and** ‹*bij* $g$›
⟨*proof*⟩

**lemma** *deterministic-Throw* : ‹*determ* $(P\ \Theta\ a \in A.\ Q\ a)$›
**if** ‹*determ* $P$› ‹$\bigwedge a.\ a \in A \Longrightarrow a \in \alpha(P) \Longrightarrow \mathit{determ}\ (Q\ a)$›
⟨*proof*⟩

**lemma** *T-snoc-tick-imp-no-continuation-if-deterministic* :
‹$u = [] \wedge e = \checkmark(r)$› **if** ‹*determ* $P$› ‹$t\ @\ u\ @\ [e] \in \mathcal{T}\ P$› ‹$t\ @\ [\checkmark(r)] \in \mathcal{T}\ P$›
⟨*proof*⟩

**lemma** *T-snoc-ev-imp-no-tick-continuation-if-deterministic* :
‹$u \neq [] \wedge$ *is-ev* $(hd\ u) \vee$ *is-ev* $e$› **if** ‹*determ* $P$› ‹$t$ @ $u$ @ $[e] \in \mathcal{T}\ P$› ‹$t$ @ $[ev\ a] \in \mathcal{T}\ P$›
⟨*proof*⟩

**lemma** *deterministic-Seq*$_{ptick}$ : ‹*determ* $(P\ ;_{\checkmark}\ Q)$›
**if** ‹*determ* $P$› ‹$\bigwedge r.\ r \in \checkmark\mathbf{s}(P) \Longrightarrow$ *determ* $(Q\ r)$›
⟨*proof*⟩

**corollary** *deterministic-Seq* : ‹*determ* $P \Longrightarrow$ *determ* $Q \Longrightarrow$ *determ* $(P\ ;\ Q)$›
⟨*proof*⟩

**lemma** (**in** *After*) *initial-imp-deterministic-After*:
‹$ev\ e \in P^0 \Longrightarrow$ *determ* $P \Longrightarrow$ *determ* $(P$ *after* $e)$›
⟨*proof*⟩

**lemma** (**in** *AfterExt*) *initial-imp-deterministic-After*$_{tick}$:
‹$e \in P^0 \Longrightarrow$ (*case* $e$ *of* $\checkmark(r) \Rightarrow$ *determ* $(\Omega\ P\ r)) \Longrightarrow$
*determ* $P \Longrightarrow$ *determ* $(P$ *after*$_{\checkmark}$ $e)$›
⟨*proof*⟩

### 2.2.6 Operators not (always) preserving Determinism

**lemma** *deterministic-imp-accepts-initials* : ‹*determ* $P \Longrightarrow$ *determ*$^0$ $P$›
⟨*proof*⟩

**corollary** *deterministic-SKIPS-iff* : ‹*determ* (*SKIPS* $R$) $\longleftrightarrow R = \{\} \vee (\exists r.\ R = \{r\})$›
⟨*proof*⟩

**lemma** *deterministic-Det*:
‹*determ* $P \Longrightarrow$ *determ* $Q \Longrightarrow$
*range tick* $\cap P^0 \cap Q^0 \neq \{\} \vee P^0 \cap Q^0 = \{\} \wedge$ *range tick* $\cap (P^0 \cup Q^0) = \{\} \Longrightarrow$ *determ* $(P \Box Q)$›
⟨*proof*⟩

## 2.3 Application to Operational Semantics

**lemma** (**in** *OpSemFD*) *tickFree-trace-trans-preserves-deterministic*:

‹$(P :: ('a,\ 'r)\ process_{ptick})\ {}_{FD}\!\rightsquigarrow^{*}\ t\ Q \Longrightarrow tF\ t \Longrightarrow deterministic\ P \Longrightarrow deterministic\ Q$›
⟨*proof*⟩

**lemma** *deterministic-imp-Refusals-iff*: ‹$deterministic\ P \Longrightarrow X \in \mathcal{R}\ P \longleftrightarrow X \cap P^0 = \{\}$›
⟨*proof*⟩

**lemma** (**in** *OpSemFD*) *deterministic-F-trace-trans-reality-check*:
‹$deterministic\ P \Longrightarrow tF\ t \Longrightarrow$
$(t,\ X) \in \mathcal{F}\ (P :: ('a,\ 'r)\ process_{ptick}) \longleftrightarrow (\exists\, Q.\ (P\ {}_{FD}\!\rightsquigarrow^{*} t\ Q) \wedge X \cap Q^0 = \{\})$›
⟨*proof*⟩

**lemma** ‹$\neg\ deterministic\ ((a \rightarrow SKIP\ undefined) \Box SKIP\ undefined)$›
⟨*proof*⟩

# Chapter 3

# ProcOmata: Functional Automata embedded into CSP Processes

We will often have to perform induction on both the list of automata and the list of states, provided that they have the same length.

**lemma** *induct-2-lists012* [*consumes 1*, *case-names Nil single Cons*] :
‹$[\![$*length xs* = *length ys*; *P* [] []; $\bigwedge$*x1 y1*. *P* [*x1*] [*y1*];
$\bigwedge$*x1 x2 xs y1 y2 ys*. *length xs* = *length ys* $\Longrightarrow$ *P xs ys* $\Longrightarrow$
*P* (*x2* # *xs*) (*y2* # *ys*) $\Longrightarrow$ *P* (*x1* # *x2* # *xs*) (*y1* # *y2* # *ys*)$]\!]$
$\Longrightarrow$ *P xs ys*›
⟨*proof*⟩

**lemma** *nat-induct-012* [*case-names 0 1 2 Suc*]:
‹$[\![$*P 0*; *P* (*Suc 0*); *P* (*Suc* (*Suc 0*)); $\bigwedge$*k*. *Suc* (*Suc 0*) $\leq$ *k* $\Longrightarrow$ *P k* $\Longrightarrow$ *P* (*Suc k*)$]\!]$ $\Longrightarrow$ *P n*›
⟨*proof*⟩

The following results will be moved to `Restriction_Spaces` in the future.

**lemma** *restriction-shift-iterated* :
‹*restriction-shift* (*f* ^^ *k*) (*int k* ∗ *m*)›
**if** ‹*restriction-shift f m*› **for** *f* :: ‹*'a* ⇒ *'a* :: *restriction-space*›
⟨*proof*⟩

**lemma** *non-destructive-iterated* :
‹*non-destructive f* $\Longrightarrow$ *non-destructive* (*f* ^^ *k*)›
**for** *f* :: ‹*'a* ⇒ *'a* :: *restriction-space*›
⟨*proof*⟩

**lemma** *constructive-iterated* :
‹*constructive* (*f* ^^ *k*)› **if** ‹*0* < *k*› ‹*constructive f*›
**for** *f* :: ‹*'a* ⇒ *'a* :: *restriction-space*›
⟨*proof*⟩

**lemma** *restriction-fix-unique-iterated* :
‹$[\![0 < k;\ \mathit{constructive}\ f;\ (f\ \hat{}\hat{}\ k)\ x = x]\!] \Longrightarrow (\upsilon\ x.\ f\ x) = x$›
⟨*proof*⟩

**lemma** *restriction-fix-iterated* :
‹$0 < k \Longrightarrow \mathit{constructive}\ f \Longrightarrow (\upsilon\ x.\ (f\ \hat{}\hat{}\ k)\ x) = (\upsilon\ x.\ f\ x)$›
⟨*proof*⟩

**corollary** *restriction-fix-ind-iterated*
[*consumes 1*, *case-names constructive adm base step*]:
‹$P\ (\upsilon\ x.\ f\ x)$› **if** ‹$0 < k$› ‹$\mathit{constructive}\ f$› ‹$\mathit{adm}_{\downarrow}\ P$› ‹$P\ x$› ‹$\bigwedge x.\ P\ x \Longrightarrow P\ ((f\ \hat{}\hat{}\ k)\ x)$›
⟨*proof*⟩

## 3.1 Definitions

### 3.1.1 Non-deterministic and deterministic Automata

**unbundle** *option-type-syntax*

**type-synonym** $('\sigma,\ 'a)\ \mathit{enabl}\ =$ ‹$'\sigma \Rightarrow 'a\ \mathit{set}$›
**type-synonym** $('\sigma,\ 'a,\ '\sigma')\ \mathit{trans} =$ ‹$'\sigma \Rightarrow 'a \Rightarrow '\sigma'$›
**type-synonym** $('\sigma,\ 'a)\ \mathit{trans}_d\ =$ ‹$('\sigma,\ 'a,\ '\sigma\ \mathit{option})\ \mathit{trans}$›
**type-synonym** $('\sigma,\ 'a)\ \mathit{trans}_{nd} =$ ‹$('\sigma,\ 'a,\ '\sigma\ \mathit{set})\ \mathit{trans}$›

**record** $('\sigma,\ 'a,\ '\sigma',\ 'r)\ A =$
$\tau$ :: ‹$('\sigma,\ 'a,\ '\sigma')\ \mathit{trans}$›
$\omega$ :: ‹$'\sigma \Rightarrow 'r$›

**type-synonym** $('\sigma,\ 'a,\ 'r)\ A_d =$ ‹$('\sigma,\ 'a,\ '\sigma\ \mathit{option},\ 'r\ \mathit{option})\ A$›
**type-synonym** $('\sigma,\ 'a,\ 'r,\ '\alpha)\ A_d\text{-}\mathit{scheme} =$ ‹$('\sigma,\ 'a,\ '\sigma\ \mathit{option},\ 'r\ \mathit{option},\ '\alpha)\ A\text{-}\mathit{scheme}$›
**type-synonym** $('\sigma,\ 'a,\ 'r)\ A_{nd} =$ ‹$('\sigma,\ 'a,\ '\sigma\ \mathit{set},\ 'r\ \mathit{set})\ A$›
**type-synonym** $('\sigma,\ 'a,\ 'r,\ '\alpha)\ A_{nd}\text{-}\mathit{scheme} =$ ‹$('\sigma,\ 'a,\ '\sigma\ \mathit{set},\ 'r\ \mathit{set},\ '\alpha)\ A\text{-}\mathit{scheme}$›

### 3.1.2 Enableness

**consts** $\varepsilon$ :: ‹$('\sigma,\ 'a,\ '\sigma',\ 'r',\ '\alpha)\ A\text{-}\mathit{scheme} \Rightarrow ('\sigma,\ 'a)\ \mathit{enabl}$›
**overloading**
$\varepsilon_d \equiv$ ‹$\varepsilon :: ('\sigma,\ 'a,\ '\sigma\ \mathit{option},\ 'r',\ '\alpha)\ A\text{-}\mathit{scheme} \Rightarrow ('\sigma,\ 'a)\ \mathit{enabl}$›
$\varepsilon_{nd} \equiv$ ‹$\varepsilon :: ('\sigma,\ 'a,\ '\sigma\ \mathit{set},\ 'r',\ '\alpha)\ A\text{-}\mathit{scheme} \Rightarrow ('\sigma,\ 'a)\ \mathit{enabl}$›
**begin**
**fun** $\varepsilon_d$ :: ‹$('\sigma,\ 'a,\ '\sigma\ \mathit{option},\ 'r',\ '\alpha)\ A\text{-}\mathit{scheme} \Rightarrow ('\sigma,\ 'a)\ \mathit{enabl}$›
**where** ‹$\varepsilon_d\ A\ \sigma = \{a.\ \tau\ A\ \sigma\ a \neq \Diamond\}$›

**fun** $\varepsilon_{nd}$ :: ‹$('\sigma, 'a, '\sigma\ set, 'r', '\alpha)\ A\text{-}scheme \Rightarrow ('\sigma, 'a)\ enabl$›
  **where** ‹$\varepsilon_{nd}\ A\ \sigma = \{a.\ \tau\ A\ \sigma\ a \neq \{\}\}$›
**end**

**lemmas** $\varepsilon\text{-}simps[simp\ del] = \varepsilon_d.simps\ \varepsilon_{nd}.simps$

### 3.1.3 States allowing Termination

**consts** $\varrho$ :: ‹$('\sigma, 'a, '\sigma', 'r', '\alpha)\ A\text{-}scheme \Rightarrow '\sigma\ set$›
**overloading**
  $\varrho_d \equiv$ ‹$\varrho :: ('\sigma, 'a, '\sigma', 'r\ option, '\alpha)\ A\text{-}scheme \Rightarrow '\sigma\ set$›
  $\varrho_{nd} \equiv$ ‹$\varrho :: ('\sigma, 'a, '\sigma', 'r\ set, '\alpha)\ A\text{-}scheme \Rightarrow '\sigma\ set$›
**begin**
**fun** $\varrho_d$ :: ‹$('\sigma, 'a, '\sigma', 'r\ option, '\alpha)\ A\text{-}scheme \Rightarrow '\sigma\ set$›
  **where** ‹$\varrho_d\ A = \{\sigma.\ \omega\ A\ \sigma \neq \Diamond\}$›
**fun** $\varrho_{nd}$ :: ‹$('\sigma, 'a, '\sigma', 'r\ set, '\alpha)\ A\text{-}scheme \Rightarrow '\sigma\ set$›
  **where** ‹$\varrho_{nd}\ A = \{\sigma.\ \omega\ A\ \sigma \neq \{\}\}$›
**end**

**lemmas** $\varrho\text{-}simps[simp\ del] = \varrho_d.simps\ \varrho_{nd}.simps$

### 3.1.4 Reachability

**inductive-set** $\mathcal{R}_d$ :: ‹$('\sigma, 'a, 'r, '\alpha)\ A_d\text{-}scheme \Rightarrow '\sigma \Rightarrow '\sigma\ set$›
  **for** $A$ :: ‹$('\sigma, 'a, 'r, '\alpha)\ A_d\text{-}scheme$› **and** $\sigma$ :: $'\sigma$
  **where** *init* : ‹$\sigma\ \in \mathcal{R}_d\ A\ \sigma$›
  | *step* : ‹$\sigma' \in \mathcal{R}_d\ A\ \sigma \Longrightarrow \lfloor\sigma''\rfloor = \tau\ A\ \sigma'\ a \Longrightarrow \sigma'' \in \mathcal{R}_d\ A\ \sigma$›

**inductive-set** $\mathcal{R}_{nd}$ :: ‹$('\sigma, 'a, 'r, '\alpha)\ A_{nd}\text{-}scheme \Rightarrow '\sigma \Rightarrow '\sigma\ set$›
  **for** $A$ :: ‹$('\sigma, 'a, 'r, '\alpha)\ A_{nd}\text{-}scheme$› **and** $\sigma$ :: $'\sigma$
  **where** *init* : ‹$\sigma\ \in \mathcal{R}_{nd}\ A\ \sigma$›
  | *step* : ‹$\sigma' \in \mathcal{R}_{nd}\ A\ \sigma \Longrightarrow \sigma'' \in \tau\ A\ \sigma'\ a \Longrightarrow \sigma'' \in \mathcal{R}_{nd}\ A\ \sigma$›

**lemma** $\mathcal{R}_d\text{-}trans$: ‹$\sigma'' \in \mathcal{R}_d\ A\ \sigma' \Longrightarrow \sigma' \in \mathcal{R}_d\ A\ \sigma \Longrightarrow \sigma'' \in \mathcal{R}_d\ A\ \sigma$›
  ⟨*proof*⟩

**lemma** $\mathcal{R}_{nd}\text{-}trans$: ‹$\sigma'' \in \mathcal{R}_{nd}\ A\ \sigma' \Longrightarrow \sigma' \in \mathcal{R}_{nd}\ A\ \sigma \Longrightarrow \sigma'' \in \mathcal{R}_{nd}\ A\ \sigma$›
  ⟨*proof*⟩

### 3.1.5 Morphisms

Our morphisms are defined considering that, except from $\tau$, the fields remain unchanged.

**definition** *from-det-to-ndet* ::
  ‹$('\sigma, 'a, 'r, '\alpha)\ A_d\text{-}scheme \Rightarrow ('\sigma, 'a, 'r, '\alpha)\ A_{nd}\text{-}scheme$›
  **where** ‹*from-det-to-ndet* $A \equiv$
    $(\!|\tau = \lambda\sigma\ a.\ case\ \tau\ A\ \sigma\ a\ of\ \lfloor\sigma'\rfloor \Rightarrow \{\sigma'\} \mid \Diamond \Rightarrow \{\},$
    $\omega = \lambda\sigma.\ case\ \omega\ A\ \sigma\ of\ \lfloor r\rfloor \Rightarrow \{r\} \mid \Diamond \Rightarrow \{\}, \ldots = more\ A|\!)$›

**definition** *from-ndet-to-det* ::
‹$('\sigma, 'a, 'r, '\alpha)\ A_{nd}$-*scheme* $\Rightarrow$ $('\sigma, 'a, 'r, '\alpha)\ A_d$-*scheme*›
**where** ‹*from-ndet-to-det* $A \equiv$
$(\!|\tau = \lambda\sigma\ a.\ if\ \tau\ A\ \sigma\ a = \{\}\ then\ \Diamond\ else\ \lfloor THE\ \sigma'.\ \sigma' \in \tau\ A\ \sigma\ a\rfloor,$
$\omega = \lambda\sigma.\ if\ \omega\ A\ \sigma = \{\}\ then\ \Diamond\ else\ \lfloor THE\ r.\ r \in \omega\ A\ \sigma\rfloor,\ \ldots = more\ A|\!)$›

**definition** *from-σ-to-σs*$_d$ ::
‹$('\sigma, 'a, 'r, '\alpha)\ A_d$-*scheme* $\Rightarrow$ $('\sigma\ list, 'a, 'r, '\alpha)\ A_d$-*scheme*›
**where** ‹*from-σ-to-σs*$_d$ $A \equiv$
$(\!|\tau = \lambda\sigma s\ a.\ case\ \tau\ A\ (hd\ \sigma s)\ a\ of\ \lfloor\sigma'\rfloor \Rightarrow \lfloor[\sigma']\rfloor \mid \Diamond \Rightarrow \Diamond,$
$\omega = \lambda\sigma s.\ \omega\ A\ (hd\ \sigma s),\ \ldots = more\ A|\!)$›
**definition** *from-σ-to-σs*$_{nd}$ ::
‹$('\sigma, 'a, 'r, '\alpha)\ A_{nd}$-*scheme* $\Rightarrow$ $('\sigma\ list, 'a, 'r, '\alpha)\ A_{nd}$-*scheme*›
**where** ‹*from-σ-to-σs*$_{nd}$ $A \equiv$
$(\!|\tau = \lambda\sigma s\ a.\ \{[\sigma'] \mid \sigma'.\ \sigma' \in \tau\ A\ (hd\ \sigma s)\ a\},$
$\omega = \lambda\sigma s.\ \omega\ A\ (hd\ \sigma s),\ \ldots = more\ A|\!)$›

**definition** *from-σs-to-σ*$_d$ ::
‹$('\sigma\ list, 'a, 'r, '\alpha)\ A_d$-*scheme* $\Rightarrow$ $('\sigma, 'a, 'r, '\alpha)\ A_d$-*scheme*›
**where** ‹*from-σs-to-σ*$_d$ $A \equiv$
$(\!|\tau = \lambda\sigma\ a.\ case\ \tau\ A\ [\sigma]\ a\ of\ \lfloor\sigma s'\rfloor \Rightarrow \lfloor hd\ \sigma s'\rfloor \mid \Diamond \Rightarrow \Diamond,$
$\omega = \lambda\sigma.\ \omega\ A\ [\sigma],\ \ldots = more\ A|\!)$›
**definition** *from-σs-to-σ*$_{nd}$ ::
‹$('\sigma\ list, 'a, 'r, '\alpha)\ A_{nd}$-*scheme* $\Rightarrow$ $('\sigma, 'a, 'r, '\alpha)\ A_{nd}$-*scheme*›
**where** ‹*from-σs-to-σ*$_{nd}$ $A \equiv$
$(\!|\tau = \lambda\sigma\ a.\ \{hd\ \sigma s' \mid \sigma s'.\ \sigma s' \in \tau\ A\ [\sigma]\ a\},$
$\omega = \lambda\sigma.\ \omega\ A\ [\sigma],\ \ldots = more\ A|\!)$›

**definition** *from-singl-to-list*$_d$ ::
‹$('\sigma, 'a, 'r, '\alpha)\ A_d$-*scheme* $\Rightarrow$ $('\sigma\ list, 'a, 'r\ list, '\alpha)\ A_d$-*scheme*›
**where** ‹*from-singl-to-list*$_d$ $A \equiv$
$(\!|\tau = \lambda\sigma s\ a.\ case\ \tau\ A\ (hd\ \sigma s)\ a\ of\ \lfloor\sigma'\rfloor \Rightarrow \lfloor[\sigma']\rfloor \mid \Diamond \Rightarrow \Diamond,$
$\omega = \lambda\sigma s.\ case\ \omega\ A\ (hd\ \sigma s)\ of\ \lfloor r\rfloor \Rightarrow \lfloor[r]\rfloor \mid \Diamond \Rightarrow \Diamond,\ \ldots = more\ A|\!)$›
**definition** *from-singl-to-list*$_{nd}$ ::
‹$('\sigma, 'a, 'r, '\alpha)\ A_{nd}$-*scheme* $\Rightarrow$ $('\sigma\ list, 'a, 'r\ list, '\alpha)\ A_{nd}$-*scheme*›
**where** ‹*from-singl-to-list*$_{nd}$ $A \equiv$
$(\!|\tau = \lambda\sigma s\ a.\ \{[\sigma'] \mid \sigma'.\ \sigma' \in \tau\ A\ (hd\ \sigma s)\ a\},$
$\omega = \lambda\sigma s.\ \{[r] \mid r.\ r \in \omega\ A\ (hd\ \sigma s)\},\ \ldots = more\ A|\!)$›

**definition** *from-list-to-singl*$_d$ ::
‹$('\sigma\ list, 'a, 'r\ list, '\alpha)\ A_d$-*scheme* $\Rightarrow$ $('\sigma, 'a, 'r, '\alpha)\ A_d$-*scheme*›
**where** ‹*from-list-to-singl*$_d$ $A \equiv$
$(\!|\tau = \lambda\sigma\ a.\ case\ \tau\ A\ [\sigma]\ a\ of\ \lfloor\sigma s'\rfloor \Rightarrow \lfloor hd\ \sigma s'\rfloor \mid \Diamond \Rightarrow \Diamond,$
$\omega = \lambda\sigma.\ case\ \omega\ A\ [\sigma]\ of\ \lfloor rs\rfloor \Rightarrow \lfloor hd\ rs\rfloor \mid \Diamond \Rightarrow \Diamond,\ \ldots = more\ A|\!)$›
**definition** *from-list-to-singl*$_{nd}$ ::
‹$('\sigma\ list, 'a, 'r\ list, '\alpha)\ A_{nd}$-*scheme* $\Rightarrow$ $('\sigma, 'a, 'r, '\alpha)\ A_{nd}$-*scheme*›
**where** ‹*from-list-to-singl*$_{nd}$ $A \equiv$
$(\!|\tau = \lambda\sigma\ a.\ \{hd\ \sigma s' \mid \sigma s'.\ \sigma s' \in \tau\ A\ [\sigma]\ a\},$
$\omega = \lambda\sigma.\ \{hd\ rs \mid rs.\ rs \in \omega\ A\ [\sigma]\},\ \ldots = more\ A|\!)$›

**lemmas** *det-ndet-conv-defs* = *from-det-to-ndet-def from-ndet-to-det-def*
**and** *σ-σs-conv-defs* = *from-σ-to-σs$_d$-def from-σ-to-σs$_{nd}$-def*
*from-σs-to-σ$_d$-def from-σs-to-σ$_{nd}$-def*
**and** *singl-list-conv-defs* = *from-singl-to-list$_d$-def from-singl-to-list$_{nd}$-def*
*from-list-to-singl$_d$-def from-list-to-singl$_{nd}$-def*

**bundle** *functional-automata-morphisms-syntax* **begin**

**notation** *from-det-to-ndet* (‹$\langle\!\langle$-$\rangle\!\rangle_{d\hookrightarrow nd}$› [*0*])
**notation** *from-ndet-to-det* (‹$\langle\!\langle$-$\rangle\!\rangle_{nd\rightsquigarrow d}$› [*0*])
**notation** *from-σ-to-σs$_d$* (‹$_d\langle\!\langle$-$\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$› [*0*])
**notation** *from-σ-to-σs$_{nd}$* (‹$_{nd}\langle\!\langle$-$\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$› [*0*])
**notation** *from-σs-to-σ$_d$* (‹$_d\langle\!\langle$-$\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}$› [*0*])
**notation** *from-σs-to-σ$_{nd}$* (‹$_{nd}\langle\!\langle$-$\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}$› [*0*])
**notation** *from-singl-to-list$_d$* (‹$_d\langle\!\langle$-$\rangle\!\rangle_{singl\hookrightarrow list}$› [*0*])
**notation** *from-singl-to-list$_{nd}$* (‹$_{nd}\langle\!\langle$-$\rangle\!\rangle_{singl\hookrightarrow list}$› [*0*])
**notation** *from-list-to-singl$_d$* (‹$_d\langle\!\langle$-$\rangle\!\rangle_{list\rightsquigarrow singl}$› [*0*])
**notation** *from-list-to-singl$_{nd}$* (‹$_{nd}\langle\!\langle$-$\rangle\!\rangle_{list\rightsquigarrow singl}$› [*0*])

**end**

**unbundle** *functional-automata-morphisms-syntax*

**lemma** *morphisms-A-scheme-more-simps* [*simp*] :
‹*more* $\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}$ = *more A*› ‹*more* $\langle\!\langle B\rangle\!\rangle_{nd\rightsquigarrow d}$ = *more B*›
‹*more* $_d\langle\!\langle C\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$ = *more C*› ‹*more* $_{nd}\langle\!\langle D\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$ = *more D*›
‹*more* $_d\langle\!\langle E\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}$ = *more E*› ‹*more* $_{nd}\langle\!\langle F\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}$ = *more F*›
‹*more* $_d\langle\!\langle G\rangle\!\rangle_{singl\hookrightarrow list}$ = *more G*› ‹*more* $_{nd}\langle\!\langle H\rangle\!\rangle_{singl\hookrightarrow list}$ = *more H*›
‹*more* $_d\langle\!\langle I\rangle\!\rangle_{list\rightsquigarrow singl}$ = *more I*› ‹*more* $_{nd}\langle\!\langle J\rangle\!\rangle_{list\rightsquigarrow singl}$ = *more J*›
⟨*proof*⟩

### 3.1.6 Generic update Functions

**definition** *update-both* **where** ‹*update-both* $A_0$ $A_1$ $\sigma_0$ $\sigma_1$ $e$ $f$ ≡ $f$ ($\tau$ $A_0$ $\sigma_0$ $e$) ($\tau$ $A_1$ $\sigma_1$ $e$)›

**definition** *update-left* **where** ‹*update-left* $A_0$ $\sigma_0$ $\sigma_1$ $e$ $f$ $g$ ≡ $f$ ($\tau$ $A_0$ $\sigma_0$ $e$) ($g$ $\sigma_1$)›

**definition** *update-right* **where** ‹*update-right* $A_1$ $\sigma_0$ $\sigma_1$ $e$ $f$ $g$ ≡ $f$ ($g$ $\sigma_0$) ($\tau$ $A_1$ $\sigma_1$ $e$)›

**lemmas** *update-defs*[*simp*] = *update-both-def update-left-def update-right-def*

**abbreviation** *f-up-set* **where** ‹*f-up-set* $f$ $B$ $C$ ≡ {$f$ $s$ $t$| $s$ $t$. ($s$, $t$) ∈ $B$ × $C$}›

**abbreviation** *f-up-opt* **where** ‹*f-up-opt f s t* ≡ *case s of* ◊ ⇒ ◊ | ⌊*s'*⌋ ⇒ *map-option* (*f s'*) *t*›

### 3.1.7 Assumptions on Automata

**definition** *finite-trans* :: ‹($'\sigma$, $'a$, $'r$, $'\alpha$) $A_{nd}$*-scheme* ⇒ *bool*›
**where** ‹*finite-trans A* ≡ ∀ $\sigma$ *a*. *finite* ($\tau$ *A* $\sigma$ *a*)›

**lemma** *finite-trans-morphisms-simps*[*simp*]:
‹*finite-trans* ⟪*A*⟫$_{d\hookrightarrow nd}$›
‹*finite-trans B* ⟹ *finite-trans* $_{nd}$⟪*B*⟫$_{\sigma\hookrightarrow\sigma s}$›
‹*finite-trans C* ⟹ *finite-trans* $_{nd}$⟪*C*⟫$_{\sigma s\rightsquigarrow\sigma}$›
‹*finite-trans D* ⟹ *finite-trans* $_{nd}$⟪*D*⟫$_{singl\hookrightarrow list}$›
‹*finite-trans E* ⟹ *finite-trans* $_{nd}$⟪*E*⟫$_{list\rightsquigarrow singl}$›
⟨*proof*⟩

**definition** *at-most-1-elem* :: ‹($'\sigma$, $'a$, $'r$, $'\alpha$) $A_{nd}$*-scheme* ⇒ *bool*›
**where** ‹*at-most-1-elem A* ≡
(∀ $\sigma$ *a*. $\tau$ *A* $\sigma$ *a* = {} ∨ (∃ $\sigma'$. $\tau$ *A* $\sigma$ *a* = {$\sigma'$})) ∧
(∀ $\sigma$. $\omega$ *A* $\sigma$ = {} ∨ (∃ *r*. $\omega$ *A* $\sigma$ = {*r*}))›

**lemma** *at-most-1-elem-def-bis* :
‹*at-most-1-elem A* ⟷ (∀ $\sigma$ *a*. ∃ $\sigma'$. $\tau$ *A* $\sigma$ *a* ⊆ {$\sigma'$}) ∧ (∀ $\sigma$. ∃ *r*. $\omega$ *A* $\sigma$ ⊆ {*r*})›
⟨*proof*⟩

**lemma** *at-most-1-elemI* :
‹⟦⋀$\sigma$ *a*. $\tau$ *A* $\sigma$ *a* = {} ∨ (∃ $\sigma'$. $\tau$ *A* $\sigma$ *a* = {$\sigma'$});
⋀$\sigma$. $\omega$ *A* $\sigma$ = {} ∨ (∃ *r*. $\omega$ *A* $\sigma$ = {*r*})⟧ ⟹ *at-most-1-elem A*›
⟨*proof*⟩

**lemma** *at-most-1-elemE* :
‹⟦$\tau$ *A* $\sigma$ *a* = {} ⟹ *thesis*; ⋀$\sigma'$. $\tau$ *A* $\sigma$ *a* = {$\sigma'$} ⟹ *thesis*⟧ ⟹ *thesis*›
‹⟦$\omega$ *A* $\sigma$ = {} ⟹ *thesis*; ⋀*r*. $\omega$ *A* $\sigma$ = {*r*} ⟹ *thesis*⟧ ⟹ *thesis*›
**if** ‹*at-most-1-elem A*›
⟨*proof*⟩

**definition** *at-most-1-elem-trans* :: ‹($'\sigma$, $'a$, $'r$, $'\alpha$) $A_{nd}$*-scheme* ⇒ *bool*›
**where** ‹*at-most-1-elem-trans A* ≡ ∀ $\sigma$ *a*. $\tau$ *A* $\sigma$ *a* = {} ∨ (∃ $\sigma'$. $\tau$ *A* $\sigma$ *a* = {$\sigma'$})›

**lemma** *at-most-1-elem-trans-def-bis* :
‹*at-most-1-elem-trans A* ⟷ (∀ $\sigma$ *a*. ∃ $\sigma'$. $\tau$ *A* $\sigma$ *a* ⊆ {$\sigma'$})›
⟨*proof*⟩

**lemma** *at-most-1-elem-transI* :
‹⟦⋀$\sigma$ *a*. $\tau$ *A* $\sigma$ *a* = {} ∨ (∃ $\sigma'$. $\tau$ *A* $\sigma$ *a* = {$\sigma'$})⟧ ⟹ *at-most-1-elem-trans A*›
⟨*proof*⟩

**lemma** *at-most-1-elem-transE* :
‹$[\![\tau\ A\ \sigma\ a = \{\} \Longrightarrow thesis;\ \bigwedge\sigma'.\ \tau\ A\ \sigma\ a = \{\sigma'\} \Longrightarrow thesis]\!] \Longrightarrow thesis$›
**if** ‹*at-most-1-elem-trans A*›
⟨*proof*⟩

**lemma** *at-most-1-elem-imp-at-most-1-elem-trans* :
‹*at-most-1-elem A* $\Longrightarrow$ *at-most-1-elem-trans A*›
⟨*proof*⟩

**definition** *length-1-trans*$_d$ :: ‹$('\sigma\ list,\ 'a,\ 'r,\ '\alpha)\ A_d$*-scheme* $\Rightarrow$ *bool*›
**where** ‹*length-1-trans*$_d$ $A \equiv$
$\forall\sigma s\ a.\ case\ \tau\ A\ \sigma s\ a\ of\ \Diamond \Rightarrow True \mid \lfloor\sigma s'\rfloor \Rightarrow length\ \sigma s' = Suc\ 0$›

**lemma** *length-1-trans*$_d I$ :
‹$[\![\bigwedge\sigma s\ a\ \sigma s'.\ \tau\ A\ \sigma s\ a = \lfloor\sigma s'\rfloor \Longrightarrow length\ \sigma s' = Suc\ 0]\!] \Longrightarrow$ *length-1-trans*$_d$ $A$›
⟨*proof*⟩

**lemma** *length-1-trans*$_d E$ :
‹$[\![$*length-1-trans*$_d$ $A;\ \tau\ A\ \sigma s\ a = \lfloor\sigma s'\rfloor;\ \bigwedge\sigma.\ \sigma s' = [\sigma] \Longrightarrow thesis]\!] \Longrightarrow thesis$›
⟨*proof*⟩

**definition** *length-1-trans*$_{nd}$ :: ‹$('\sigma\ list,\ 'a,\ 'r,\ '\alpha)\ A_{nd}$*-scheme* $\Rightarrow$ *bool*›
**where** ‹*length-1-trans*$_{nd}$ $A \equiv \forall\sigma s\ a.\ \forall\sigma s' \in \tau\ A\ \sigma s\ a.\ length\ \sigma s' = Suc\ 0$›

**lemma** *length-1-trans*$_{nd} I$ :
‹$[\![\bigwedge\sigma s\ a\ \sigma s'.\ \sigma s' \in \tau\ A\ \sigma s\ a \Longrightarrow length\ \sigma s' = Suc\ 0]\!] \Longrightarrow$ *length-1-trans*$_{nd}$ $A$›
⟨*proof*⟩

**lemma** *length-1-trans*$_{nd} E$ :
‹$[\![$*length-1-trans*$_{nd}$ $A;\ \sigma s' \in \tau\ A\ \sigma s\ a;\ \bigwedge\sigma.\ \sigma s' = [\sigma] \Longrightarrow thesis]\!] \Longrightarrow thesis$›
⟨*proof*⟩

**definition** *length-1*$_d$ :: ‹$('\sigma\ list,\ 'a,\ 'r\ list,\ '\alpha)\ A_d$*-scheme* $\Rightarrow$ *bool*›
**where** ‹*length-1*$_d$ $A \equiv$
$(\forall\sigma s\ a.\ case\ \tau\ A\ \sigma s\ a\ of\ \Diamond \Rightarrow True \mid \lfloor\sigma s'\rfloor \Rightarrow length\ \sigma s' = Suc\ 0) \wedge$
$(\forall\sigma s.\ case\ \omega\ A\ \sigma s\ of\ \Diamond \Rightarrow True \mid \lfloor rs\rfloor \Rightarrow length\ rs = Suc\ 0)$›

**lemma** *length-1*$_d I$ :
‹$[\![\bigwedge\sigma s\ a\ \sigma s'.\ \tau\ A\ \sigma s\ a = \lfloor\sigma s'\rfloor \Longrightarrow length\ \sigma s' = Suc\ 0;$
$\bigwedge\sigma s\ rs.\ \omega\ A\ \sigma s = \lfloor rs\rfloor \Longrightarrow length\ rs = Suc\ 0]\!] \Longrightarrow$ *length-1*$_d$ $A$›
⟨*proof*⟩

**lemma** *length-1*$_d E$ :
‹$[\![$*length-1*$_d$ $A;\ \tau\ A\ \sigma s\ a = \lfloor\sigma s'\rfloor;\ \bigwedge\sigma.\ \sigma s' = [\sigma] \Longrightarrow thesis]\!] \Longrightarrow thesis$›

‹$[\![$*length-1*$_d$ $A$; $\omega$ $A$ $\sigma s$ = $\lfloor rs \rfloor$; $\bigwedge r.$ $rs$ = $[r]$ $\Longrightarrow$ *thesis*$]\!] \Longrightarrow$ *thesis*›
⟨*proof*⟩

**definition** *length-1*$_{nd}$ :: ‹($'\sigma$ *list*, $'a$, $'r$ *list*, $'\alpha$) $A_{nd}$*-scheme* $\Rightarrow$ *bool*›
**where** ‹*length-1*$_{nd}$ $A \equiv (\forall \sigma s\ a.\ \forall \sigma s' \in \tau\ A\ \sigma s\ a.\ length\ \sigma s' = Suc\ 0) \wedge$
$(\forall \sigma s.\ \forall rs \in \omega\ A\ \sigma s.\ length\ rs = Suc\ 0)$›

**lemma** *length-1*$_{nd}$*I* :
‹$[\![\bigwedge \sigma s\ a\ \sigma s'.\ \sigma s' \in \tau\ A\ \sigma s\ a \Longrightarrow length\ \sigma s' = Suc\ 0;$
$\bigwedge \sigma s\ rs.\ rs \in \omega\ A\ \sigma s \Longrightarrow length\ rs = Suc\ 0]\!] \Longrightarrow$ *length-1*$_{nd}$ $A$›
⟨*proof*⟩

**lemma** *length-1*$_{nd}$*E* :
‹$[\![$*length-1*$_{nd}$ $A$; $\sigma s' \in \tau\ A\ \sigma s\ a$; $\bigwedge \sigma.\ \sigma s' = [\sigma] \Longrightarrow$ *thesis*$]\!] \Longrightarrow$ *thesis*›
‹$[\![$*length-1*$_{nd}$ $A$; $rs \in \omega\ A\ \sigma s$; $\bigwedge r.\ rs = [r] \Longrightarrow$ *thesis*$]\!] \Longrightarrow$ *thesis*›
⟨*proof*⟩

**definition** *indep-enabl* :: ‹($'\sigma_0$, $'a$, $'r_0$, $'\alpha$) $A_d$*-scheme* $\Rightarrow$ $'\sigma_0$ $\Rightarrow$ $'a$ *set* $\Rightarrow$ ($'\sigma_1$, $'a$, $'r_1$, $'\beta$) $A_d$*-scheme* $\Rightarrow$ $'\sigma_1$ $\Rightarrow$ *bool*›
**where** ‹*indep-enabl* $A_0$ $\sigma_0$ $E$ $A_1$ $\sigma_1 \equiv \forall t_0 \in \mathcal{R}_d\ A_0\ \sigma_0.\ \forall t_1 \in \mathcal{R}_d\ A_1\ \sigma_1.\ \varepsilon\ A_0\ t_0 \cap \varepsilon\ A_1\ t_1 \subseteq E$›

**lemma** *indep-enablI* :
‹$(\bigwedge t_0\ t_1.\ t_0 \in \mathcal{R}_d\ A_0\ \sigma_0 \Longrightarrow t_1 \in \mathcal{R}_d\ A_1\ \sigma_1 \Longrightarrow \varepsilon\ A_0\ t_0 \cap \varepsilon\ A_1\ t_1 \subseteq E)$
$\Longrightarrow$ *indep-enabl* $A_0$ $\sigma_0$ $E$ $A_1$ $\sigma_1$›
**and** *indep-enablD* :
‹$[\![$*indep-enabl* $A_0$ $\sigma_0$ $E$ $A_1$ $\sigma_1$; $t_0 \in \mathcal{R}_d\ A_0\ \sigma_0$; $t_1 \in \mathcal{R}_d\ A_1\ \sigma_1]\!] \Longrightarrow \varepsilon\ A_0\ t_0 \cap \varepsilon\ A_1\ t_1 \subseteq E$›
⟨*proof*⟩

**definition** *ϱ-disjoint-ε* :: ‹($'\sigma$, $'a$, $'\sigma'$, $'r'$, $'\alpha$) *A-scheme* $\Rightarrow$ *bool*›
**where** ‹*ϱ-disjoint-ε* $A \equiv \forall \sigma \in \varrho\ A.\ \varepsilon\ A\ \sigma = \{\}$›

**lemma** *ϱ-disjoint-εI* : ‹$(\bigwedge \sigma.\ \sigma \in \varrho\ A \Longrightarrow \varepsilon\ A\ \sigma = \{\}) \Longrightarrow$ *ϱ-disjoint-ε* $A$›
**and** *ϱ-disjoint-εD* : ‹*ϱ-disjoint-ε* $A \Longrightarrow \sigma \in \varrho\ A \Longrightarrow \varepsilon\ A\ \sigma = \{\}$›
⟨*proof*⟩

**definition** *at-most-1-elem-term* :: ‹($'\sigma$, $'a$, $'r$, $'\alpha$) $A_{nd}$*-scheme* $\Rightarrow$ *bool*›
**where** ‹*at-most-1-elem-term* $A \equiv \forall \sigma.\ \omega\ A\ \sigma = \{\} \vee (\exists r.\ \omega\ A\ \sigma = \{r\})$›

**lemma** *at-most-1-elem-term-def-bis* :

‹*at-most-1-elem-term A* $\longleftrightarrow$ $(\forall \sigma.\ \exists r.\ \omega\ A\ \sigma \subseteq \{r\})$›
⟨*proof*⟩

**lemma** *at-most-1-elem-termI* :
‹$[\![\bigwedge\sigma.\ \omega\ A\ \sigma = \{\} \vee (\exists r.\ \omega\ A\ \sigma = \{r\})]\!] \Longrightarrow$ *at-most-1-elem-term A*›
⟨*proof*⟩

**lemma** *at-most-1-elem-termE* :
‹$[\![\omega\ A\ \sigma = \{\} \Longrightarrow thesis;\ \bigwedge r.\ \omega\ A\ \sigma = \{r\} \Longrightarrow thesis]\!] \Longrightarrow thesis$›
**if** ‹*at-most-1-elem-term A*›
⟨*proof*⟩

**lemma** *at-most-1-elem-imp-at-most-1-elem-term* :
‹*at-most-1-elem A* $\Longrightarrow$ *at-most-1-elem-term A*›
⟨*proof*⟩

## 3.2 First Properties

### 3.2.1 $\varepsilon$, $\varrho$ and $\omega$ first equalities

**lemma** *base-trans-ε*[*simp*]:
‹$\varepsilon\ ((\!|\tau = \lambda\sigma\ a.\ \Diamond, \omega = \lambda\sigma.\ \Diamond, \ldots = some|\!) :: ('\sigma, 'a, 'r, '\alpha)\ A_d\text{-}scheme)\ \sigma = \{\}$›
‹$\varepsilon\ ((\!|\tau = \lambda\sigma\ a.\ \{\}, \omega = \lambda\sigma.\ \{\}, \ldots = some|\!) :: ('\sigma, 'a, 'r, '\alpha)\ A_{nd}\text{-}scheme)\ \sigma = \{\}$›
⟨*proof*⟩

**lemma** *base-trans-ϱ*[*simp*]:
‹$\varrho\ ((\!|\tau = \lambda\sigma\ a.\ \Diamond, \omega = \lambda\sigma.\ \Diamond, \ldots = some|\!) :: ('\sigma, 'a, 'r, '\alpha)\ A_d\text{-}scheme) = \{\}$›
‹$\varrho\ ((\!|\tau = \lambda\sigma\ a.\ \{\}, \omega = \lambda\sigma.\ \{\}, \ldots = some|\!) :: ('\sigma, 'a, 'r, '\alpha)\ A_{nd}\text{-}scheme) = \{\}$›
⟨*proof*⟩

**lemma** *σ-σs-conv-ε*[*simp*]:
‹$\varepsilon\ {}_d\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ \sigma s = \varepsilon\ A\ (hd\ \sigma s)$› ‹$\varepsilon\ {}_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ \sigma s = \varepsilon\ B\ (hd\ \sigma s)$›
‹$\varepsilon\ {}_d\langle\!\langle C\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\ \sigma = \varepsilon\ C\ [\sigma]$› ‹$\varepsilon\ {}_{nd}\langle\!\langle D\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\ \sigma = \varepsilon\ D\ [\sigma]$›
⟨*proof*⟩

**lemma** *σ-σs-conv-ϱ*[*simp*]:
‹$\varrho\ {}_d\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s} = \{\sigma s.\ hd\ \sigma s \in \varrho\ A\}$› ‹$\varrho\ {}_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s} = \{\sigma s.\ hd\ \sigma s \in \varrho\ B\}$›
‹$\varrho\ {}_d\langle\!\langle C\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma} = \{\sigma.\ [\sigma] \in \varrho\ C\}$› ‹$\varrho\ {}_{nd}\langle\!\langle D\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma} = \{\sigma.\ [\sigma] \in \varrho\ D\}$›
⟨*proof*⟩

**lemma** *singl-list-conv-ε*[*simp*]:
‹$\varepsilon\ {}_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}\ \sigma s = \varepsilon\ A\ (hd\ \sigma s)$› ‹$\varepsilon\ {}_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}\ \sigma s = \varepsilon\ B\ (hd\ \sigma s)$›
‹$\varepsilon\ {}_d\langle\!\langle C\rangle\!\rangle_{list\rightsquigarrow singl}\ \sigma = \varepsilon\ C\ [\sigma]$› ‹$\varepsilon\ {}_{nd}\langle\!\langle D\rangle\!\rangle_{list\rightsquigarrow singl}\ \sigma = \varepsilon\ D\ [\sigma]$›
⟨*proof*⟩

**lemma** *singl-list-conv-ϱ*[*simp*]:

‹$\varrho$ $_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list} = \{\sigma s.\ hd\ \sigma s \in \varrho\ A\}$› ‹$\varrho$ $_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list} = \{\sigma s.\ hd\ \sigma s \in \varrho\ B\}$›
‹$\varrho$ $_d\langle\!\langle C\rangle\!\rangle_{list\rightsquigarrow singl} = \{\sigma.\ [\sigma] \in \varrho\ C\}$› ‹$\varrho$ $_{nd}\langle\!\langle D\rangle\!\rangle_{list\rightsquigarrow singl} = \{\sigma.\ [\sigma] \in \varrho\ D\}$›
⟨*proof*⟩

**lemma** *det-ndet-conv-ε*[*simp*]: ‹$\varepsilon$ $\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd} = \varepsilon\ A$› ‹$\varepsilon$ $\langle\!\langle B\rangle\!\rangle_{nd\rightsquigarrow d} = \varepsilon\ B$›
⟨*proof*⟩

**lemma** *det-ndet-conv-ϱ*[*simp*]: ‹$\varrho$ $\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd} = \varrho\ A$› ‹$\varrho$ $\langle\!\langle B\rangle\!\rangle_{nd\rightsquigarrow d} = \varrho\ B$›
⟨*proof*⟩

**lemma** *ω-from-det-to-ndet* :
‹$\omega$ $\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd} = (\lambda\sigma.\ case\ \omega\ A\ \sigma\ of\ \lfloor r\rfloor \Rightarrow \{r\} \mid \Diamond \Rightarrow \{\})$›
⟨*proof*⟩

**lemma** *ε-ω-useless* [*simp*] :
‹$\varepsilon$ $(A(\!|\omega := some\text{-}\omega|\!)) = \varepsilon\ A$› ‹$\varepsilon$ $(B(\!|\omega := some\text{-}\omega'|\!)) = \varepsilon\ B$›
**for** $A$ :: ‹$('\sigma,\ 'a,\ '\sigma\ option,\ 'r\ option,\ '\alpha)$ *A-scheme*›
**and** $B$ :: ‹$('\sigma,\ 'a,\ '\sigma\ set,\ 'r\ set,\ '\alpha)$ *A-scheme*›
⟨*proof*⟩

**lemma** *ϱ-disjoint-ε-updated-ω* [*simp*] :
‹*ϱ-disjoint-ε* $(A(\!|\omega := \lambda\sigma.\ \Diamond|\!))$›
‹*ϱ-disjoint-ε* $(B(\!|\omega := \lambda\sigma.\ \{\}|\!))$›
⟨*proof*⟩

**lemma** *ϱ-disjoint-ε-det-ndet-conv-iff* [*simp*] :
‹*ϱ-disjoint-ε* $\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd} \longleftrightarrow$ *ϱ-disjoint-ε* $A$›
‹*ϱ-disjoint-ε* $\langle\!\langle B\rangle\!\rangle_{nd\rightsquigarrow d} \longleftrightarrow$ *ϱ-disjoint-ε* $B$›
⟨*proof*⟩

**lemma** *at-most-1-elem-term-updated-ω* [*simp*] :
‹*at-most-1-elem-term* $(A(\!|\omega := \lambda\sigma.\ \{\}|\!))$›
⟨*proof*⟩

**lemma** *at-most-1-elem-term-from-det-to-ndet* [*simp*] :
‹*at-most-1-elem-term* $\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}$›
⟨*proof*⟩

**lemma** *at-most-1-elem-term-unit* [*simp*] :
‹*at-most-1-elem-term* $(A :: ('\sigma,\ 'a,\ unit,\ '\alpha)\ A_{nd}\text{-}scheme)$›
⟨*proof*⟩

### 3.2.2 Properties of our morphisms

**method** *expand-A-scheme* =
*match* **conclusion in** ‹$A = B$› **for** $A\ B$ :: ‹$('\sigma,\ 'a,\ '\sigma',\ 'r',\ '\alpha)$ *A-scheme*› ⇒

‹*cases A*, *cases B*›

**lemma** *base-trans-det-ndet-conv*:
‹$\langle\!\langle(\!|\tau = \lambda\sigma\ a.\ \Diamond,\ \omega = \lambda\sigma.\ \Diamond,\ \ldots = some|\!)\rangle\!\rangle_{d\hookrightarrow nd} =$
$(\!|\tau = \lambda\sigma\ a.\ \{\},\ \omega = \lambda\sigma.\ \{\},\ \ldots = some|\!)$›
‹$\langle\!\langle(\!|\tau = \lambda\sigma\ a.\ \{\},\ \omega = \lambda\sigma.\ \{\},\ \ldots = some|\!)\rangle\!\rangle_{nd\rightsquigarrow d} =$
$(\!|\tau = \lambda\sigma\ a.\ \Diamond,\ \omega = \lambda\sigma.\ \Diamond,\ \ldots = some|\!)$›
⟨*proof*⟩

**lemma** *from-det-to-ndet-σ-σs-conv-commute*:
‹${}_{nd}\langle\!\langle\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle_{\sigma\hookrightarrow\sigma s} = \langle\!\langle{}_{d}\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\rangle\!\rangle_{d\hookrightarrow nd}$› ‹${}_{nd}\langle\!\langle\langle\!\langle B\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma} = \langle\!\langle{}_{d}\langle\!\langle B\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\rangle\!\rangle_{d\hookrightarrow nd}$›
⟨*proof*⟩

**lemma** *from-det-to-ndet-singl-list-conv-commute*:
‹${}_{nd}\langle\!\langle\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle_{singl\hookrightarrow list} = \langle\!\langle{}_{d}\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}\rangle\!\rangle_{d\hookrightarrow nd}$› ‹${}_{nd}\langle\!\langle\langle\!\langle B\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle_{list\rightsquigarrow singl} =$
$\langle\!\langle{}_{d}\langle\!\langle B\rangle\!\rangle_{list\rightsquigarrow singl}\rangle\!\rangle_{d\hookrightarrow nd}$›
⟨*proof*⟩

**lemma** *from-ndet-to-det-σ-σs-conv-commute*:
‹*at-most-1-elem-trans A* $\Longrightarrow {}_{d}\langle\!\langle\langle\!\langle A\rangle\!\rangle_{nd\rightsquigarrow d}\rangle\!\rangle_{\sigma\hookrightarrow\sigma s} = \langle\!\langle{}_{nd}\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\rangle\!\rangle_{nd\rightsquigarrow d}$›
‹*at-most-1-elem-trans B* $\Longrightarrow {}_{d}\langle\!\langle\langle\!\langle B\rangle\!\rangle_{nd\rightsquigarrow d}\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma} = \langle\!\langle{}_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\rangle\!\rangle_{nd\rightsquigarrow d}$›
⟨*proof*⟩

**lemma** *from-ndet-to-det-singl-list-conv-commute*:
‹*at-most-1-elem A* $\Longrightarrow {}_{d}\langle\!\langle\langle\!\langle A\rangle\!\rangle_{nd\rightsquigarrow d}\rangle\!\rangle_{singl\hookrightarrow list} = \langle\!\langle{}_{nd}\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}\rangle\!\rangle_{nd\rightsquigarrow d}$›
‹*at-most-1-elem B* $\Longrightarrow {}_{d}\langle\!\langle\langle\!\langle B\rangle\!\rangle_{nd\rightsquigarrow d}\rangle\!\rangle_{list\rightsquigarrow singl} = \langle\!\langle{}_{nd}\langle\!\langle B\rangle\!\rangle_{list\rightsquigarrow singl}\rangle\!\rangle_{nd\rightsquigarrow d}$›
⟨*proof*⟩

**lemma** *behaviour-σ-σs-conv*:
‹$\varepsilon\ {}_{d}\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ [\sigma] = \varepsilon\ A\ \sigma$›
‹$\tau\ {}_{d}\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ [\sigma]\ a = (case\ \tau\ A\ \sigma\ a\ of\ \Diamond \Rightarrow \Diamond \mid \lfloor t\rfloor \Rightarrow \lfloor[t]\rfloor)$›
‹$\varrho\ {}_{d}\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s} = \{\sigma s.\ hd\ \sigma s \in \varrho\ A\}$›
‹$\omega\ {}_{d}\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ [\sigma] = \omega\ A\ \sigma$›
‹$\varepsilon\ {}_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ [\sigma] = \varepsilon\ B\ \sigma$›
‹$\tau\ {}_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ [\sigma]\ a = \{[\sigma'] \mid \sigma'.\ \sigma' \in \tau\ B\ \sigma\ a\}$›
‹$\varrho\ {}_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s} = \{\sigma s.\ hd\ \sigma s \in \varrho\ B\}$›
‹$\omega\ {}_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ [\sigma] = \omega\ B\ \sigma$›
‹$\varepsilon\ {}_{d}\langle\!\langle C\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\ \sigma = \varepsilon\ C\ [\sigma]$›
‹$\tau\ {}_{d}\langle\!\langle C\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\ \sigma\ a = (case\ \tau\ C\ [\sigma]\ a\ of\ \Diamond \Rightarrow \Diamond \mid \lfloor\sigma s'\rfloor \Rightarrow \lfloor hd\ \sigma s'\rfloor)$›
‹$\varrho\ {}_{d}\langle\!\langle C\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma} = \{\sigma.\ [\sigma] \in \varrho\ C\}$›
‹$\omega\ {}_{d}\langle\!\langle C\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\ \sigma = \omega\ C\ [\sigma]$›
‹$\varepsilon\ {}_{nd}\langle\!\langle D\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\ \sigma = \varepsilon\ D\ [\sigma]$›
‹$\tau\ {}_{nd}\langle\!\langle D\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\ \sigma\ a = \{hd\ \sigma s' \mid \sigma s'.\ \sigma s' \in \tau\ D\ [\sigma]\ a\}$›
‹$\varrho\ {}_{nd}\langle\!\langle D\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma} = \{\sigma.\ [\sigma] \in \varrho\ D\}$› ‹$\omega\ {}_{nd}\langle\!\langle D\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\ \sigma = \omega\ D\ [\sigma]$›

⟨*proof*⟩

**lemma** *behaviour-singl-list-conv*:
‹$\varepsilon\ {}_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}\ [\sigma] = \varepsilon\ A\ \sigma$›
‹$\tau\ {}_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}\ [\sigma]\ a = (case\ \tau\ A\ \sigma\ a\ of\ \Diamond \Rightarrow \Diamond \mid \lfloor t\rfloor \Rightarrow \lfloor[t]\rfloor)$›
‹$\varrho\ {}_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list} = \{\sigma s.\ hd\ \sigma s \in \varrho\ A\}$›
‹$\omega\ {}_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}\ [\sigma] = (case\ \omega\ A\ \sigma\ of\ \Diamond \Rightarrow \Diamond \mid \lfloor r\rfloor \Rightarrow \lfloor[r]\rfloor)$›
‹$\varepsilon\ {}_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}\ [\sigma] = \varepsilon\ B\ \sigma$›
‹$\tau\ {}_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}\ [\sigma]\ a = \{[\sigma']\ |\sigma'.\ \sigma' \in \tau\ B\ \sigma\ a\}$›
‹$\varrho\ {}_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list} = \{\sigma s.\ hd\ \sigma s \in \varrho\ B\}$›
‹$\omega\ {}_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}\ [\sigma] = \{[r]\ |r.\ r \in \omega\ B\ \sigma\}$›
‹$\varepsilon\ {}_d\langle\!\langle C\rangle\!\rangle_{list\rightsquigarrow singl}\ \sigma = \varepsilon\ C\ [\sigma]$›
‹$\tau\ {}_d\langle\!\langle C\rangle\!\rangle_{list\rightsquigarrow singl}\ \sigma\ a = (case\ \tau\ C\ [\sigma]\ a\ of\ \Diamond \Rightarrow \Diamond \mid \lfloor\sigma s'\rfloor \Rightarrow \lfloor hd\ \sigma s'\rfloor)$›
‹$\varrho\ {}_d\langle\!\langle C\rangle\!\rangle_{list\rightsquigarrow singl} = \{\sigma.\ [\sigma] \in \varrho\ C\}$›
‹$\omega\ {}_d\langle\!\langle C\rangle\!\rangle_{list\rightsquigarrow singl}\ \sigma = (case\ \omega\ C\ [\sigma]\ of\ \Diamond \Rightarrow \Diamond \mid \lfloor rs\rfloor \Rightarrow \lfloor hd\ rs\rfloor)$›
‹$\varepsilon\ {}_{nd}\langle\!\langle D\rangle\!\rangle_{list\rightsquigarrow singl}\ \sigma = \varepsilon\ D\ [\sigma]$›
‹$\tau\ {}_{nd}\langle\!\langle D\rangle\!\rangle_{list\rightsquigarrow singl}\ \sigma\ a = \{hd\ \sigma s' |\ \sigma s'.\ \sigma s' \in \tau\ D\ [\sigma]\ a\}$›
‹$\varrho\ {}_{nd}\langle\!\langle D\rangle\!\rangle_{list\rightsquigarrow singl} = \{\sigma.\ [\sigma] \in \varrho\ D\}$›
‹$\omega\ {}_{nd}\langle\!\langle D\rangle\!\rangle_{list\rightsquigarrow singl}\ \sigma = \{hd\ rs\ |rs.\ rs \in \omega\ D\ [\sigma]\}$›
⟨*proof*⟩

**lemma** *empty-from-det-to-ndet-is-None-trans* [*simp*] : ‹$\tau\ \langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}\ \sigma\ a = \{\} \longleftrightarrow \tau\ A\ \sigma\ a = \Diamond$›
⟨*proof*⟩

**lemma** *at-most-1-elem-from-det-to-ndet* [*simp*] : ‹*at-most-1-elem* $\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}$›
⟨*proof*⟩

**lemma** *from-ndet-to-det-from-det-to-ndet* [*simp*] : ‹$\langle\!\langle\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle_{nd\rightsquigarrow d} = A$›
⟨*proof*⟩

**lemma** *from-det-to-ndet-from-ndet-to-det* [*simp*] :
‹$\langle\!\langle\langle\!\langle A\rangle\!\rangle_{nd\rightsquigarrow d}\rangle\!\rangle_{d\hookrightarrow nd} = A$› **if** ‹*at-most-1-elem A*›
⟨*proof*⟩

**theorem** *bij-betw-from-det-to-ndet* :
‹*bij-betw* $(\lambda A.\ \langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd})$ *UNIV* $\{A.$ *at-most-1-elem A*$\}$›
⟨*proof*⟩

**lemma** *bij-betw-from-ndet-to-det* :
‹*bij-betw* $(\lambda A.\ \langle\!\langle A\rangle\!\rangle_{nd\rightsquigarrow d})$ $\{A.$ *at-most-1-elem A*$\}$ *UNIV*›
⟨*proof*⟩

**lemma** *length-1-trans-from-σ-to-σs* [*simp*] :
‹*length-1-trans*$_d$ $_d\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$› ‹*length-1-trans*$_{nd}$ $_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$›
⟨*proof*⟩

**lemma** *τ-hd-from-σ-to-σs-eq* [*simp*] :
‹$\tau$ $_d\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$ [*hd* $\sigma s$] $a = \tau$ $_d\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$ $\sigma s$ $a$›
‹$\tau$ $_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$ [*hd* $\sigma s$] $a = \tau$ $_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$ $\sigma s$ $a$›
⟨*proof*⟩

**lemma** *ω-hd-from-σ-to-σs-eq* [*simp*] :
‹$\omega$ $_d\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$ [*hd* $\sigma s$] $= \omega$ $_d\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$ $\sigma s$›
‹$\omega$ $_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$ [*hd* $\sigma s$] $= \omega$ $_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$ $\sigma s$›
⟨*proof*⟩

**lemma** *from-σs-to-σ-from-σ-to-σs* [*simp*] :
‹$_d\langle\!\langle {}_d\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma} = A$› ‹$_{nd}\langle\!\langle {}_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma} = B$›
⟨*proof*⟩

**lemma** *from-σ-to-σs-from-σs-to-σ* [*simp*] :
‹$[\![$*length-1-trans*$_d$ $A$; $\bigwedge\sigma s\ a.\ \tau\ A$ [*hd* $\sigma s$] $a = \tau\ A\ \sigma s\ a$;
$\bigwedge\sigma s.\ \omega\ A$ [*hd* $\sigma s$] $= \omega\ A\ \sigma s]\!] \Longrightarrow$ $_d\langle\!\langle {}_d\langle\!\langle A\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\rangle\!\rangle_{\sigma\hookrightarrow\sigma s} = A$›
‹$[\![$*length-1-trans*$_{nd}$ $B$; $\bigwedge\sigma s\ a.\ \tau\ B$ [*hd* $\sigma s$] $a = \tau\ B\ \sigma s\ a$;
$\bigwedge\sigma s.\ \omega\ B$ [*hd* $\sigma s$] $= \omega\ B\ \sigma s]\!] \Longrightarrow$ $_{nd}\langle\!\langle {}_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}\rangle\!\rangle_{\sigma\hookrightarrow\sigma s} = B$›
⟨*proof*⟩

**theorem** *bij-betw-from-σ-to-σs* :
‹*bij-betw* ($\lambda A.$ $_d\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$) *UNIV*
{$A$. *length-1-trans*$_d$ $A \wedge (\forall\sigma s\ a.\ \tau\ A$ [*hd* $\sigma s$] $a = \tau\ A\ \sigma s\ a) \wedge (\forall\sigma s.\ \omega\ A$ [*hd* $\sigma s$] $= \omega\ A\ \sigma s)$}›
(**is** ‹*bij-betw* ($\lambda A.$ $_d\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$) *UNIV* $?S_d$›)
‹*bij-betw* ($\lambda B.$ $_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$) *UNIV*
{$B$. *length-1-trans*$_{nd}$ $B \wedge (\forall\sigma s\ a.\ \tau\ B\ \sigma s\ a = \tau\ B$ [*hd* $\sigma s$] $a) \wedge (\forall\sigma s.\ \omega\ B$ [*hd* $\sigma s$] $= \omega\ B\ \sigma s)$}›
⟨*proof*⟩

**lemma** *bij-betw-from-σs-to-σ* :
‹*bij-betw* ($\lambda A.$ $_d\langle\!\langle A\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}$)
{$A$. *length-1-trans*$_d$ $A \wedge (\forall\sigma s\ a.\ \tau\ A$ [*hd* $\sigma s$] $a = \tau\ A\ \sigma s\ a) \wedge (\forall\sigma s.\ \omega\ A$ [*hd* $\sigma s$] $= \omega\ A\ \sigma s)$} *UNIV*›
‹*bij-betw* ($\lambda B.$ $_{nd}\langle\!\langle B\rangle\!\rangle_{\sigma s\rightsquigarrow\sigma}$)
{$B$. *length-1-trans*$_{nd}$ $B \wedge (\forall\sigma s\ a.\ \tau\ B\ \sigma s\ a = \tau\ B$ [*hd* $\sigma s$] $a) \wedge (\forall\sigma s.\ \omega\ B$ [*hd* $\sigma s$] $= \omega\ B\ \sigma s)$} *UNIV*›
⟨*proof*⟩

**lemma** *length-1-from-singl-to-list* [*simp*] :

‹*length-1*$_d$ $_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}$› ‹*length-1*$_{nd}$ $_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}$›
⟨*proof*⟩

**lemma** *τ-hd-from-singl-to-list-eq* [*simp*] :
‹$\tau$ $_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}$ [*hd* $\sigma s$] $a = \tau$ $_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}$ $\sigma s$ $a$›
‹$\tau$ $_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}$ [*hd* $\sigma s$] $a = \tau$ $_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}$ $\sigma s$ $a$›
⟨*proof*⟩

**lemma** *ω-hd-from-singl-to-list-eq* [*simp*] :
‹$\omega$ $_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}$ [*hd* $\sigma s$] $= \omega$ $_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}$ $\sigma s$›
‹$\omega$ $_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}$ [*hd* $\sigma s$] $= \omega$ $_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}$ $\sigma s$›
⟨*proof*⟩

**lemma** *from-list-to-singl-from-singl-to-list* [*simp*] :
‹$_d\langle\!\langle_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}\rangle\!\rangle_{list\rightsquigarrow singl} = A$› ‹$_{nd}\langle\!\langle_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}\rangle\!\rangle_{list\rightsquigarrow singl} = B$›
⟨*proof*⟩

**lemma** *from-singl-to-list-from-list-to-singl* [*simp*] :
‹$[\![$*length-1*$_d$ $A$; $\bigwedge\sigma s$ $a$. $\tau$ $A$ [*hd* $\sigma s$] $a = \tau$ $A$ $\sigma s$ $a$;
$\bigwedge\sigma s$. $\omega$ $A$ [*hd* $\sigma s$] $= \omega$ $A$ $\sigma s]\!] \Longrightarrow$ $_d\langle\!\langle_d\langle\!\langle A\rangle\!\rangle_{list\rightsquigarrow singl}\rangle\!\rangle_{singl\hookrightarrow list} = A$›
‹$[\![$*length-1*$_{nd}$ $B$; $\bigwedge\sigma s$ $a$. $\tau$ $B$ [*hd* $\sigma s$] $a = \tau$ $B$ $\sigma s$ $a$;
$\bigwedge\sigma s$. $\omega$ $B$ [*hd* $\sigma s$] $= \omega$ $B$ $\sigma s]\!] \Longrightarrow$ $_{nd}\langle\!\langle_{nd}\langle\!\langle B\rangle\!\rangle_{list\rightsquigarrow singl}\rangle\!\rangle_{singl\hookrightarrow list} = B$›
⟨*proof*⟩

**theorem** *bij-betw-from-singl-to-list* :
‹*bij-betw* ($\lambda A$. $_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}$) *UNIV*
{$A$. *length-1*$_d$ $A \wedge (\forall\sigma s$ $a$. $\tau$ $A$ [*hd* $\sigma s$] $a = \tau$ $A$ $\sigma s$ $a) \wedge (\forall\sigma s$. $\omega$ $A$ [*hd* $\sigma s$] $=$ $\omega$ $A$ $\sigma s)$}›
(**is** ‹*bij-betw* ($\lambda A$. $_d\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}$) *UNIV* $?S_d$›)
‹*bij-betw* ($\lambda B$. $_{nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}$) *UNIV*
{$B$. *length-1*$_{nd}$ $B \wedge (\forall\sigma s$ $a$. $\tau$ $B$ $\sigma s$ $a = \tau$ $B$ [*hd* $\sigma s$] $a) \wedge (\forall\sigma s$. $\omega$ $B$ [*hd* $\sigma s$] $=$ $\omega$ $B$ $\sigma s)$}›
⟨*proof*⟩

**lemma** *bij-betw-from-list-to-singl* :
‹*bij-betw* ($\lambda A$. $_d\langle\!\langle A\rangle\!\rangle_{list\rightsquigarrow singl}$)
{$A$. *length-1*$_d$ $A \wedge (\forall\sigma s$ $a$. $\tau$ $A$ [*hd* $\sigma s$] $a = \tau$ $A$ $\sigma s$ $a) \wedge (\forall\sigma s$. $\omega$ $A$ [*hd* $\sigma s$] $=$ $\omega$ $A$ $\sigma s)$} *UNIV*›
‹*bij-betw* ($\lambda B$. $_{nd}\langle\!\langle B\rangle\!\rangle_{list\rightsquigarrow singl}$)
{$B$. *length-1*$_{nd}$ $B \wedge (\forall\sigma s$ $a$. $\tau$ $B$ $\sigma s$ $a = \tau$ $B$ [*hd* $\sigma s$] $a) \wedge (\forall\sigma s$. $\omega$ $B$ [*hd* $\sigma s$] $=$ $\omega$ $B$ $\sigma s)$} *UNIV*›
⟨*proof*⟩

### 3.2.3 Reachability results (for $\mathcal{R}_d$ and $\mathcal{R}_{nd}$)

**lemma** *$\mathcal{R}$-base-trans*[*simp*]: ‹$\mathcal{R}_d$ $(\!|\tau = \lambda\sigma$ $a$. $\Diamond$, $\omega = \lambda\sigma$. $\Diamond$, ... $=$ *some*$|\!)$ $= (\lambda\sigma$. $\{\sigma\})$›

‹$\mathcal{R}_{nd}$ (|$\tau = \lambda\sigma\ a.\ \{\}$, $\omega = \lambda\sigma.\ \{\}$, ... = *some*|) = ($\lambda\sigma.\ \{\sigma\}$)›
⟨*proof*⟩

**theorem** $\mathcal{R}_{nd}$*-from-det-to-ndet* : ‹$\mathcal{R}_{nd}$ ⟪$A$⟫$_{d\hookrightarrow nd}$ $\sigma = \mathcal{R}_d\ A\ \sigma$›
⟨*proof*⟩

**lemma** *bij-betw-*$\mathcal{R}_{nd}$*-if-same-*$\tau$ : ‹*bij-betw* $f$ ($\mathcal{R}_{nd}$ $B_0$ $\sigma_0$) ($\mathcal{R}_{nd}$ $B_1$ ($f$ $\sigma_0$))›
**if** ‹*inj-on* $f$ ($\mathcal{R}_{nd}$ $B_0$ $\sigma_0$)› **and** ‹$\bigwedge\sigma_0{}'$ $a$. $\sigma_0{}' \in \mathcal{R}_{nd}$ $B_0$ $\sigma_0 \Longrightarrow \tau$ $B_1$ ($f$ $\sigma_0{}'$) $a$ = $f$ ‘ $\tau$ $B_0$ $\sigma_0{}'$ $a$›
⟨*proof*⟩

**lemma** *bij-betw-*$\mathcal{R}_d$*-if-same-*$\tau$: ‹*bij-betw* $f$ ($\mathcal{R}_d$ $A_0$ $\sigma_0$) ($\mathcal{R}_d$ $A_1$ ($f$ $\sigma_0$))›
**if** ‹*inj-on* $f$ ($\mathcal{R}_d$ $A_0$ $\sigma_0$)› **and** ‹$\bigwedge\sigma_0{}'$ $a$. $\sigma_0{}' \in \mathcal{R}_d$ $A_0$ $\sigma_0 \Longrightarrow \tau$ $A_1$ ($f$ $\sigma_0{}'$) $a$ = *map-option* $f$ ($\tau$ $A_0$ $\sigma_0{}'$ $a$)›
⟨*proof*⟩

**lemmas** *same-*$\tau$*-implies-same-*$\mathcal{R}_{nd}$ = *bij-betw-*$\mathcal{R}_{nd}$*-if-same-*$\tau$[**where** $f$ = *id*, *simplified bij-betw-def*, *simplified*]
**and** *same-*$\tau$*-implies-same-*$\mathcal{R}_d$ = *bij-betw-*$\mathcal{R}_d$*-if-same-*$\tau$[**where** $f$ = *id*, *simplified bij-betw-def option.map-id*, *simplified*]

**corollary** $\mathcal{R}_d$*-*$\omega$*-useless* [*simp*] : ‹$\mathcal{R}_d$ ($A$(|$\omega$ := *some-*$\omega$|)) $\sigma = \mathcal{R}_d\ A\ \sigma$›
⟨*proof*⟩

**corollary** $\mathcal{R}_{nd}$*-*$\omega$*-useless* [*simp*] : ‹$\mathcal{R}_{nd}$ ($A$(|$\omega$ := *some-*$\omega$|)) $\sigma = \mathcal{R}_{nd}\ A\ \sigma$›
⟨*proof*⟩

**corollary** *indep-enabl-*$\omega$*-useless* [*simp*] :
‹*indep-enabl* ($A_0$(|$\omega$ := *some-*$\omega$|)) $\sigma_0$ $E$ $A_1$ $\sigma_1 \longleftrightarrow$ *indep-enabl* $A_0$ $\sigma_0$ $E$ $A_1$ $\sigma_1$›
‹*indep-enabl* $A_0$ $\sigma_0$ $E$ ($A_1$(|$\omega$ := *some-*$\omega$|)) $\sigma_1 \longleftrightarrow$ *indep-enabl* $A_0$ $\sigma_0$ $E$ $A_1$ $\sigma_1$›
⟨*proof*⟩

**method** $\mathcal{R}$*-subset-method* **uses** *defs opt induct init simps* =
*induct rule*: *induct*, *auto simp add*: *init defs* $\varepsilon$*-simps split*: *if-splits*,
(*metis* (*no-types*, *opaque-lifting*) *simps*)+

**method** $\mathcal{R}_d$*-subset-method* **uses** *defs opt* =
$\mathcal{R}$*-subset-method defs*: *defs opt*: *opt induct*: $\mathcal{R}_d$*.induct init*: $\mathcal{R}_d$*.init simps*: $\mathcal{R}_d$*.simps*

**method** $\mathcal{R}_{nd}$*-subset-method* **uses** *defs opt* =
$\mathcal{R}$*-subset-method defs*: *defs opt*: *opt induct*: $\mathcal{R}_{nd}$*.induct init*: $\mathcal{R}_{nd}$*.init simps*: $\mathcal{R}_{nd}$*.simps*

**lemma** $\mathcal{R}_{nd}$*-from-*$\sigma$*-to-*$\sigma$*s-description*: ‹$\mathcal{R}_{nd}$ $_{nd}$⟪$B$⟫$_{\sigma\hookrightarrow\sigma s}$ [$\sigma$] = {[$\sigma'$]| $\sigma'$. $\sigma' \in \mathcal{R}_{nd}$

$B\ \sigma\}$›
⟨*proof*⟩

**lemma** $\mathcal{R}_d$*-from-*$\sigma$*-to-*$\sigma s$*-description*: ‹$\mathcal{R}_{d\ d}\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ [\sigma] = \{[\sigma']|\ \sigma'.\ \sigma' \in \mathcal{R}_d\ A\ \sigma\}$›
⟨*proof*⟩

**lemma** $\mathcal{R}_{nd}$*-from-singl-to-list-description*: ‹$\mathcal{R}_{nd\ nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}\ [\sigma] = \{[\sigma']|\ \sigma'.\ \sigma' \in \mathcal{R}_{nd}\ B\ \sigma\}$›
⟨*proof*⟩

**lemma** $\mathcal{R}_d$*-from-singl-to-list-description*: ‹$\mathcal{R}_{d\ d}\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}\ [\sigma] = \{[\sigma']|\ \sigma'.\ \sigma' \in \mathcal{R}_d\ A\ \sigma\}$›
⟨*proof*⟩

**lemma** *length-*$\mathcal{R}_d$*-from-*$\sigma$*-to-*$\sigma s$:
‹$\sigma s' \in \mathcal{R}_{d\ d}\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ \sigma s \Longrightarrow \sigma s' = \sigma s \vee length\ \sigma s' = 1$›
⟨*proof*⟩

**lemma** *length-*$\mathcal{R}_{nd}$*-from-*$\sigma$*-to-*$\sigma s$:
‹$\sigma s' \in \mathcal{R}_{nd\ nd}\langle\!\langle B\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ \sigma s \Longrightarrow \sigma s' = \sigma s \vee length\ \sigma s' = 1$›
⟨*proof*⟩

**lemma** *length-*$\mathcal{R}_d$*-from-singl-to-list*:
‹$\sigma s' \in \mathcal{R}_{d\ d}\langle\!\langle A\rangle\!\rangle_{singl\hookrightarrow list}\ \sigma s \Longrightarrow \sigma s' = \sigma s \vee length\ \sigma s' = 1$›
⟨*proof*⟩

**lemma** *length-*$\mathcal{R}_{nd}$*-from-singl-to-list*:
‹$\sigma s' \in \mathcal{R}_{nd\ nd}\langle\!\langle B\rangle\!\rangle_{singl\hookrightarrow list}\ \sigma s \Longrightarrow \sigma s' = \sigma s \vee length\ \sigma s' = 1$›
⟨*proof*⟩

## 3.3 Normalization

### 3.3.1 Non-deterministic Case

First version, without final state notion

**abbreviation** *P-nd-step* :: ‹$[('\sigma,\ 'a)\ enabl,\ ('\sigma,\ 'a)\ trans_{nd},\ '\sigma \Rightarrow ('a,\ 'r)\ process_{ptick},\ '\sigma] \Rightarrow ('a,\ 'r)\ process_{ptick}$›
**where** ‹*P-nd-step* $\varepsilon_A\ \tau_A\ X\ \sigma \equiv \Box\ e \in \varepsilon_A\ \sigma \rightarrow \sqcap\ \sigma' \in \tau_A\ \sigma\ e.\ X\ \sigma'$›

**definition** *P-nd* :: ‹$('\sigma,\ 'a,\ 'r,\ '\alpha)\ A_{nd}$*-scheme* $\Rightarrow '\sigma \Rightarrow ('a,\ 'r)\ process_{ptick}$› (‹$P\langle\!\langle$-$\rangle\!\rangle_{nd}$› *1000*)
**where** ‹$P\langle\!\langle A\rangle\!\rangle_{nd} \equiv \upsilon\ X.$ *P-nd-step* $(\varepsilon\ A)\ (\tau\ A)\ X$›

**lemma** *P-nd-step-constructive* [*simp*] : ‹*constructive* (*P-nd-step* $\varepsilon_A$ $\tau_A$)› ⟨*proof*⟩

**lemma** *P-nd-step-cont* [*simp*] : ‹$\forall \sigma$ *a*. *finite* ($\tau_A$ $\sigma$ *a*) $\Longrightarrow$ *cont* (*P-nd-step* $\varepsilon_A$ $\tau_A$)›
⟨*proof*⟩

**lemma** *P-nd-step-constructive-bis* : ‹*constructive* (*P-nd-step* ($\varepsilon$ *A*) ($\tau$ *A*))› ⟨*proof*⟩

**lemma** *P-nd-step-cont-bis* [*simp*] : ‹*finite-trans A* $\Longrightarrow$ *cont* (*P-nd-step* ($\varepsilon$ *A*) ($\tau$ *A*))›
⟨*proof*⟩

**lemma** *P-nd-rec*: ‹$P\langle\!\langle A\rangle\!\rangle_{nd}$ = ($\lambda\sigma$. *P-nd-step* ($\varepsilon$ *A*) ($\tau$ *A*) $P\langle\!\langle A\rangle\!\rangle_{nd}$ $\sigma$)›
⟨*proof*⟩

**lemma** *P-nd-is-fix* : ‹*finite-trans A* $\Longrightarrow$ $P\langle\!\langle A\rangle\!\rangle_{nd}$ = ($\mu$ *X*. *P-nd-step* ($\varepsilon$ *A*) ($\tau$ *A*) *X*)›
⟨*proof*⟩

**lemma** *non-destructive-imp-restriction-cont* [*simp*] :
‹*non-destructive f* $\Longrightarrow$ *restriction-cont f*›
⟨*proof*⟩

**lemma** *P-nd-ω-useless*: ‹$P\langle\!\langle A\rangle\!\rangle_{nd}$ = $P\langle\!\langle A(\!|\omega$ := *some-ω*$|\!)\rangle\!\rangle_{nd}$›
⟨*proof*⟩

**lemma** *P-nd-ω-useless-bis* : ‹$P\langle\!\langle A\rangle\!\rangle_{nd}$ = $P\langle\!\langle A(\!|\omega := \lambda\sigma.\ \{\}|\!)\rangle\!\rangle_{nd}$›
⟨*proof*⟩

**lemma** *P-nd-induct* [*case-names adm base step*] :
‹$adm_{\downarrow}$ *P* $\Longrightarrow$ *P* $\sigma$ $\Longrightarrow$ ($\bigwedge$*X*. *P X* $\Longrightarrow$ *P* (*P-nd-step* ($\varepsilon$ *A*) ($\tau$ *A*) *X*)) $\Longrightarrow$ *P* $P\langle\!\langle A\rangle\!\rangle_{nd}$›
⟨*proof*⟩

**lemma** *P-nd-induct-iterated* [*consumes 1*, *case-names adm base step*] :
‹⟦*0* < *k*; $adm_{\downarrow}$ *P*; *P* $\sigma$; $\bigwedge$*X*. *P X* $\Longrightarrow$ *P* ((*P-nd-step* ($\varepsilon$ *A*) ($\tau$ *A*) ^^ *k*) *X*)⟧ $\Longrightarrow$ *P* $P\langle\!\langle A\rangle\!\rangle_{nd}$›
⟨*proof*⟩

New version with final state notion where we just have *SKIPS*.

**abbreviation** $P_{SKIPS}$*-nd-step* ::
‹[('$\sigma$, '*a*) *enabl*, ('$\sigma$, '*a*) $trans_{nd}$, '$\sigma$ ⇒ '*r set*, '$\sigma$ ⇒ ('*a*, '*r*) $process_{ptick}$, '$\sigma$] ⇒ ('*a*, '*r*) $process_{ptick}$›
**where** ‹$P_{SKIPS}$*-nd-step* $\varepsilon_A$ $\tau_A$ $\omega_A$ *X* $\sigma$ ≡ *if* $\omega_A$ $\sigma$ = {} *then P-nd-step* $\varepsilon_A$ $\tau_A$ *X* $\sigma$ *else SKIPS* ($\omega_A$ $\sigma$)›

**definition** $P_{SKIPS}$-*nd* :: ‹$('\sigma, 'a, 'r, '\alpha)$ $A_{nd}$-*scheme* $\Rightarrow$ $'\sigma \Rightarrow ('a, 'r)$ *process*$_{ptick}$› (‹$P_{SKIPS}\langle\!\langle$-$\rangle\!\rangle_{nd}$› *1000*)
**where** ‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd} \equiv \upsilon\ X.\ P_{SKIPS}$-*nd-step* $(\varepsilon\ A)\ (\tau\ A)\ (\omega\ A)\ X$›

**lemma** $P_{SKIPS}$-*nd-step-constructive* [*simp*] : ‹*constructive* ($P_{SKIPS}$-*nd-step* $\varepsilon_A$ $\tau_A$ $\omega_A$)› ⟨*proof*⟩

**lemma** $P_{SKIPS}$-*nd-step-cont* [*simp*] : ‹$\forall \sigma\ a.$ *finite* $(\tau_A\ \sigma\ a) \Longrightarrow$ *cont* ($P_{SKIPS}$-*nd-step* $\varepsilon_A$ $\tau_A$ $\omega_A$)›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*nd-step-constructive-bis* : ‹*constructive* ($P_{SKIPS}$-*nd-step* $(\varepsilon\ A)$ $(\tau\ A)$ $(\omega\ A)$)› ⟨*proof*⟩

**lemma** $P_{SKIPS}$-*nd-step-cont-bis* [*simp*] : ‹*finite-trans* $A \Longrightarrow$ *cont* ($P_{SKIPS}$-*nd-step* $(\varepsilon\ A)$ $(\tau\ A)$ $(\omega\ A)$)›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*nd-rec*: ‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd} = (\lambda\sigma.\ P_{SKIPS}$-*nd-step* $(\varepsilon\ A)$ $(\tau\ A)$ $(\omega\ A)$ $P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma)$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*nd-is-fix* : ‹*finite-trans* $A \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd} = (\mu\ X.\ P_{SKIPS}$-*nd-step* $(\varepsilon\ A)$ $(\tau\ A)$ $(\omega\ A)$ $X)$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*nd-induct* [*case-names adm base step*] :
‹$adm_{\downarrow}\ P \Longrightarrow P\ \sigma \Longrightarrow (\bigwedge X.\ P\ X \Longrightarrow P$ ($P_{SKIPS}$-*nd-step* $(\varepsilon\ A)$ $(\tau\ A)$ $(\omega\ A)$ $X)) \Longrightarrow P\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*nd-induct-iterated* [*consumes 1*, *case-names adm base step*] :
‹$[\![0 < k;\ adm_{\downarrow}\ P;\ P\ \sigma;\ \bigwedge X.\ P\ X \Longrightarrow P$ (($P_{SKIPS}$-*nd-step* $(\varepsilon\ A)$ $(\tau\ A)$ $(\omega\ A)$ $^{\frown\frown}k$) $X)]\!] \Longrightarrow P\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}$›
⟨*proof*⟩

Correspondence when we always have $\omega\ A\ \sigma = \{\}$.

**lemma** $P_{SKIPS}$-*nd-empty-*$\varrho$ : ‹$\varrho\ A = \{\} \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd} = P\langle\!\langle A\rangle\!\rangle_{nd}$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*nd-updated-*$\omega$: ‹$P\langle\!\langle A\rangle\!\rangle_{nd} = P_{SKIPS}\langle\!\langle A(\!|\omega := \lambda\sigma.\ \{\}|\!)\rangle\!\rangle_{nd}$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*nd-empty-*$\varrho$*-inter-*$\mathcal{R}_{nd}$:
‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma = P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma$› **if** ‹$\varrho\ A \cap \mathcal{R}_{nd}\ A\ \sigma = \{\}$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-nd-rec-notin-*$\varrho$:
‹$\sigma \notin \varrho\ A \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma =$ *P-nd-step* $(\varepsilon\ A)\ (\tau\ A)\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-nd-rec-in-*$\varrho$: ‹$\sigma \in \varrho\ A \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma =$ *SKIPS* $(\omega\ A\ \sigma)$›
⟨*proof*⟩

### 3.3.2 Deterministic Case

First version, without final state notion.

**abbreviation** *P-d-step* :: ‹[$('\sigma,\ 'a)$ *enabl*, $('\sigma,\ 'a)$ $trans_d$, $'\sigma \Rightarrow ('a,\ 'r)$ $process_{ptick}$, $'\sigma] \Rightarrow ('a,\ 'r)$ $process_{ptick}$›
**where** ‹*P-d-step* $\varepsilon_A\ \tau_A\ X\ s \equiv \Box\ e \in \varepsilon_A\ s \rightarrow X\ \lceil\tau_A\ s\ e\rceil$›

**definition** *P-d* :: ‹$('\sigma,\ 'a,\ 'r,\ '\alpha)$ $A_d$*-scheme* $\Rightarrow '\sigma \Rightarrow ('a,\ 'r)$ $process_{ptick}$› (‹$P\langle\!\langle$-$\rangle\!\rangle_d$› *1000*)
**where** ‹$P\langle\!\langle A\rangle\!\rangle_d \equiv \upsilon\ X.$ *P-d-step* $(\varepsilon\ A)\ (\tau\ A)\ X$›

**lemma** *P-d-step-constructive*[*simp*] : ‹*constructive* (*P-d-step* $\varepsilon_A\ \tau_A$)› ⟨*proof*⟩

**lemmas** *P-d-step-constructive-bis* = *P-d-step-constructive*[*of* ‹$\varepsilon\ A$› ‹$\tau\ A$›] **for** *A*

**lemma** *P-d-step-cont*[*simp*]: ‹*cont* (*P-d-step* $\varepsilon_A\ \tau_A$)›
⟨*proof*⟩

**lemmas** *P-d-step-cont-bis* = *P-d-step-cont*[*of* ‹$\varepsilon\ A$› ‹$\tau\ A$›] **for** *A*

**lemma** *P-d-rec*: ‹$P\langle\!\langle A\rangle\!\rangle_d = (\lambda s.$ *P-d-step* $(\varepsilon\ A)\ (\tau\ A)\ P\langle\!\langle A\rangle\!\rangle_d\ s)$›
⟨*proof*⟩

**lemma** *P-d-is-fix* : ‹$P\langle\!\langle A\rangle\!\rangle_d = (\mu\ X.$ *P-d-step* $(\varepsilon\ A)\ (\tau\ A)\ X)$›
⟨*proof*⟩

**lemma** *P-d-*$\omega$*-useless*: ‹$P\langle\!\langle A\rangle\!\rangle_d = P\langle\!\langle A(\!|\omega :=$ *some-*$\omega|\!)\rangle\!\rangle_d$›
⟨*proof*⟩

**lemma** *P-d-*$\omega$*-useless-bis*: ‹$P\langle\!\langle A\rangle\!\rangle_d = P\langle\!\langle A(\!|\omega := \lambda\sigma.\ \Diamond|\!)\rangle\!\rangle_d$›
⟨*proof*⟩

**lemma** *P-d-induct* [*case-names adm base step*] :
‹$[\![adm_{\downarrow}\ P;\ P\ \sigma;\ \bigwedge X.\ P\ X \Longrightarrow P$ (*P-d-step* $(\varepsilon\ A)\ (\tau\ A)\ X)]\!] \Longrightarrow P\ P\langle\!\langle A\rangle\!\rangle_d$›
⟨*proof*⟩

**lemma** *P-d-induct-iterated* [*consumes 1*, *case-names adm base step*] :

‹$[\![0 < k;\ adm_{\downarrow}\ P;\ P\ \sigma;\ \bigwedge X.\ P\ X \Longrightarrow P\ ((P\text{-}d\text{-}step\ (\varepsilon\ A)\ (\tau\ A)\ \hat{}\hat{}\ k)\ X)]\!] \Longrightarrow P\ P\langle\!\langle A\rangle\!\rangle_d$›
⟨*proof*⟩

New version with final state notion where we just *SKIP*.

**abbreviation** $P_{SKIPS}$*-d-step* ::
‹$[('\sigma,\ 'a)\ enabl,\ ('\sigma,\ 'a)\ trans_d,\ '\sigma \Rightarrow 'r\ option,\ '\sigma \Rightarrow ('a,\ 'r)\ process_{ptick},\ '\sigma] \Rightarrow ('a,\ 'r)\ process_{ptick}$›
**where** ‹$P_{SKIPS}$*-d-step* $\varepsilon_A\ \tau_A\ \omega_A\ X\ \sigma \equiv case\ \omega_A\ \sigma\ of\ \lfloor r\rfloor \Rightarrow SKIP\ r \mid \Diamond \Rightarrow P\text{-}d\text{-}step\ \varepsilon_A\ \tau_A\ X\ \sigma$›

**definition** $P_{SKIPS}$*-d* :: ‹$('\sigma,\ 'a,\ 'r,\ '\alpha)\ A_d\text{-}scheme \Rightarrow '\sigma \Rightarrow ('a,\ 'r)\ process_{ptick}$› (‹$P_{SKIPS}\langle\!\langle\text{-}\rangle\!\rangle_d$› *1000*)
**where** ‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d \equiv \upsilon\ X.\ P_{SKIPS}\text{-}d\text{-}step\ (\varepsilon\ A)\ (\tau\ A)\ (\omega\ A)\ X$›

**lemma** $P_{SKIPS}$*-d-step-constructive*[*simp*]: ‹$constructive\ (P_{SKIPS}\text{-}d\text{-}step\ \varepsilon_A\ \tau_A\ \mathcal{S}_{FA})$›
⟨*proof*⟩

**lemmas** $P_{SKIPS}$*-d-step-constructive-bis* = $P_{SKIPS}$*-d-step-constructive*[*of* ‹$\varepsilon\ A$› ‹$\tau\ A$› ‹$\omega\ A$›] **for** *A*

**lemma** $P_{SKIPS}$*-d-step-cont*[*simp*]: ‹$cont\ (P_{SKIPS}\text{-}d\text{-}step\ \varepsilon_A\ \tau_A\ \mathcal{S}_{FA})$›
⟨*proof*⟩

**lemmas** $P_{SKIPS}$*-d-step-cont-bis* = $P_{SKIPS}$*-d-step-cont*[*of* ‹$\varepsilon\ A$› ‹$\tau\ A$› ‹$\omega\ A$›] **for** *A*

**lemma** $P_{SKIPS}$*-d-rec*: ‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d = (\lambda\sigma.\ P_{SKIPS}\text{-}d\text{-}step\ (\varepsilon\ A)\ (\tau\ A)\ (\omega\ A)\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma)$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-d-is-fix* : ‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d = (\mu\ X.\ P_{SKIPS}\text{-}d\text{-}step\ (\varepsilon\ A)\ (\tau\ A)\ (\omega\ A)\ X)$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-d-induct* [*case-names adm base step*] :
‹$adm_{\downarrow}\ P \Longrightarrow P\ \sigma \Longrightarrow (\bigwedge X.\ P\ X \Longrightarrow P\ (P_{SKIPS}\text{-}d\text{-}step\ (\varepsilon\ A)\ (\tau\ A)\ (\omega\ A)\ X)) \Longrightarrow P\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-d-induct-iterated* [*consumes 1*, *case-names adm base step*] :
‹$[\![0 < k;\ adm_{\downarrow}\ P;\ P\ \sigma;\ \bigwedge X.\ P\ X \Longrightarrow P\ ((P_{SKIPS}\text{-}d\text{-}step\ (\varepsilon\ A)\ (\tau\ A)\ (\omega\ A)\ \hat{}\hat{}\ k)\ X)]\!] \Longrightarrow P\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d$›
⟨*proof*⟩

Correspondence when we always have $\omega\ A\ \sigma = \{\}$.

**lemma** $P_{SKIPS}$*-d-empty-*$\varrho$ : ‹$\varrho\ A = \{\} \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d = P\langle\!\langle A\rangle\!\rangle_d$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-d-updated-*$\omega$: ‹$P\langle\!\langle A\rangle\!\rangle_d = P_{SKIPS}\langle\!\langle A(\!|\omega := \lambda\sigma.\ \Diamond|\!)\rangle\!\rangle_d$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-d-empty-*$\varrho$*-inter-*$\mathcal{R}_d$:
‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma = P\langle\!\langle A\rangle\!\rangle_d\ \sigma$› **if** ‹$\varrho\ A \cap \mathcal{R}_d\ A\ \sigma = \{\}$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-d-rec-notin-*$\varrho$:
‹$\sigma \notin \varrho\ A \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma = P\text{-}d\text{-}step\ (\varepsilon\ A)\ (\tau\ A)\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-d-rec-in-*$\varrho$: ‹$\sigma \in \varrho\ A \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma = SKIP\ \lceil\omega\ A\ \sigma\rceil$›
⟨*proof*⟩

### 3.3.3 Link between deterministic and non-deterministic ProcOmata

**lemma** $P_{SKIPS}$*-nd-from-det-to-ndet-is-*$P_{SKIPS}$*-d* : ‹$P_{SKIPS}\langle\!\langle\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle_{nd} = P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d$›
⟨*proof*⟩

**corollary** *P-nd-from-det-to-ndet-is-P-d* : ‹$P\langle\!\langle\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle_{nd} = P\langle\!\langle A\rangle\!\rangle_d$›
⟨*proof*⟩

### 3.3.4 Prove Equality between ProcOmata

**This is the easiest method we can think about.**

**lemma** *P-d-eqI* : ‹$(\bigwedge\sigma\ a.\ \tau\ A\ \sigma\ a = \tau\ B\ \sigma\ a) \Longrightarrow P\langle\!\langle A\rangle\!\rangle_d = P\langle\!\langle B\rangle\!\rangle_d$›
⟨*proof*⟩

**lemma** *P-nd-eqI* : ‹$(\bigwedge\sigma\ a.\ \tau\ A\ \sigma\ a = \tau\ B\ \sigma\ a) \Longrightarrow P\langle\!\langle A\rangle\!\rangle_{nd} = P\langle\!\langle B\rangle\!\rangle_{nd}$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-d-eqI* :
‹$(\bigwedge\sigma\ a.\ \sigma \notin \varrho\ A \Longrightarrow \tau\ A\ \sigma\ a = \tau\ B\ \sigma\ a) \Longrightarrow (\bigwedge\sigma.\ \omega\ A\ \sigma = \omega\ B\ \sigma) \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d = P_{SKIPS}\langle\!\langle B\rangle\!\rangle_d$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-nd-eqI* :
‹$(\bigwedge\sigma\ a.\ \sigma \notin \varrho\ A \Longrightarrow \tau\ A\ \sigma\ a = \tau\ B\ \sigma\ a) \Longrightarrow (\bigwedge\sigma.\ \omega\ A\ \sigma = \omega\ B\ \sigma) \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd} = P_{SKIPS}\langle\!\langle B\rangle\!\rangle_{nd}$›
⟨*proof*⟩

**We establish now a much more powerful theorem.**

**theorem** $P_{SKIPS}$*-nd-eqI-strong*:

**assumes** *inj-on-f* : ‹*inj-on f* $(\mathcal{R}_{nd}\ A_0\ \sigma_0)$›
**and** *eq-trans* : ‹$\bigwedge\sigma_0{}'\ a.\ \sigma_0{}' \in \mathcal{R}_{nd}\ A_0\ \sigma_0 \Longrightarrow \tau\ A_1\ (f\ \sigma_0{}')\ a = f\ `\ (\tau\ A_0\ \sigma_0{}'\ a)$›
**and** *eq-fin* : ‹$\bigwedge\sigma_0{}'.\ \sigma_0{}' \in \mathcal{R}_{nd}\ A_0\ \sigma_0 \Longrightarrow \omega\ A_1\ (f\ \sigma_0{}') = \omega\ A_0\ \sigma_0{}'$›
**shows** ‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0 = P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_{nd}\ (f\ \sigma_0)$›
⟨*proof*⟩

**theorem** *P-nd-eqI-strong*:
‹$[\![$*inj-on f* $(\mathcal{R}_{nd}\ A_0\ \sigma_0)$;
$\bigwedge\sigma_0{}'\ a.\ \sigma_0{}' \in \mathcal{R}_{nd}\ A_0\ \sigma_0 \Longrightarrow \tau\ A_1\ (f\ \sigma_0{}')\ a = f\ `\ (\tau\ A_0\ \sigma_0{}'\ a)]\!]$
$\Longrightarrow P\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0 = P\langle\!\langle A_1\rangle\!\rangle_{nd}\ (f\ \sigma_0)$›
⟨*proof*⟩

**theorem** $P_{SKIPS}$*-d-eqI-strong*:
**assumes** ‹*inj-on f* $(\mathcal{R}_d\ A_0\ \sigma_0)$›
**and** ‹$\bigwedge\sigma_0{}'\ a.\ \sigma_0{}' \in \mathcal{R}_d\ A_0\ \sigma_0 \Longrightarrow \tau\ A_1\ (f\ \sigma_0{}')\ a =$ *map-option f* $(\tau\ A_0\ \sigma_0{}'\ a)$›
**and** ‹$\bigwedge\sigma_0{}'.\ \sigma_0{}' \in \mathcal{R}_d\ A_0\ \sigma_0 \Longrightarrow \omega\ A_1\ (f\ \sigma_0{}') = \omega\ A_0\ \sigma_0{}'$›
**shows** ‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0 = P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_d\ (f\ \sigma_0)$›
⟨*proof*⟩

**theorem** *P-d-eqI-strong*:
‹$[\![$*inj-on f* $(\mathcal{R}_d\ A_0\ \sigma_0)$;
$\bigwedge\sigma_0{}'\ a.\ \sigma_0{}' \in \mathcal{R}_d\ A_0\ \sigma_0 \Longrightarrow \tau\ A_1\ (f\ \sigma_0{}')\ a =$ *map-option f* $(\tau\ A_0\ \sigma_0{}'\ a)]\!]$
$\Longrightarrow P\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0 = P\langle\!\langle A_1\rangle\!\rangle_d\ (f\ \sigma_0)$›
⟨*proof*⟩

**lemmas** $P_{SKIPS}$*-nd-eqI-strong-id* = $P_{SKIPS}$*-nd-eqI-strong*[*of id*, *simplified*]
**and** $P_{SKIPS}$*-d-eqI-strong-id* = $P_{SKIPS}$*-d-eqI-strong*
[*of id*, *simplified id-def option.map-ident*, *simplified*]
**and** *P-nd-eqI-strong-id* = *P-nd-eqI-strong*[*of id*, *simplified*]
**and** *P-d-eqI-strong-id* = *P-d-eqI-strong*
[*of id*, *simplified id-def option.map-ident*, *simplified*]

**corollary** $P_{SKIPS}$*-nd-from-σ-to-σs-is-*$P_{SKIPS}$*-nd* : ‹$P_{SKIPS}\langle\!\langle_{nd}\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\rangle\!\rangle_{nd}\ [\sigma] = P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$*-d-from-σ-to-σs-is-*$P_{SKIPS}$*-d* : ‹$P_{SKIPS}\langle\!\langle_d\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\rangle\!\rangle_d\ [\sigma] = P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma$›
⟨*proof*⟩

**corollary** *P-nd-from-σ-to-σs-is-P-nd* : ‹$P\langle\!\langle_{nd}\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\rangle\!\rangle_{nd}\ [\sigma] = P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma$›
⟨*proof*⟩

**corollary** *P-d-from-σ-to-σs-is-P-d* : ‹$P\langle\!\langle_{d}\langle\!\langle A\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\rangle\!\rangle_{d}\ [\sigma] = P\langle\!\langle A\rangle\!\rangle_{d}\ \sigma$›
⟨*proof*⟩

Behaviour of normalizations. We will use the following methods in combining theories.

**fun** *recursive-modifier-fun*$_d$ :: ‹$[('\sigma \times 'a) \Rightarrow '\sigma\ option,\ ((('\sigma \times 'a) \times '\sigma\ option)\ list] \Rightarrow ('\sigma \times 'a) \Rightarrow '\sigma\ option$›
**where** ‹*recursive-modifier-fun*$_d$ $f\ [] = f$›
| ‹*recursive-modifier-fun*$_d$ $f\ (((s,\ e),\ t)\ \#\ \mathcal{G}_A) =$ *recursive-modifier-fun*$_d$ $(f((s,\ e) := t))\ \mathcal{G}_A$›

**abbreviation** *recursive-constructor-A*$_d$ :: ‹$[((('\sigma \times 'a) \times '\sigma\ option)\ list,\ '\sigma \Rightarrow 'r\ option] \Rightarrow ('\sigma,\ 'a,\ 'r)\ A_d$›
**where** ‹*recursive-constructor-A*$_d$ $\mathcal{G}_A\ \omega_A \equiv (\!|\tau =$ *curry* (*recursive-modifier-fun*$_d$ $(\lambda(s,\ e).\ \Diamond)\ \mathcal{G}_A),\ \omega = \omega_A|\!)$›

**lemma** *ε-det-breaker*:
‹$\varepsilon\ ((\!|\tau =$ *curry* $(g((\sigma'::'\sigma,\ a) \mapsto \sigma''::'\sigma)),\ \omega =$ *some-ω*, ... = *some-more*$|\!)\ )\ \sigma =$
(*if* $\sigma = \sigma'$ *then* $\{a\} \cup \varepsilon\ (\!|\tau =$ *curry* $g,\ \omega =$ *some-ω*$|\!)\ \sigma'$ *else* $\varepsilon\ (\!|\tau =$ *curry* $g,\ \omega =$ *some-ω*$|\!)\ \sigma)$›
⟨*proof*⟩

**method** *ε-det-calc* = (*unfold recursive-modifier-fun*$_d$*.simps ε-det-breaker*, *simp cong*: *if-cong*)[*1*]

**method** *τ-det-calc* = (*unfold recursive-modifier-fun*$_d$*.simps*, *simp cong*: *if-cong*)[*1*]

**lemma** *bij-Renaming-P*$_{SKIPS}$*-nd* :
**fixes** $A$ :: ‹$('\sigma,\ 'a,\ 'r,\ '\alpha)\ A_{nd}$*-scheme*› **and** $f$ :: ‹$'a \Rightarrow 'b$› **and** $g$ :: ‹$'r \Rightarrow 's$›
**assumes** ‹*bij f*›
**defines** *B-def* : ‹$B \equiv (\!|\tau = \lambda\sigma\ b.\ \tau\ A\ \sigma\ (inv\ f\ b),\ \omega = \lambda\sigma.\ g\ `\ (\omega\ A\ \sigma)|\!)$›
**shows** ‹*Renaming* $(P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma)\ f\ g = P_{SKIPS}\langle\!\langle B\rangle\!\rangle_{nd}\ \sigma$› (**is** ‹*?lhs* $\sigma$ = -›)
⟨*proof*⟩

**lemma** *bij-Renaming-P*$_{SKIPS}$*-d* :
‹*bij f* $\Longrightarrow$ *Renaming* $(P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{d}\ \sigma)\ f\ g =$
$P_{SKIPS}\langle\!\langle(\!|\tau = \lambda\sigma\ b.\ \tau\ A\ \sigma\ (inv\ f\ b),\ \omega = \lambda\sigma.$ *map-option* $g\ (\omega\ A\ \sigma)|\!)\rangle\!\rangle_{d}\ \sigma$›

⟨*proof*⟩

**lemma** *RenamingTick-*$P_{SKIPS}$*-nd* :
‹*RenamingTick* $(P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma)\ g = P_{SKIPS}\langle\!\langle(\!|\tau = \tau\ A,\ \omega = \lambda\sigma.\ g\ `\ \omega\ A\ \sigma|\!)\rangle\!\rangle_{nd}\ \sigma$›
⟨*proof*⟩

**lemma** *RenamingTick-*$P_{SKIPS}$*-d* :
‹*RenamingTick* $(P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{d}\ \sigma)\ g = P_{SKIPS}\langle\!\langle(\!|\tau = \tau\ A,\ \omega = \lambda\sigma.\ \textit{map-option}\ g\ (\omega\ A\ \sigma)|\!)\rangle\!\rangle_{d}\ \sigma$›
⟨*proof*⟩

# Chapter 4

# Advanced Properties of ProcOmata

$\langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle$

# Chapter 5

# Combining Automata for Synchronization Product

## 5.1 Definitions

### 5.1.1 General Patterns

**abbreviation** *combine-sets-Sync* :: ‹$'a$ *set* $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a$ *set*›
**where** ‹*combine-sets-Sync* $S_0$ $E$ $S_1$ $\equiv$ $(S_0 - E - S_1) \cup (S_1 - E - S_0) \cup (S_0 \cap S_1 - E) \cup S_0 \cap S_1 \cap E$›

**definition** *combine-Sync-*$\varepsilon$ ::
‹$[('\sigma_0, 'e, '\sigma_0', 'r_0, '\alpha_0)$ *A-scheme*, $'e$ *set*,
$('\sigma_1, 'e, '\sigma_1', 'r_1, '\alpha_1)$ *A-scheme*, $'\sigma \Rightarrow '\sigma_0, '\sigma \Rightarrow '\sigma_1, '\sigma] \Rightarrow 'e$ *set*›
**where** ‹*combine-Sync-*$\varepsilon$ $A_0$ $E$ $A_1$ $i_0$ $i_1$ $\sigma s$ $\equiv$ *combine-sets-Sync* $(\varepsilon\ A_0\ (i_0\ \sigma s))\ E\ (\varepsilon\ A_1\ (i_1\ \sigma s))$›

**lemma** *combine-Sync-*$\varepsilon$*-def-bis* :
‹*combine-Sync-*$\varepsilon$ $A_0$ $E$ $A_1$ $i_0$ $i_1$ $\sigma s$ =
$\varepsilon\ A_0\ (i_0\ \sigma s) \cup \varepsilon\ A_1\ (i_1\ \sigma s) - E \cup \varepsilon\ A_0\ (i_0\ \sigma s) \cap \varepsilon\ A_1\ (i_1\ \sigma s)$›
⟨*proof*⟩

**fun** *combine*$_d$*-Sync-*$\tau$ ::
‹$[('\sigma_0, 'e, 'r_0, '\alpha_0)$ $A_d$*-scheme*, $'e$ *set*, $('\sigma_1, 'e, 'r_1, '\alpha_1)$ $A_d$*-scheme*,
$'\sigma \Rightarrow '\sigma_0, '\sigma \Rightarrow '\sigma_1, '\sigma_0 \Rightarrow '\sigma_1 \Rightarrow '\sigma] \Rightarrow ('\sigma, 'e)$ *trans*$_d$›
**and** *combine*$_{nd}$*-Sync-*$\tau$ ::
‹$[('\sigma_0, 'e, 'r_0, '\alpha_0)$ $A_{nd}$*-scheme*, $'e$ *set*, $('\sigma_1, 'e, 'r_1, '\alpha_1)$ $A_{nd}$*-scheme*,
$'\sigma \Rightarrow '\sigma_0, '\sigma \Rightarrow '\sigma_1, '\sigma_0 \Rightarrow '\sigma_1 \Rightarrow '\sigma] \Rightarrow ('\sigma, 'e)$ *trans*$_{nd}$›
**where** ‹*combine*$_d$*-Sync-*$\tau$ $A_0$ $E$ $A_1$ $i_0$ $i_1$ $f$ $\sigma s$ $e$ =
( *if* $e \in \varepsilon\ A_0\ (i_0\ \sigma s) \cap \varepsilon\ A_1\ (i_1\ \sigma s)$
*then update-both* $A_0$ $A_1$ $(i_0\ \sigma s)$ $(i_1\ \sigma s)$ $e$ (*f-up-opt* $f$)
*else* *if* $e \in \varepsilon\ A_0\ (i_0\ \sigma s) - E - \varepsilon\ A_1\ (i_1\ \sigma s)$
*then update-left* $A_0$ $(i_0\ \sigma s)$ $(i_1\ \sigma s)$ $e$ (*f-up-opt* $f$) $(\lambda s.\ \lfloor s \rfloor)$
*else* *if* $e \in \varepsilon\ A_1\ (i_1\ \sigma s) - E - \varepsilon\ A_0\ (i_0\ \sigma s)$

*then update-right* $A_1$ $(i_0\ \sigma s)$ $(i_1\ \sigma s)$ $e$ (*f-up-opt f*) $(\lambda s.\ \lfloor s \rfloor)$
*else* $\Diamond$)›
| ‹*combine*$_{nd}$*-Sync-*$\tau$ $A_0$ $E$ $A_1$ $i_0$ $i_1$ $f$ $\sigma s$ $e$ =
( *if* $e \in \varepsilon\ A_0\ (i_0\ \sigma s) \cap \varepsilon\ A_1\ (i_1\ \sigma s) \cap E$
*then update-both* $A_0$ $A_1$ $(i_0\ \sigma s)$ $(i_1\ \sigma s)$ $e$ (*f-up-set f*)
*else* *if* $e \in \varepsilon\ A_0\ (i_0\ \sigma s) \cap \varepsilon\ A_1\ (i_1\ \sigma s) - E$
*then update-left* $A_0$ $(i_0\ \sigma s)$ $(i_1\ \sigma s)$ $e$ (*f-up-set f*) $(\lambda s.\ \{s\})$
$\cup$ *update-right* $A_1$ $(i_0\ \sigma s)$ $(i_1\ \sigma s)$ $e$ (*f-up-set f*) $(\lambda s.\ \{s\})$
*else* *if* $e \in \varepsilon\ A_0\ (i_0\ \sigma s) - E - \varepsilon\ A_1\ (i_1\ \sigma s)$
*then update-left* $A_0$ $(i_0\ \sigma s)$ $(i_1\ \sigma s)$ $e$ (*f-up-set f*) $(\lambda s.\ \{s\})$
*else* *if* $e \in \varepsilon\ A_1\ (i_1\ \sigma s) - E - \varepsilon\ A_0\ (i_0\ \sigma s)$
*then update-right* $A_1$ $(i_0\ \sigma s)$ $(i_1\ \sigma s)$ $e$ (*f-up-set f*) $(\lambda s.\ \{s\})$
*else* $\{\}$)›

**fun** *combine*$_d$*-Sync-*$\omega$ ::
‹[$('\sigma_0,\ 'e,\ 'r_0,\ '\alpha_0)$ $A_d$*-scheme*, $'e$ *set*, $('\sigma_1,\ 'e,\ 'r_1,\ '\alpha_1)$ $A_d$*-scheme*,
$'\sigma \Rightarrow '\sigma_0,\ '\sigma \Rightarrow '\sigma_1,\ 'r_0 \Rightarrow 'r_1 \Rightarrow 'r$ *option*] $\Rightarrow$ ($'\sigma \Rightarrow 'r$ *option*)›
**and** *combine*$_{nd}$*-Sync-*$\omega$ ::
‹[$('\sigma_0,\ 'e,\ 'r_0,\ '\alpha_0)$ $A_{nd}$*-scheme*, $'e$ *set*, $('\sigma_1,\ 'e,\ 'r_1,\ '\alpha_1)$ $A_{nd}$*-scheme*,
$'\sigma \Rightarrow '\sigma_0,\ '\sigma \Rightarrow '\sigma_1,\ 'r_0 \Rightarrow 'r_1 \Rightarrow 'r$ *option*] $\Rightarrow$ ($'\sigma \Rightarrow 'r$ *set*)›
**where** ‹*combine*$_d$*-Sync-*$\omega$ $A_0$ $E$ $A_1$ $i_0$ $i_1$ $g$ $\sigma s$ =
(*case* $\omega$ $A_0$ $(i_0\ \sigma s)$
*of* $\Diamond \Rightarrow \Diamond \mid \lfloor r_0 \rfloor \Rightarrow$ (*case* $\omega$ $A_1$ $(i_1\ \sigma s)$ *of* $\Diamond \Rightarrow \Diamond \mid \lfloor r_1 \rfloor \Rightarrow g\ r_0\ r_1$))›
| ‹*combine*$_{nd}$*-Sync-*$\omega$ $A_0$ $E$ $A_1$ $i_0$ $i_1$ $g$ $\sigma s$ =
$\{r \mid r\ r_0\ r_1.\ g\ r_0\ r_1 = \lfloor r \rfloor \wedge r_0 \in \omega\ A_0\ (i_0\ \sigma s) \wedge r_1 \in \omega\ A_1\ (i_1\ \sigma s)\}$›

**fun** *combine*$_d$*-Sync* ::
‹[$('\sigma_0,\ 'e,\ 'r_0,\ '\alpha_0)$ $A_d$*-scheme*, $'e$ *set*, $('\sigma_1,\ 'e,\ 'r_1,\ '\alpha_1)$ $A_d$*-scheme*,
$'\sigma \Rightarrow '\sigma_0,\ '\sigma \Rightarrow '\sigma_1,\ '\sigma_0 \Rightarrow '\sigma_1 \Rightarrow '\sigma,\ 'r_0 \Rightarrow 'r_1 \Rightarrow 'r$ *option*] $\Rightarrow$ ($'\sigma,\ 'e,\ 'r$) $A_d$›
**and** *combine*$_{nd}$*-Sync* ::
‹[$('\sigma_0,\ 'e,\ 'r_0,\ '\alpha_0)$ $A_{nd}$*-scheme*, $'e$ *set*, $('\sigma_1,\ 'e,\ 'r_1,\ '\alpha_1)$ $A_{nd}$*-scheme*,
$'\sigma \Rightarrow '\sigma_0,\ '\sigma \Rightarrow '\sigma_1,\ '\sigma_0 \Rightarrow '\sigma_1 \Rightarrow '\sigma,\ 'r_0 \Rightarrow 'r_1 \Rightarrow 'r$ *option*] $\Rightarrow$ ($'\sigma,\ 'e,\ 'r$) $A_{nd}$›
**where** ‹*combine*$_d$*-Sync* $A_0$ $E$ $A_1$ $i_0$ $i_1$ $f$ $g$ =
(|$\tau$ = *combine*$_d$*-Sync-*$\tau$ $A_0$ $E$ $A_1$ $i_0$ $i_1$ $f$, $\omega$ = *combine*$_d$*-Sync-*$\omega$ $A_0$ $E$ $A_1$ $i_0$ $i_1$ $g$|)›
| ‹*combine*$_{nd}$*-Sync* $A_0$ $E$ $A_1$ $i_0$ $i_1$ $f$ $g$ =
(|$\tau$ = *combine*$_{nd}$*-Sync-*$\tau$ $A_0$ $E$ $A_1$ $i_0$ $i_1$ $f$, $\omega$ = *combine*$_{nd}$*-Sync-*$\omega$ $A_0$ $E$ $A_1$ $i_0$ $i_1$ $g$|)›

### 5.1.2 Specializations

**definition** *combine*$_{dPairlist}$*-Sync* ::
‹[$('\sigma,\ 'e,\ 'r,\ '\alpha)$ $A_d$*-scheme*, $'e$ *set*, $('\sigma,\ 'e,\ 'r,\ '\beta)$ $A_d$*-scheme*] $\Rightarrow$ ($'\sigma$ *list*, $'e$, $'r$) $A_d$›
**where** ‹*combine*$_{dPairlist}$*-Sync* $A_0$ $E$ $A_1$ $\equiv$
*combine*$_d$*-Sync* $A_0$ $E$ $A_1$ *hd* $(\lambda\sigma s.\ hd\ (tl\ \sigma s))$ $(\lambda s\ t.\ [s,\ t])$ $(\lambda s\ t.$ *if* $s = t$ *then* $\lfloor s \rfloor$ *else* $\Diamond$)›
**definition** *combine*$_{ndPairlist}$*-Sync* ::

‹[('σ, 'e, 'r, 'α) $A_{nd}$-scheme, 'e set, ('σ, 'e, 'r, 'β) $A_{nd}$-scheme] ⇒ ('σ list, 'e, 'r) $A_{nd}$›
**where** ‹$combine_{ndPairlist}$-Sync $A_0$ $E$ $A_1$ ≡
$combine_{nd}$-Sync $A_0$ $E$ $A_1$ hd (λσs. hd (tl σs)) (λs t. [s, t]) (λs t. if s = t then ⌊s⌋ else ◊)›

**definition** $combine_{dPair}$-Sync ::
‹[('$\sigma_0$, 'e, 'r, 'α) $A_d$-scheme, 'e set, ('$\sigma_1$, 'e, 'r, 'β) $A_d$-scheme] ⇒ ('$\sigma_0$ × '$\sigma_1$, 'e, 'r) $A_d$›
**where** ‹$combine_{dPair}$-Sync $A_0$ $E$ $A_1$ ≡
$combine_d$-Sync $A_0$ $E$ $A_1$ fst snd Pair (λs t. if s = t then ⌊s⌋ else ◊)›
**definition** $combine_{ndPair}$-Sync ::
‹[('$\sigma_0$, 'e, 'r, 'α) $A_{nd}$-scheme, 'e set, ('$\sigma_1$, 'e, 'r, 'β) $A_{nd}$-scheme] ⇒ ('$\sigma_0$ × '$\sigma_1$, 'e, 'r) $A_{nd}$›
**where** ‹$combine_{ndPair}$-Sync $A_0$ $E$ $A_1$ ≡
$combine_{nd}$-Sync $A_0$ $E$ $A_1$ fst snd Pair (λs t. if s = t then ⌊s⌋ else ◊)›

**definition** $combine_{dListslenL}$-Sync ::
‹[('σ list, 'e, 'r, 'α) $A_d$-scheme, nat, 'e set, ('σ list, 'e, 'r, 'β) $A_d$-scheme] ⇒ ('σ list, 'e, 'r) $A_d$›
**where** ‹$combine_{dListslenL}$-Sync $A_0$ $len_0$ $E$ $A_1$ ≡
$combine_d$-Sync $A_0$ $E$ $A_1$ (take $len_0$) (drop $len_0$) (@) (λs t. if s = t then ⌊s⌋ else ◊)›
**definition** $combine_{ndListslenL}$-Sync ::
‹[('σ list, 'e, 'r, 'α) $A_{nd}$-scheme, nat, 'e set, ('σ list, 'e, 'r, 'β) $A_{nd}$-scheme] ⇒ ('σ list, 'e, 'r) $A_{nd}$›
**where** ‹$combine_{ndListslenL}$-Sync $A_0$ $len_0$ $E$ $A_1$ ≡
$combine_{nd}$-Sync $A_0$ $E$ $A_1$ (take $len_0$) (drop $len_0$) (@) (λs t. if s = t then ⌊s⌋ else ◊)›

**definition** $combine_{dRlist}$-Sync ::
‹[('σ, 'e, 'r, 'α) $A_d$-scheme, 'e set, ('σ list, 'e, 'r, 'β) $A_d$-scheme] ⇒ ('σ list, 'e, 'r) $A_d$›
**where** ‹$combine_{dRlist}$-Sync $A_0$ $E$ $A_1$ ≡
$combine_d$-Sync $A_0$ $E$ $A_1$ hd tl (#) (λs t. if s = t then ⌊s⌋ else ◊)›
**definition** $combine_{ndRlist}$-Sync ::
‹[('σ, 'e, 'r, 'α) $A_{nd}$-scheme, 'e set, ('σ list, 'e, 'r, 'β) $A_{nd}$-scheme] ⇒ ('σ list, 'e, 'r) $A_{nd}$›
**where** ‹$combine_{ndRlist}$-Sync $A_0$ $E$ $A_1$ ≡
$combine_{nd}$-Sync $A_0$ $E$ $A_1$ hd tl (#) (λs t. if s = t then ⌊s⌋ else ◊)›

**lemmas** $combine_{Pairlist}$-Sync-defs = $combine_{dPairlist}$-Sync-def $combine_{ndPairlist}$-Sync-def
**and** $combine_{Pair}$-Sync-defs = $combine_{dPair}$-Sync-def $combine_{ndPair}$-Sync-def
**and** $combine_{ListslenL}$-Sync-defs = $combine_{dListslenL}$-Sync-def $combine_{ndListslenL}$-Sync-def
**and** $combine_{Rlist}$-Sync-defs = $combine_{dRlist}$-Sync-def $combine_{ndRlist}$-Sync-def

**lemmas** combine-Sync-defs =
$combine_{Pairlist}$-Sync-defs $combine_{Pair}$-Sync-defs $combine_{ListslenL}$-Sync-defs $combine_{Rlist}$-Sync-defs

**bundle** *combine$_{nd}$-Sync-syntax* **begin**

**notation** *combine$_{dPairlist}$-Sync* (‹⟪- $_{d}\otimes$⟦-⟧$_{Pairlist}$ -⟫› [0, 0, 0])
**notation** *combine$_{ndPairlist}$-Sync* (‹⟪- $_{nd}\otimes$⟦-⟧$_{Pairlist}$ -⟫› [0, 0, 0])
**notation** *combine$_{dPair}$-Sync* (‹⟪- $_{d}\otimes$⟦-⟧$_{Pair}$ -⟫› [0, 0, 0])
**notation** *combine$_{ndPair}$-Sync* (‹⟪- $_{nd}\otimes$⟦-⟧$_{Pair}$ -⟫› [0, 0, 0])
**notation** *combine$_{dListslenL}$-Sync* (‹⟪- $_{d}\otimes$⟦-, -⟧$_{ListslenL}$ -⟫› [0, 0, 0, 0])
**notation** *combine$_{ndListslenL}$-Sync* (‹⟪- $_{nd}\otimes$⟦-, -⟧$_{ListslenL}$ -⟫› [0, 0, 0, 0])
**notation** *combine$_{dRlist}$-Sync* (‹⟪- $_{d}\otimes$⟦-⟧$_{Rlist}$ -⟫› [0, 0, 0])
**notation** *combine$_{ndRlist}$-Sync* (‹⟪- $_{nd}\otimes$⟦-⟧$_{Rlist}$ -⟫› [0, 0, 0])

**end**

**unbundle** *combine$_{nd}$-Sync-syntax*

## 5.2 First Properties

**lemma** *finite-trans-combine$_{nd}$-Sync-simps* [*simp*] :
‹*finite-trans* $A_0 \Longrightarrow$ *finite-trans* $A_1 \Longrightarrow$ *finite-trans* ⟪$A_0$ $_{nd}\otimes$⟦$E$⟧$_{Pairlist}$ $A_1$⟫›
‹*finite-trans* $B_0 \Longrightarrow$ *finite-trans* $B_1 \Longrightarrow$ *finite-trans* ⟪$B_0$ $_{nd}\otimes$⟦$E$⟧$_{Pair}$ $B_1$⟫›
‹*finite-trans* $C_0 \Longrightarrow$ *finite-trans* $C_1 \Longrightarrow$ *finite-trans* ⟪$C_0$ $_{nd}\otimes$⟦$len_0$, $E$⟧$_{ListslenL}$ $C_1$⟫›
‹*finite-trans* $D_0 \Longrightarrow$ *finite-trans* $D_1 \Longrightarrow$ *finite-trans* ⟪$D_0$ $_{nd}\otimes$⟦$E$⟧$_{Rlist}$ $D_1$⟫›
⟨*proof*⟩

**lemma** *ε-combine$_{Pairlist}$-Sync*:
‹$\varepsilon$ ⟪$A_0$ $_{d}\otimes$⟦$E$⟧$_{Pairlist}$ $A_1$⟫ $\sigma s$ = *combine-Sync-ε* $A_0$ $E$ $A_1$ *hd* (*hd* ∘ *tl*) $\sigma s$›
‹$\varepsilon$ ⟪$B_0$ $_{nd}\otimes$⟦$E$⟧$_{Pairlist}$ $B_1$⟫ $\sigma s$ = *combine-Sync-ε* $B_0$ $E$ $B_1$ *hd* (*hd* ∘ *tl*) $\sigma s$›
⟨*proof*⟩

**lemma** *ε-combine$_{Pair}$-Sync*:
‹$\varepsilon$ ⟪$A_0$ $_{d}\otimes$⟦$E$⟧$_{Pair}$ $A_1$⟫ $\sigma s$ = *combine-Sync-ε* $A_0$ $E$ $A_1$ *fst snd* $\sigma s$›
‹$\varepsilon$ ⟪$B_0$ $_{nd}\otimes$⟦$E$⟧$_{Pair}$ $B_1$⟫ $\sigma s$ = *combine-Sync-ε* $B_0$ $E$ $B_1$ *fst snd* $\sigma s$›
⟨*proof*⟩

**lemma** *ε-combine$_{ListslenL}$-Sync*:
‹$\varepsilon$ ⟪$A_0$ $_{d}\otimes$⟦$len_0$, $E$⟧$_{ListslenL}$ $A_1$⟫ $\sigma s$ = *combine-Sync-ε* $A_0$ $E$ $A_1$ (*take* $len_0$) (*drop* $len_0$) $\sigma s$›
‹$\varepsilon$ ⟪$B_0$ $_{nd}\otimes$⟦$len_0$, $E$⟧$_{ListslenL}$ $B_1$⟫ $\sigma s$ = *combine-Sync-ε* $B_0$ $E$ $B_1$ (*take* $len_0$) (*drop* $len_0$) $\sigma s$›
⟨*proof*⟩

**lemma** *ε-combine$_{Rlist}$-Sync*:
‹$\varepsilon$ ⟪$A_0$ $_{d}\otimes$⟦$E$⟧$_{Rlist}$ $A_1$⟫ $\sigma s$ = *combine-Sync-ε* $A_0$ $E$ $A_1$ *hd tl* $\sigma s$›
‹$\varepsilon$ ⟪$B_0$ $_{nd}\otimes$⟦$E$⟧$_{Rlist}$ $B_1$⟫ $\sigma s$ = *combine-Sync-ε* $B_0$ $E$ $B_1$ *hd tl* $\sigma s$›
⟨*proof*⟩

**lemma** $\varrho$-*combine*$_{Pairlist}$-*Sync*:
‹$\varrho$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\!\rangle = \{\sigma s.\ hd\ \sigma s \in \varrho\ A_0 \wedge hd\ (tl\ \sigma s) \in \varrho\ A_1 \wedge \omega\ A_0\ (hd\ \sigma s) = \omega\ A_1\ (hd\ (tl\ \sigma s))\}$›
‹$\varrho$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Pairlist}\ B_1\rangle\!\rangle = \{\sigma s.\ hd\ \sigma s \in \varrho\ B_0 \wedge hd\ (tl\ \sigma s) \in \varrho\ B_1 \wedge \omega\ B_0\ (hd\ \sigma s) \cap \omega\ B_1\ (hd\ (tl\ \sigma s)) \neq \{\}\}$›
⟨*proof*⟩

**lemma** $\varrho$-*combine*$_{Pair}$-*Sync*:
‹$\varrho$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pair}\ A_1\rangle\!\rangle = \{(\sigma_0, \sigma_1).\ \sigma_0 \in \varrho\ A_0 \wedge \sigma_1 \in \varrho\ A_1 \wedge \omega\ A_0\ \sigma_0 = \omega\ A_1\ \sigma_1\}$›
‹$\varrho$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Pair}\ B_1\rangle\!\rangle = \{(\sigma_0, \sigma_1).\ \sigma_0 \in \varrho\ B_0 \wedge \sigma_1 \in \varrho\ B_1 \wedge \omega\ B_0\ \sigma_0 \cap \omega\ B_1\ \sigma_1 \neq \{\}\}$›
⟨*proof*⟩

**lemma** $\varrho$-*combine*$_{ListslenL}$-*Sync*:
‹$\varrho$ $\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle =$
$\{\sigma s.\ take\ len_0\ \sigma s \in \varrho\ A_0 \wedge drop\ len_0\ \sigma s \in \varrho\ A_1 \wedge \omega\ A_0\ (take\ len_0\ \sigma s) = \omega\ A_1\ (drop\ len_0\ \sigma s)\}$›
‹$\varrho$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{ListslenL}\ B_1\rangle\!\rangle =$
$\{\sigma s.\ take\ len_0\ \sigma s \in \varrho\ B_0 \wedge drop\ len_0\ \sigma s \in \varrho\ B_1 \wedge \omega\ B_0\ (take\ len_0\ \sigma s) \cap \omega\ B_1\ (drop\ len_0\ \sigma s) \neq \{\}\}$›
⟨*proof*⟩

**lemma** $\varrho$-*combine*$_{Rlist}$-*Sync*:
‹$\varrho$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Rlist}\ A_1\rangle\!\rangle =$
$\{\sigma s.\ hd\ \sigma s \in \varrho\ A_0 \wedge tl\ \sigma s \in \varrho\ A_1 \wedge \omega\ A_0\ (hd\ \sigma s) = \omega\ A_1\ (tl\ \sigma s)\}$›
‹$\varrho$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Rlist}\ B_1\rangle\!\rangle =$
$\{\sigma s.\ hd\ \sigma s \in \varrho\ B_0 \wedge tl\ \sigma s \in \varrho\ B_1 \wedge \omega\ B_0\ (hd\ \sigma s) \cap \omega\ B_1\ (tl\ \sigma s) \neq \{\}\}$›
⟨*proof*⟩

**lemma** *combine-Sync-*$\tau$ [*simp*] :
‹*combine*$_d$-*Sync-*$\tau$ $(A_0(\!|\omega := some\text{-}\omega_0|\!))\ E\ (A_1(\!|\omega := some\text{-}\omega_1|\!)) = combine_d\text{-}Sync\text{-}\tau\ A_0\ E\ A_1$›
‹*combine*$_{nd}$-*Sync-*$\tau$ $(B_0(\!|\omega := some\text{-}\omega_0'|\!))\ E\ (B_1(\!|\omega := some\text{-}\omega_1'|\!)) = combine_{nd}\text{-}Sync\text{-}\tau\ B_0\ E\ B_1$›
**for** $A$ :: ‹$('\sigma,\ 'a,\ '\sigma\ option,\ 'r\ option,\ '\alpha)$ *A-scheme*›
**and** $B$ :: ‹$('\sigma,\ 'a,\ '\sigma\ set,\ 'r\ set,\ '\alpha)$ *A-scheme*›
⟨*proof*⟩

## 5.3 Reachability

**lemma** $\mathcal{R}_d$-*combine*$_{dListslenL}$-*Sync-subset*:
‹$\mathcal{R}_d$ $\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle\ (s_0\ @\ s_1) \subseteq \{t_0\ @\ t_1 |\ t_0\ t_1.\ t_0 \in \mathcal{R}_d\ A_0\ s_0 \wedge t_1 \in \mathcal{R}_d\ A_1\ s_1\}$› (**is** ‹$?S_A \subseteq$ -›)
**if** *same-length-*$\mathcal{R}_d$: ‹$\bigwedge t_0.\ t_0 \in \mathcal{R}_d\ A_0\ s_0 \Longrightarrow length\ t_0 = len_0$›
⟨*proof*⟩

**lemma** $\mathcal{R}_{nd}$-*combine*$_{ndListslenL}$-*Sync-subset*:
‹$\mathcal{R}_{nd}$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{ListslenL}\ B_1\rangle\!\rangle$ $(s_0\ @\ s_1) \subseteq \{t_0\ @\ t_1 |\ t_0\ t_1.\ t_0 \in \mathcal{R}_{nd}\ B_0\ s_0 \wedge t_1 \in \mathcal{R}_{nd}\ B_1\ s_1\}$› (**is** ‹$?S_B \subseteq$ -›)
**if** *same-length*-$\mathcal{R}_{nd}$: ‹$\bigwedge t_0.\ t_0 \in \mathcal{R}_{nd}\ B_0\ s_0 \Longrightarrow$ *length* $t_0 = len_0$›
⟨*proof*⟩

**lemma** $\mathcal{R}_d$-*combine*$_{dPairlist}$-*Sync-subset*:
‹$\mathcal{R}_d$ $\langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\!\rangle$ $[s_0,\ s_1] \subseteq \{[t_0,\ t_1] |\ t_0\ t_1.\ t_0 \in \mathcal{R}_d\ A_0\ s_0 \wedge t_1 \in \mathcal{R}_d\ A_1\ s_1\}$› (**is** ‹$?S_A \subseteq$ -›)
**and** $\mathcal{R}_{nd}$-*combine*$_{ndPairlist}$-*Sync-subset*:
‹$\mathcal{R}_{nd}$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Pairlist}\ B_1\rangle\!\rangle$ $[s_0,\ s_1] \subseteq \{[t_0,\ t_1] |\ t_0\ t_1.\ t_0 \in \mathcal{R}_{nd}\ B_0\ s_0 \wedge t_1 \in \mathcal{R}_{nd}\ B_1\ s_1\}$› (**is** ‹$?S_B \subseteq$ -›)
⟨*proof*⟩

**lemma** $\mathcal{R}_d$-*combine*$_{dPair}$-*Sync-subset*:
‹$\mathcal{R}_d$ $\langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{Pair}\ A_1\rangle\!\rangle$ $(s_0,\ s_1) \subseteq \mathcal{R}_d\ A_0\ s_0 \times \mathcal{R}_d\ A_1\ s_1$› (**is** ‹$?S_A \subseteq$ -›)
**and** $\mathcal{R}_{nd}$-*combine*$_{ndPair}$-*Sync-subset*:
‹$\mathcal{R}_{nd}$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Pair}\ B_1\rangle\!\rangle$ $(s_0,\ s_1) \subseteq \mathcal{R}_{nd}\ B_0\ s_0 \times \mathcal{R}_{nd}\ B_1\ s_1$› (**is** ‹$?S_B \subseteq$ -›)
⟨*proof*⟩

**lemma** $\mathcal{R}_d$-*combine*$_{dRlist}$-*Sync-subset*:
‹$\mathcal{R}_d$ $\langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{Rlist}\ A_1\rangle\!\rangle$ $(s_0\ \#\ \sigma s) \subseteq \{t_0\ \#\ \sigma t |\ t_0\ \sigma t.\ t_0 \in \mathcal{R}_d\ A_0\ s_0 \wedge \sigma t \in \mathcal{R}_d\ A_1\ \sigma s\}$› (**is** ‹$?S_A \subseteq$ -›)
**and** $\mathcal{R}_{nd}$-*combine*$_{ndRlist}$-*Sync-subset*:
‹$\mathcal{R}_{nd}$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Rlist}\ B_1\rangle\!\rangle$ $(s_0\ \#\ \sigma s) \subseteq \{t_0\ \#\ \sigma t |\ t_0\ \sigma t.\ t_0 \in \mathcal{R}_{nd}\ B_0\ s_0 \wedge \sigma t \in \mathcal{R}_{nd}\ B_1\ \sigma s\}$› (**is** ‹$?S_B \subseteq$ -›)
⟨*proof*⟩

## 5.4 Normalization

**lemma** $\omega$-*combine*$_{Pairlist}$-*Sync-behaviour*:
‹$\omega$ $\langle\!\langle\langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}$ $[s_0,\ s_1] = \omega$ $\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Pairlist}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle$ $[s_0,\ s_1]$›
⟨*proof*⟩

**lemma** $\omega$-*combine*$_{Pair}$-*Sync-behaviour*:
‹$\omega$ $\langle\!\langle\langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{Pair}\ A_1\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}$ $(s_0,\ s_1) = \omega$ $\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Pair}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle$ $(s_0,\ s_1)$›
⟨*proof*⟩

**lemma** $\omega$-*combine*$_{ListslenL}$-*Sync-behaviour*:
‹$\omega$ $\langle\!\langle\langle\!\langle A_0\ {}_{d}\otimes[\![len_0,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}$ $(\sigma s_0\ @\ \sigma s_1) = \omega$ $\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![len_0,\ E]\!]_{ListslenL}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle$ $(\sigma s_0\ @\ \sigma s_1)$›
⟨*proof*⟩

**lemma** $\omega$-*combine*$_{Rlist}$-*Sync-behaviour*:

‹$\omega$ $\langle\langle\langle A_0\ {}_d\otimes[\![E]\!]_{Rlist}\ A_1\rangle\rangle\rangle_{d\hookrightarrow nd}\ (s_0\ \#\ \sigma s_1) = \omega\ \langle\langle\langle A_0\rangle\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Rlist}\ \langle\langle A_1\rangle\rangle_{d\hookrightarrow nd}\rangle\rangle$ $(s_0\ \#\ \sigma s_1)$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{Pairlist}$-*Sync-behaviour-when-indep*:
‹$\varepsilon\ A_0\ s_0 \cap \varepsilon\ A_1\ s_1 \subseteq E \Longrightarrow$
$\tau\ \langle\langle\langle A_0\ {}_d\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\rangle\rangle_{d\hookrightarrow nd}\ [s_0,\ s_1]\ e = \tau\ \langle\langle\langle A_0\rangle\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Pairlist}$ $\langle\langle A_1\rangle\rangle_{d\hookrightarrow nd}\rangle\rangle\ [s_0,\ s_1]\ e$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{Pair}$-*Sync-behaviour-when-indep*:
‹$\varepsilon\ A_0\ s_0 \cap \varepsilon\ A_1\ s_1 \subseteq E \Longrightarrow$
$\tau\ \langle\langle\langle A_0\ {}_d\otimes[\![E]\!]_{Pair}\ A_1\rangle\rangle\rangle_{d\hookrightarrow nd}\ (s_0,\ s_1)\ e = \tau\ \langle\langle\langle A_0\rangle\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Pair}\ \langle\langle A_1\rangle\rangle_{d\hookrightarrow nd}\rangle\rangle$ $(s_0,\ s_1)\ e$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{ListslenL}$-*Sync-behaviour-when-indep*:
‹$\varepsilon\ A_0\ \sigma s_0 \cap \varepsilon\ A_1\ \sigma s_1 \subseteq E \Longrightarrow$ *length* $\sigma s_0 = len_0 \Longrightarrow$
$\tau\ \langle\langle\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{ListslenL}\ A_1\rangle\rangle\rangle_{d\hookrightarrow nd}\ (\sigma s_0\ @\ \sigma s_1)\ e = \tau\ \langle\langle\langle A_0\rangle\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![len_0,$ $E]\!]_{ListslenL}\ \langle\langle A_1\rangle\rangle_{d\hookrightarrow nd}\rangle\rangle\ (\sigma s_0\ @\ \sigma s_1)\ e$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{Rlist}$-*Sync-behaviour-when-indep*:
‹$\varepsilon\ A_0\ s_0 \cap \varepsilon\ A_1\ \sigma s_1 \subseteq E \Longrightarrow$
$\tau\ \langle\langle\langle A_0\ {}_d\otimes[\![E]\!]_{Rlist}\ A_1\rangle\rangle\rangle_{d\hookrightarrow nd}\ (s_0\ \#\ \sigma s_1)\ e = \tau\ \langle\langle\langle A_0\rangle\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Rlist}$ $\langle\langle A_1\rangle\rangle_{d\hookrightarrow nd}\rangle\rangle\ (s_0\ \#\ \sigma s_1)\ e$›
⟨*proof*⟩

**method** $P_{SKIPS}$-*when-indep-method* **uses** *R-d-subset* =
*fold* $P_{SKIPS}$-*nd-from-det-to-ndet-is-*$P_{SKIPS}$-*d*,
*rule* $P_{SKIPS}$-*nd-eqI-strong-id*,
*unfold* $\mathcal{R}_{nd}$-*from-det-to-ndet*,
*all* ‹*drule set-mp*[*OF R-d-subset*, *rotated*]›

**method** *P-when-indep-method* **uses** *R-d-subset* =
*fold P-nd-from-det-to-ndet-is-P-d*,
*rule P-nd-eqI-strong-id*,
*unfold* $\mathcal{R}_{nd}$-*from-det-to-ndet*,
*all* ‹*drule set-mp*[*OF R-d-subset*, *rotated*]›

**lemma** $P_{SKIPS}$-*combine*$_{Pairlist}$-*Sync-behaviour-when-indep*:
‹$P_{SKIPS}\langle\langle\langle A_0\ {}_d\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\rangle\rangle_d\ [s_0,\ s_1] = P_{SKIPS}\langle\langle\langle\langle A_0\rangle\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Pairlist}$ $\langle\langle A_1\rangle\rangle_{d\hookrightarrow nd}\rangle\rangle\rangle_{nd}\ [s_0,\ s_1]$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ s_1$›

⟨*proof*⟩

**lemma** *P-combine*$_{Pairlist}$*-Sync-behaviour-when-indep*:
‹$P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ [s_0,\ s_1] = P\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Pairlist}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\rangle\!\rangle_{nd}$ $[s_0,\ s_1]$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ s_1$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-combine*$_{Pair}$*-Sync-behaviour-when-indep*:
‹$P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pair}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (s_0,\ s_1) = P_{SKIPS}\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Pair}$ $\langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\rangle\!\rangle_{nd}\ (s_0,\ s_1)$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ s_1$›
⟨*proof*⟩

**lemma** *P-combine*$_{Pair}$*-Sync-behaviour-when-indep*:
‹$P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pair}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (s_0,\ s_1) = P\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Pair}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\rangle\!\rangle_{nd}$ $(s_0,\ s_1)$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ s_1$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-combine*$_{ListslenL}$*-Sync-behaviour-when-indep*:
‹$P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (\sigma s_0\ @\ \sigma s_1) = P_{SKIPS}\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}$ ${}_{nd}\otimes[\![len_0,\ E]\!]_{ListslenL}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\rangle\!\rangle_{nd}\ (\sigma s_0\ @\ \sigma s_1)$›
**if** ‹*indep-enabl* $A_0\ \sigma s_0\ E\ A_1\ \sigma s_1$› **and** ‹$\bigwedge\sigma t_0.\ \sigma t_0 \in \mathcal{R}_d\ A_0\ \sigma s_0 \Longrightarrow$ *length* $\sigma t_0 = len_0$›
⟨*proof*⟩

**lemma** *P-combine*$_{ListslenL}$*-Sync-behaviour-when-indep*:
‹$P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (\sigma s_0\ @\ \sigma s_1) = P\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![len_0,$ $E]\!]_{ListslenL}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\rangle\!\rangle_{nd}\ (\sigma s_0\ @\ \sigma s_1)$›
**if** ‹*indep-enabl* $A_0\ \sigma s_0\ E\ A_1\ \sigma s_1$› **and** ‹$\bigwedge\sigma t_0.\ \sigma t_0 \in \mathcal{R}_d\ A_0\ \sigma s_0 \Longrightarrow$ *length* $\sigma t_0 = len_0$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-combine*$_{Rlist}$*-Sync-behaviour-when-indep*:
‹$P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Rlist}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (s_0\ \#\ \sigma s_1) = P_{SKIPS}\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Rlist}$ $\langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\rangle\!\rangle_{nd}\ (s_0\ \#\ \sigma s_1)$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ \sigma s_1$›
⟨*proof*⟩

**lemma** *P-combine*$_{Rlist}$*-Sync-behaviour-when-indep*:
‹$P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Rlist}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (s_0\ \#\ \sigma s_1) = P\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{Rlist}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\rangle\!\rangle_{nd}$ $(s_0\ \#\ \sigma s_1)$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ \sigma s_1$›
⟨*proof*⟩

# Chapter 6

# Compactification of Synchronization Product

## 6.1 Iterated Combine

### 6.1.1 Definitions

**fun** *iterated-combine*$_d$*-Sync* :: ‹$'e$ *set* $\Rightarrow$ $(\sigma, 'e, 'r)$ $A_d$ *list* $\Rightarrow$ $(\sigma$ *list*, $'e, 'r)$ $A_d$› (‹$\langle\!\langle_d\bigotimes [\![-]\!]\ -\rangle\!\rangle$› $[0, 0]$)
  **where** ‹$\langle\!\langle_d\bigotimes [\![E]\!]\ []\rangle\!\rangle = (\!|\tau = \lambda\sigma s\ a.\ \Diamond, \omega = \lambda\sigma s.\ \Diamond|\!)$›
  | ‹$\langle\!\langle_d\bigotimes [\![E]\!]\ [A_0]\rangle\!\rangle = {}_d\langle\!\langle A_0\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$›
  | ‹$\langle\!\langle_d\bigotimes [\![E]\!]\ A_0 \# A_1 \# As\rangle\!\rangle = \langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Rlist}\ \langle\!\langle_d\bigotimes [\![E]\!]\ A_1 \# As\rangle\!\rangle\rangle\!\rangle$›

**fun** *iterated-combine*$_{nd}$*-Sync* :: ‹$'e$ *set* $\Rightarrow$ $(\sigma, 'e, 'r)$ $A_{nd}$ *list* $\Rightarrow$ $(\sigma$ *list*, $'e, 'r)$ $A_{nd}$› (‹$\langle\!\langle_{nd}\bigotimes [\![-]\!]\ -\rangle\!\rangle$› $[0, 0]$)
  **where** ‹$\langle\!\langle_{nd}\bigotimes [\![E]\!]\ []\rangle\!\rangle = (\!|\tau = \lambda\sigma s\ a.\ \{\}, \omega = \lambda\sigma s.\ \{\}|\!)$›
  | ‹$\langle\!\langle_{nd}\bigotimes [\![E]\!]\ [A_0]\rangle\!\rangle = {}_{nd}\langle\!\langle A_0\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$›
  | ‹$\langle\!\langle_{nd}\bigotimes [\![E]\!]\ A_0 \# A_1 \# As\rangle\!\rangle = \langle\!\langle A_0\ {}_{nd}\otimes[\![E]\!]_{Rlist}\ \langle\!\langle_{nd}\bigotimes [\![E]\!]\ A_1 \# As\rangle\!\rangle\rangle\!\rangle$›

**lemma** *iterated-combine*$_d$*-Sync-simps-bis*: ‹$As \neq [] \Longrightarrow \langle\!\langle_d\bigotimes [\![E]\!]\ A_0 \# As\rangle\!\rangle = \langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Rlist}\ \langle\!\langle_d\bigotimes [\![E]\!]\ As\rangle\!\rangle\rangle\!\rangle$›
  **and** *iterated-combine*$_{nd}$*-Sync-simps-bis*: ‹$Bs \neq [] \Longrightarrow \langle\!\langle_{nd}\bigotimes [\![E]\!]\ B_0 \# Bs\rangle\!\rangle = \langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Rlist}\ \langle\!\langle_{nd}\bigotimes [\![E]\!]\ Bs\rangle\!\rangle\rangle\!\rangle$›
  ⟨*proof*⟩

**fun** *iterated-combine*$_d$*-Sync-*$\varepsilon$ :: ‹$(\sigma, 'e, 'r, '\alpha)$ $A_d$*-scheme list* $\Rightarrow$ $'e$ *set* $\Rightarrow$ $\sigma$ *list* $\Rightarrow$ $'e$ *set*›
  **where** ‹*iterated-combine*$_d$*-Sync-*$\varepsilon$ $[]\ E\ \sigma s = \{\}$›
  | ‹*iterated-combine*$_d$*-Sync-*$\varepsilon$ $[A_0]\ E\ \sigma s = \varepsilon\ A_0\ (hd\ \sigma s)$›
  | ‹*iterated-combine*$_d$*-Sync-*$\varepsilon$ $(A_0 \# A_1 \# As)\ E\ \sigma s =$
    *combine-sets-Sync* $(\varepsilon\ A_0\ (hd\ \sigma s))\ E$ (*iterated-combine*$_d$*-Sync-*$\varepsilon$ $(A_1 \# As)\ E\ (tl\ \sigma s)$)›

**fun** *iterated-combine*$_{nd}$*-Sync-*$\varepsilon$ :: ‹($'\sigma$, $'e$, $'r$, $'\alpha$) $A_{nd}$*-scheme list* $\Rightarrow$ $'e$ *set* $\Rightarrow$ $'\sigma$ *list* $\Rightarrow$ $'e$ *set*›
**where** ‹*iterated-combine*$_{nd}$*-Sync-*$\varepsilon$ [] $E$ $\sigma s$ = {}›
| ‹*iterated-combine*$_{nd}$*-Sync-*$\varepsilon$ [$A_0$] $E$ $\sigma s$ = $\varepsilon$ $A_0$ (*hd* $\sigma s$)›
| ‹*iterated-combine*$_{nd}$*-Sync-*$\varepsilon$ ($A_0$ # $A_1$ # *As*) $E$ $\sigma s$ =
*combine-sets-Sync* ($\varepsilon$ $A_0$ (*hd* $\sigma s$)) $E$ (*iterated-combine*$_{nd}$*-Sync-*$\varepsilon$ ($A_1$ # *As*) $E$ (*tl* $\sigma s$))›

**lemma** *iterated-combine*$_d$*-Sync-*$\varepsilon$*-simps-bis*:
‹*As* $\neq$ [] $\Longrightarrow$ *iterated-combine*$_d$*-Sync-*$\varepsilon$ ($A_0$ # *As*) $E$ $\sigma s$ =
*combine-sets-Sync* ($\varepsilon$ $A_0$ (*hd* $\sigma s$)) $E$ (*iterated-combine*$_d$*-Sync-*$\varepsilon$ *As* $E$ (*tl* $\sigma s$))›
**and** *iterated-combine*$_{nd}$*-Sync-*$\varepsilon$*-simps-bis*:
‹*Bs* $\neq$ [] $\Longrightarrow$ *iterated-combine*$_{nd}$*-Sync-*$\varepsilon$ ($B_0$ # *Bs*) $E$ $\sigma s$ =
*combine-sets-Sync* ($\varepsilon$ $B_0$ (*hd* $\sigma s$)) $E$ (*iterated-combine*$_{nd}$*-Sync-*$\varepsilon$ *Bs* $E$ (*tl* $\sigma s$))›
⟨*proof*⟩

### 6.1.2 First Results

**lemma** $\varepsilon$*-iterated-combine*$_d$*-Sync*:
‹*length* $\sigma s$ = *length As* $\Longrightarrow$ $\varepsilon$ $\langle\!\langle_d\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle$ $\sigma s$ = *iterated-combine*$_d$*-Sync-*$\varepsilon$ *As* $E$ $\sigma s$›
⟨*proof*⟩

**lemma** $\varepsilon$*-iterated-combine*$_{nd}$*-Sync*:
‹*length* $\sigma s$ = *length Bs* $\Longrightarrow$ $\varepsilon$ $\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *Bs*$\rangle\!\rangle$ $\sigma s$ = *iterated-combine*$_{nd}$*-Sync-*$\varepsilon$ *Bs* $E$ $\sigma s$›
⟨*proof*⟩

**lemma** *combine*$_{ListslenL}$*-Sync-combine*$_{Rlist}$*-Sync-eq*:
‹$\varepsilon$ $\langle\!\langle_d\langle\!\langle A_0\rangle\!\rangle_{\sigma\hookrightarrow\sigma\ s}\ {}_d\otimes[\![1,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle$ $\sigma s$ = $\varepsilon$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Rlist}\ A_1\rangle\!\rangle$ $\sigma s$›
‹$\tau$ $\langle\!\langle_d\langle\!\langle A_0\rangle\!\rangle_{\sigma\hookrightarrow\sigma\ s}\ {}_d\otimes[\![1,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle$ ($s_0$ # $\sigma s$) $e$ = $\tau$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Rlist}\ A_1\rangle\!\rangle$ ($s_0$ # $\sigma s$) $e$›
‹$\varepsilon$ $\langle\!\langle_{nd}\langle\!\langle B_0\rangle\!\rangle_{\sigma\hookrightarrow\sigma\ nd}\otimes[\![1,\ E]\!]_{ListslenL}\ B_1\rangle\!\rangle$ $\sigma s$ = $\varepsilon$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Rlist}\ B_1\rangle\!\rangle$ $\sigma s$›
‹$\tau$ $\langle\!\langle_{nd}\langle\!\langle B_0\rangle\!\rangle_{\sigma\hookrightarrow\sigma\ nd}\otimes[\![1,\ E]\!]_{ListslenL}\ B_1\rangle\!\rangle$ ($s_0$ # $\sigma s$) $e$ = $\tau$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Rlist}\ B_1\rangle\!\rangle$ ($s_0$ # $\sigma s$) $e$›
⟨*proof*⟩

**lemma** *combine*$_{Pairlist}$*-Sync-and-iterated-combine*$_{nd}$*-Sync-eq*:
‹$\varepsilon$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\!\rangle$ [$s_0$, $s_1$] = $\varepsilon$ $\langle\!\langle_d\bigotimes [\![E]\!]$ [$A_0$, $A_1$]$\rangle\!\rangle$ [$s_0$, $s_1$]›
‹$\tau$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\!\rangle$ [$s_0$, $s_1$] $e$ = $\tau$ $\langle\!\langle_d\bigotimes [\![E]\!]$ [$A_0$, $A_1$]$\rangle\!\rangle$ [$s_0$, $s_1$] $e$›
‹$\varepsilon$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Pairlist}\ B_1\rangle\!\rangle$ [$s_0$, $s_1$] = $\varepsilon$ $\langle\!\langle_{nd}\bigotimes [\![E]\!]$ [$B_0$, $B_1$]$\rangle\!\rangle$ [$s_0$, $s_1$]›
‹$\tau$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Pairlist}\ B_1\rangle\!\rangle$ [$s_0$, $s_1$] $e$ = $\tau$ $\langle\!\langle_{nd}\bigotimes [\![E]\!]$ [$B_0$, $B_1$]$\rangle\!\rangle$ [$s_0$, $s_1$] $e$›
⟨*proof*⟩

**lemmas** *combine*$_{Pairlist}$*-Sync-and-combine*$_{Rlist}$*-Sync-eq* =
*combine*$_{Pairlist}$*-Sync-and-iterated-combine*$_{nd}$*-Sync-eq*[*simplified*]

### 6.1.3 Reachability

**lemma** *same-length-*$\mathcal{R}_d$*-iterated-combine*$_d$*-Sync-description*:
‹*length* $\sigma s$ = *length As* $\Longrightarrow$ $\sigma s' \in \mathcal{R}_d$ $\langle\!\langle_d\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle$ $\sigma s$ $\Longrightarrow$
*length* $\sigma s'$ = *length As* $\wedge$ ($\forall$ *i* < *length As*. $\sigma s'$ ! *i* $\in \mathcal{R}_d$ (*As* ! *i*) ($\sigma s$ ! *i*))›
⟨*proof*⟩

**lemma** *same-length-*$\mathcal{R}_{nd}$*-iterated-combine*$_{nd}$*-Sync-description*:
‹*length* $\sigma s$ = *length Bs* $\Longrightarrow$ $\sigma s' \in \mathcal{R}_{nd}$ $\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *Bs*$\rangle\!\rangle$ $\sigma s$ $\Longrightarrow$
*length* $\sigma s'$ = *length Bs* $\wedge$ ($\forall$ *i* < *length Bs*. $\sigma s'$ ! *i* $\in \mathcal{R}_{nd}$ (*Bs* ! *i*) ($\sigma s$ ! *i*))›
⟨*proof*⟩

### 6.1.4 Transmission of Properties

**lemma** *finite-trans-transmission-to-iterated-combine*$_{nd}$*-Sync*:
‹($\bigwedge$*A*. *A* $\in$ *set As* $\Longrightarrow$ *finite-trans A*) $\Longrightarrow$ *finite-trans* $\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle$›
⟨*proof*⟩

**lemma** *ϱ-disjoint-ε-transmission-to-iterated-combine*$_d$*-Sync*:
‹($\bigwedge$*A*. *A* $\in$ *set As* $\Longrightarrow$ *ϱ-disjoint-ε A*) $\Longrightarrow$ *ϱ-disjoint-ε* $\langle\!\langle_d\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle$›
⟨*proof*⟩

**lemma** *ϱ-disjoint-ε-transmission-to-iterated-combine*$_{nd}$*-Sync*:
‹($\bigwedge$*A*. *A* $\in$ *set As* $\Longrightarrow$ *ϱ-disjoint-ε A*) $\Longrightarrow$ *ϱ-disjoint-ε* $\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle$›
⟨*proof*⟩

**lemma** *at-most-1-elem-term-transmission-to-iterated-combine*$_{nd}$*-Sync*:
‹($\bigwedge$*A*. *A* $\in$ *set As* $\Longrightarrow$ *at-most-1-elem-term A*) $\Longrightarrow$ *at-most-1-elem-term* $\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle$›
⟨*proof*⟩

**lemma** *same-length-indep-transmission-to-iterated-combine*$_d$*-Sync*:
‹*length* $\sigma s$ = *length As* $\Longrightarrow$
($\bigwedge$*i j*. *i* $\leq$ *length As* $\Longrightarrow$ *j* $\leq$ *length As* $\Longrightarrow$ *i* $\neq$ *j* $\Longrightarrow$
*indep-enabl* ((*A*$_0$ # *As*) ! *i*) ((*s*$_0$ # $\sigma s$) ! *i*) *E* ((*A*$_0$ # *As*) ! *j*) ((*s*$_0$ # $\sigma s$) ! *j*))
$\Longrightarrow$
*indep-enabl A*$_0$ *s*$_0$ *E* $\langle\!\langle_d\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle$ $\sigma s$›
⟨*proof*⟩

**lemma** *ω-iterated-combine*$_d$*-Sync* :
‹*length* $\sigma s$ = *length As* $\Longrightarrow$

$\omega$ $\langle\!\langle_d\bigotimes [\![E]\!]\ As\rangle\!\rangle\ \sigma s$ = (*case those* (*map2* $\omega$ *As* $\sigma s$) *of* $\Diamond \Rightarrow \Diamond$
| $\lfloor$*terms*$\rfloor \Rightarrow$ *if card* (*set terms*) = *Suc 0 then* $\lfloor$*THE r. set terms* = $\{r\}\rfloor$ *else* $\Diamond$)›
⟨*proof*⟩

**lemma** $\omega$-*iterated-combine*$_{nd}$-*Sync* :
‹*length* $\sigma s$ = *length As* $\Longrightarrow$
$\omega$ $\langle\!\langle_{nd}\bigotimes [\![E]\!]\ As\rangle\!\rangle\ \sigma s$ = (*if As* = [] *then* {} *else* {*r*. $\forall$ *i* < *length As*. *r* $\in$ $\omega$ (*As* ! *i*) ($\sigma s$ ! *i*)})›
⟨*proof*⟩

### 6.1.5 Normalization

**lemma** $\omega$-*iterated-combine*$_{nd}$-*Sync-det-ndet-conv*:
‹*length* $\sigma s$ = *length As* $\Longrightarrow$
$\omega$ $\langle\!\langle_{nd}\bigotimes [\![E]\!]\ map\ (\lambda A.\ \langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd})\ As\rangle\!\rangle\ \sigma s = \omega\ \langle\!\langle\langle\!\langle_d\bigotimes [\![E]\!]\ As\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}\ \sigma s$›
⟨*proof*⟩

**lemma** $\tau$-*iterated-combine*$_{nd}$-*Sync-behaviour-when-indep*:
‹*length* $\sigma s$ = *length As* $\Longrightarrow$
($\bigwedge$*i j*. [$\![$*i* < *length As*; *j* < *length As*; *i* $\neq$ *j*$\!]$]
$\Longrightarrow$ *indep-enabl* (*As* ! *i*) ($\sigma s$ ! *i*) *E* (*As* ! *j*) ($\sigma s$ ! *j*)) $\Longrightarrow$
$\tau$ $\langle\!\langle\langle\!\langle_d\bigotimes [\![E]\!]\ As\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}\ \sigma s\ e = \tau\ \langle\!\langle_{nd}\bigotimes [\![E]\!]\ map\ (\lambda A.\ \langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd})\ As\rangle\!\rangle\ \sigma s\ e$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*iterated-combine*$_{nd}$-*Sync-behaviour-when-indep*:
**assumes** *same-length*: ‹*length* $\sigma s$ = *length As*›
**and** *indep*: ‹$\bigwedge$*i j*. [$\![$*i* < *length As*; *j* < *length As*; *i* $\neq$ *j*$\!]$] $\Longrightarrow$
*indep-enabl* (*As* ! *i*) ($\sigma s$ ! *i*) *E* (*As* ! *j*) ($\sigma s$ ! *j*)›
**shows** ‹$P_{SKIPS}\langle\!\langle\langle\!\langle_d\bigotimes [\![E]\!]\ As\rangle\!\rangle\rangle\!\rangle_d\ \sigma s = P_{SKIPS}\langle\!\langle\langle\!\langle_{nd}\bigotimes [\![E]\!]\ map\ (\lambda A.\ \langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd})\ As\rangle\!\rangle\rangle\!\rangle_{nd}\ \sigma s$›
⟨*proof*⟩

**lemma** *P-d-iterated-combine*$_{nd}$-*Sync-behaviour-when-indep*:
**assumes** *same-length*: ‹*length* $\sigma s$ = *length As*›
**and** *indep*: ‹$\bigwedge$*i j*. [$\![$*i* < *length As*; *j* < *length As*; *i* $\neq$ *j*$\!]$] $\Longrightarrow$
*indep-enabl* (*As* ! *i*) ($\sigma s$ ! *i*) *E* (*As* ! *j*) ($\sigma s$ ! *j*)›
**shows** ‹$P\langle\!\langle\langle\!\langle_d\bigotimes [\![E]\!]\ As\rangle\!\rangle\rangle\!\rangle_d\ \sigma s = P\langle\!\langle\langle\!\langle_{nd}\bigotimes [\![E]\!]\ map\ (\lambda A.\ \langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd})\ As\rangle\!\rangle\rangle\!\rangle_{nd}\ \sigma s$›
⟨*proof*⟩

## 6.2 Compactification Theorems

### 6.2.1 Binary

#### Pair

**theorem** $P_{SKIPS}$-*nd-combine*$_{Pair}$-*Sync* :

**fixes** $E$ :: ‹$'a$ *set*› **and** $A_0$ :: ‹$('\sigma,\ 'a,\ 'r,\ '\alpha)$ $A_{nd}$-*scheme*›
**assumes** $\varrho$-*disjoint*-$\varepsilon$ : ‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$›
**and** *at-most-1-elem-term* : ‹*at-most-1-elem-term* $A_0$› ‹*at-most-1-elem-term* $A_1$›
**defines** *A-def*: ‹$A \equiv \langle\!\langle A_0\ {}_{nd}\otimes[\![E]\!]_{Pair}\ A_1\rangle\!\rangle$›
**defines** *P-def*: ‹$P \equiv P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}$› **and** *Q-def*: ‹$Q \equiv P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_{nd}$› **and** *S-def*: ‹$S \equiv P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}$›
**shows** ‹$P\ \sigma_0\ [\![E]\!]\ Q\ \sigma_1 = S\ (\sigma_0,\ \sigma_1)$›
⟨*proof*⟩

**corollary** *P-nd-combine*$_{Pair}$*-Sync* :
‹$P\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ [\![E]\!]\ P\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes[\![E]\!]_{Pair}\ A_1\rangle\!\rangle\!\rangle_{nd}\ (\sigma_0,\ \sigma_1)$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$*-d-combine*$_{Pair}$*-Sync*:
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ [\![E]\!]\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_d\ \sigma_1 = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pair}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma_0,\ \sigma_1)$›
**if** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› **and** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$› **and** ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma_1$›
⟨*proof*⟩

**corollary** *P-d-combine*$_{Pair}$*-Sync*:
‹$P\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ [\![E]\!]\ P\langle\!\langle A_1\rangle\!\rangle_d\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pair}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma_0,\ \sigma_1)$›
**if** ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma_1$›
⟨*proof*⟩

## Pairlist

**theorem** $P_{SKIPS}$*-nd-combine*$_{Pairlist}$*-Sync* :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ [\![E]\!]\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma_1 = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\!\rangle\!\rangle_{nd}\ [\sigma_0,\ \sigma_1]$›
**if** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$› ‹*at-most-1-elem-term* $A_0$› ‹*at-most-1-elem-term* $A_1$›
⟨*proof*⟩

**corollary** *P-nd-combine*$_{Pairlist}$*-Sync* :
‹$P\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ [\![E]\!]\ P\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\!\rangle\!\rangle_{nd}\ [\sigma_0,\ \sigma_1]$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$*-d-combine*$_{Pairlist}$*-Sync* :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ [\![E]\!]\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_d\ \sigma_1 = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\!\rangle\!\rangle_d\ [\sigma_0,\ \sigma_1]$›
**if** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› **and** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$› **and** ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma_1$›
⟨*proof*⟩

**corollary** *P-d-combine*$_{Pairlist}$*-Sync* :
‹$P\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ [\![E]\!]\ P\langle\!\langle A_1\rangle\!\rangle_d\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\!\rangle\!\rangle_d\ [\sigma_0,\ \sigma_1]$›
**if** ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma_1$›
⟨*proof*⟩

### 6.2.2 Rlist

**theorem** $P_{SKIPS}$-*nd-combine*$_{Rlist}$-*Sync* :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ [\![E]\!]\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma_1 = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes[\![E]\!]_{Rlist}\ A_1\rangle\!\rangle\!\rangle_{nd}\ (\sigma_0\ \#\ \sigma_1)$›
**if** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$› ‹*at-most-1-elem-term* $A_0$› ‹*at-most-1-elem-term* $A_1$›
⟨*proof*⟩

**corollary** *P-nd-combine*$_{Rlist}$-*Sync* :
‹$P\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ [\![E]\!]\ P\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes[\![E]\!]_{Rlist}\ A_1\rangle\!\rangle\!\rangle_{nd}\ (\sigma_0\ \#\ \sigma_1)$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$-*d-combine*$_{Rlist}$-*Sync* :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{d}\ \sigma_0\ [\![E]\!]\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_{d}\ \sigma_1 = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{Rlist}\ A_1\rangle\!\rangle\!\rangle_{d}\ (\sigma_0\ \#\ \sigma_1)$›
**if** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› **and** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$› **and** ‹*indep-enabl* $A_0$ $\sigma_0$ $E$ $A_1$ $\sigma_1$›
⟨*proof*⟩

**corollary** *P-d-combine*$_{Rlist}$-*Sync* :
‹$P\langle\!\langle A_0\rangle\!\rangle_{d}\ \sigma_0\ [\![E]\!]\ P\langle\!\langle A_1\rangle\!\rangle_{d}\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{Rlist}\ A_1\rangle\!\rangle\!\rangle_{d}\ (\sigma_0\ \#\ \sigma_1)$›
**if** ‹*indep-enabl* $A_0$ $\sigma_0$ $E$ $A_1$ $\sigma_1$›
⟨*proof*⟩

### 6.2.3 ListslenL

**theorem** $P_{SKIPS}$-*nd-combine*$_{ListslenL}$-*Sync* :
**assumes** *same-length-reach0* : ‹$\bigwedge\sigma_0'.\ \sigma_0' \in \mathcal{R}_{nd}\ A_0\ \sigma_0 \Longrightarrow$ *length* $\sigma_0' = len_0$›
**and** $\varrho$-*disjoint*-$\varepsilon$ : ‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$› ‹*at-most-1-elem-term* $A_0$› ‹*at-most-1-elem-term* $A_1$›
**shows** ‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ [\![E]\!]\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma_1 = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle\!\rangle_{nd}\ (\sigma_0\ @\ \sigma_1)$›
⟨*proof*⟩

**corollary** *P-nd-combine*$_{ListslenL}$-*Sync* :
‹$P\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ [\![E]\!]\ P\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle\!\rangle_{nd}\ (\sigma_0\ @\ \sigma_1)$›
**if** ‹$\bigwedge\sigma_0'.\ \sigma_0' \in \mathcal{R}_{nd}\ A_0\ \sigma_0 \Longrightarrow$ *length* $\sigma_0' = len_0$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$-*d-combine*$_{ListslenL}$-*Sync* :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{d}\ \sigma_0\ [\![E]\!]\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_{d}\ \sigma_1 = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_{d}\otimes[\![len_0,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle\!\rangle_{d}\ (\sigma_0\ @\ \sigma_1)$›
**if** ‹$\bigwedge\sigma_0'.\ \sigma_0' \in \mathcal{R}_{d}\ A_0\ \sigma_0 \Longrightarrow$ *length* $\sigma_0' = len_0$›
‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$› ‹*indep-enabl* $A_0$ $\sigma_0$ $E$ $A_1$ $\sigma_1$›
⟨*proof*⟩

**corollary** *P-d-combine*$_{ListslenL}$-*Sync* :
‹$P\langle\!\langle A_0\rangle\!\rangle_{d}\ \sigma_0\ [\![E]\!]\ P\langle\!\langle A_1\rangle\!\rangle_{d}\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ {}_{d}\otimes[\![len_0,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle\!\rangle_{d}\ (\sigma_0\ @\ \sigma_1)$›
**if** ‹$\bigwedge\sigma_0'.\ \sigma_0' \in \mathcal{R}_{d}\ A_0\ \sigma_0 \Longrightarrow$ *length* $\sigma_0' = len_0$› ‹*indep-enabl* $A_0$ $\sigma_0$ $E$ $A_1$ $\sigma_1$›

⟨*proof*⟩

### 6.2.4 Multiple

**theorem** $P_{SKIPS}$*-nd-compactification-Sync*:
‹$[\![$*length* $\sigma s$ = *length As*; $\bigwedge A.\ A \in$ *set As* $\Longrightarrow$ $\varrho$*-disjoint-*$\varepsilon$ $A$;
$\bigwedge A.\ A \in$ *set As* $\Longrightarrow$ *at-most-1-elem-term* $A]\!]$
$\Longrightarrow$ $[\![E]\!]$ $(\sigma,\ A) \in\#$ *mset* (*zip* $\sigma s$ *As*). $P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma = P_{SKIPS}\langle\!\langle\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle\rangle\!\rangle_{nd}\ \sigma s$›
⟨*proof*⟩

**lemma** *P-nd-compactification-Sync*:
‹*length* $\sigma s$ = *length As* $\Longrightarrow$
$[\![E]\!]$ $(\sigma,\ A) \in\#$ *mset* (*zip* $\sigma s$ *As*). $P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma = P\langle\!\langle\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle\rangle\!\rangle_{nd}\ \sigma s$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-d-compactification-Sync*:
‹$[\![$*length* $\sigma s$ = *length As*; $\bigwedge A.\ A \in$ *set As* $\Longrightarrow$ $\varrho$*-disjoint-*$\varepsilon$ $A$;
$\bigwedge i\ j.\ [\![i <$ *length As*; $j <$ *length As*; $i \neq j]\!] \Longrightarrow$
*indep-enabl* (*As* ! $i$) ($\sigma s$ ! $i$) $E$ (*As* ! $j$) ($\sigma s$ ! $j$)$]\!] \Longrightarrow$
$[\![E]\!]$ $(\sigma,\ A) \in\#$ *mset* (*zip* $\sigma s$ *As*). $P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{d}\ \sigma = P_{SKIPS}\langle\!\langle\langle\!\langle_{d}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle\rangle\!\rangle_{d}$ $\sigma s$›
⟨*proof*⟩

**lemma** *P-d-compactification-Sync*:
‹$[\![$*length* $\sigma s$ = *length As*;
$\bigwedge i\ j.\ [\![i <$ *length As*; $j <$ *length As*; $i \neq j]\!] \Longrightarrow$
*indep-enabl* (*As* ! $i$) ($\sigma s$ ! $i$) $E$ (*As* ! $j$) ($\sigma s$ ! $j$)$]\!] \Longrightarrow$
$[\![E]\!]$ $(\sigma,\ A) \in\#$ *mset* (*zip* $\sigma s$ *As*). $P\langle\!\langle A\rangle\!\rangle_{d}\ \sigma = P\langle\!\langle\langle\!\langle_{d}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle\rangle\!\rangle_{d}\ \sigma s$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$*-nd-compactification-Sync-order-is-arbitrary*:
‹$P_{SKIPS}\langle\!\langle\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle\rangle\!\rangle_{nd}\ \sigma s = P_{SKIPS}\langle\!\langle\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *As'*$\rangle\!\rangle\rangle\!\rangle_{nd}\ \sigma s'$›
**if** ‹*length* $\sigma s$ = *length As*› ‹*length* $\sigma s'$ = *length As'*›
‹*mset* (*zip* $\sigma s$ *As*) = *mset* (*zip* $\sigma s'$ *As'*)›
‹$\bigwedge A.\ A \in$ *set As* $\Longrightarrow$ $\varrho$*-disjoint-*$\varepsilon$ $A$›
‹$\bigwedge A.\ A \in$ *set As* $\Longrightarrow$ *at-most-1-elem-term* $A$›
⟨*proof*⟩

**corollary** *P-nd-compactification-Sync-order-is-arbitrary*:
‹$P\langle\!\langle\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle\rangle\!\rangle_{nd}\ \sigma s = P\langle\!\langle\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *As'*$\rangle\!\rangle\rangle\!\rangle_{nd}\ \sigma s'$›
**if** ‹*length* $\sigma s$ = *length As*› ‹*length* $\sigma s'$ = *length As'*›
‹*mset* (*zip* $\sigma s$ *As*) = *mset* (*zip* $\sigma s'$ *As'*)›
⟨*proof*⟩

**corollary** $P_{SKIPS}$*-d-compactification-Sync-order-is-arbitrary*:

‹$P_{SKIPS}\langle\langle\langle_d\bigotimes [\![E]\!]\ As\rangle\rangle\rangle_d\ \sigma s = P_{SKIPS}\langle\langle\langle_d\bigotimes [\![E]\!]\ As'\rangle\rangle\rangle_d\ \sigma s'$›
**if** ‹*length* $\sigma s$ = *length As*› ‹*length* $\sigma s'$ = *length As'*›
‹*mset* (*zip* $\sigma s$ *As*) = *mset* (*zip* $\sigma s'$ *As'*)›
‹$\bigwedge A.\ A \in$ *set As* $\Longrightarrow$ $\varrho$-*disjoint*-$\varepsilon$ *A*›
‹$\bigwedge i\ j.\ [\![i <$ *length As*; $j <$ *length As*; $i \neq j]\!] \Longrightarrow$
*indep-enabl* (*As* ! *i*) ($\sigma s$ ! *i*) *E* (*As* ! *j*) ($\sigma s$ ! *j*)›
⟨*proof*⟩

**corollary** *P-d-compactification-Sync-order-is-arbitrary*:
‹$P\langle\langle\langle_d\bigotimes [\![E]\!]\ As\rangle\rangle\rangle_d\ \sigma s = P\langle\langle\langle_d\bigotimes [\![E]\!]\ As'\rangle\rangle\rangle_d\ \sigma s'$›
**if** ‹*length* $\sigma s$ = *length As*› ‹*length* $\sigma s'$ = *length As'*›
‹*mset* (*zip* $\sigma s$ *As*) = *mset* (*zip* $\sigma s'$ *As'*)›
‹$\bigwedge i\ j.\ [\![i <$ *length As*; $j <$ *length As*; $i \neq j]\!] \Longrightarrow$
*indep-enabl* (*As* ! *i*) ($\sigma s$ ! *i*) *E* (*As* ! *j*) ($\sigma s$ ! *j*)›
⟨*proof*⟩

## 6.3 Derived Versions

**lemma** $P_{SKIPS}$*-nd-compactification-Sync-upt-version*:
‹$[\![E]\!]\ P \in\#$ *mset* (*map Q* [*0*..<*n*]). $P = P_{SKIPS}\langle\langle\langle_{nd}\bigotimes [\![E]\!]$ *map A* [*0*..<*n*]$\rangle\rangle\rangle_{nd}$ (*replicate n 0*)›
**if** ‹$\bigwedge i.\ i < n \Longrightarrow$ $\varrho$-*disjoint*-$\varepsilon$ (*A i*)›
‹$\bigwedge i.\ i < n \Longrightarrow$ *at-most-1-elem-term* (*A i*)›
‹$\bigwedge i.\ i < n \Longrightarrow P_{SKIPS}\langle A\ i\rangle_{nd}\ 0 = Q\ i$›
⟨*proof*⟩

**lemma** *P-nd-compactification-Sync-upt-version*:
‹$[\![E]\!]\ P \in\#$ *mset* (*map Q* [*0*..<*n*]). $P = P\langle\langle\langle_{nd}\bigotimes [\![E]\!]$ *map A* [*0*..<*n*]$\rangle\rangle\rangle_{nd}$ (*replicate n 0*)›
**if** ‹$\bigwedge i.\ i < n \Longrightarrow P\langle A\ i\rangle_{nd}\ 0 = Q\ i$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-d-compactification-Sync-upt-version*:
‹$[\![E]\!]\ P \in\#$ *mset* (*map Q* [*0*..<*n*]). $P = P_{SKIPS}\langle\langle\langle_d\bigotimes [\![E]\!]$ *map A* [*0*..<*n*]$\rangle\rangle\rangle_d$ (*replicate n 0*)›
**if** ‹$\bigwedge i.\ i < n \Longrightarrow$ $\varrho$-*disjoint*-$\varepsilon$ (*A i*)›
‹$\bigwedge i\ j.\ i < n \Longrightarrow j < n \Longrightarrow i \neq j \Longrightarrow$ *indep-enabl* (*A i*) *0 E* (*A j*) *0*›
‹$\bigwedge i.\ i < n \Longrightarrow P_{SKIPS}\langle A\ i\rangle_d\ 0 = Q\ i$›
⟨*proof*⟩

**lemma** *P-d-compactification-Sync-upt-version*:
‹$[\![E]\!]\ P \in\#$ *mset* (*map Q* [*0*..<*n*]). $P = P\langle\langle\langle_d\bigotimes [\![E]\!]$ *map A* [*0*..<*n*]$\rangle\rangle\rangle_d$ (*replicate n 0*)›
**if** ‹$\bigwedge i.\ i < n \Longrightarrow P\langle A\ i\rangle_d\ 0 = Q\ i$›
‹$\bigwedge i\ j.\ i < n \Longrightarrow j < n \Longrightarrow i \neq j \Longrightarrow$ *indep-enabl* (*A i*) *0 E* (*A j*) *0*›
⟨*proof*⟩

## 6.4 More on Iterated Combine

**lemma** *ε-iterated-combine$_{nd}$-Sync-general-form*:
‹*length σs = length As* $\Longrightarrow$ *ε* $\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle$ *σs* =
(*if As* = [] *then* {}
*else* ($\bigcup$ *i* < *length As*. *ε* (*As* ! *i*) (*σs* ! *i*)) − *E* ∪ ($\bigcap$ *i* < *length As*. *ε* (*As* ! *i*) (*σs* ! *i*)))›
**for** *As* :: ‹('*σ*, '*e*, '*r*) $A_{nd}$ *list*›
⟨*proof*⟩

**lemma** *ε-iterated-combine$_{d}$-Sync-general-form*:
‹*length σs = length As* $\Longrightarrow$ *ε* $\langle\!\langle_{d}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle$ *σs* =
(*if As* = [] *then* {}
*else* ($\bigcup$ *i* < *length As*. *ε* (*As* ! *i*) (*σs* ! *i*)) − *E* ∪ ($\bigcap$ *i* < *length As*. *ε* (*As* ! *i*) (*σs* ! *i*)))›
**for** *As* :: ‹('*σ*, '*e*, '*r*) $A_{d}$ *list*›
⟨*proof*⟩

**lemma** *τ-iterated-combine$_{nd}$-Sync-general-form*:
‹[\![*length σs = length As*; *σs'* ∈ *τ* $\langle\!\langle_{nd}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle$ *σs a*; *i* < *length As*]\!] $\Longrightarrow$
*σs'* ! *i* ∈ *insert* (*σs* ! *i*) (*τ* (*As* ! *i*) (*σs* ! *i*) *a*)›
⟨*proof*⟩

**lemma** *τ-iterated-combine$_{d}$-Sync-general-form*:
‹*length σs = length As* $\Longrightarrow$
*τ* $\langle\!\langle_{d}\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle$ *σs a* =
(*if As* = [] *then* ◊ *else*
*if a* ∈ ($\bigcup$ *i* < *length As*. *ε* (*As* ! *i*) (*σs* ! *i*)) − *E* ∪ ($\bigcap$ *i* < *length As*. *ε* (*As* ! *i*) (*σs* ! *i*))
*then* ⌊*map2* (λ*σ A*. *if a* ∈ *ε A σ then* ⌈*τ A σ a*⌉ *else σ*) *σs As*⌋ *else* ◊)›
**for** *As* :: ‹('*σ*, '*e*, '*r*) $A_{d}$ *list*›
⟨*proof*⟩

**lemma** *indep-implies-only-one-enabled'*:
‹∃!*i*. *i* < *length As* ∧ *a* ∈ *ε* (*As* ! *i*) (*σs* ! *i*)›
**if** ‹*length σs = length As*›
**and** ‹$\bigwedge$ *i j*. [\![*i* < *length As*; *j* < *length As*; *i* ≠ *j*]\!] $\Longrightarrow$
*ε* (*As* ! *i*) (*σs* ! *i*) ∩ *ε* (*As* ! *j*) (*σs* ! *j*) ⊆ *E*›
**and** ‹*a* ∈ ($\bigcup$ *i*<*length As*. *ε* (*As* ! *i*) (*σs* ! *i*)) − *E*›
⟨*proof*⟩

**lemma** *indep-implies-only-one-enabled*:
‹[\![*length σs = length As*;
$\bigwedge$ *i j*. [\![*i* < *length As*; *j* < *length As*; *i* ≠ *j*]\!] $\Longrightarrow$
*indep-enabl* (*As* ! *i*) (*σs* ! *i*) *E* (*As* ! *j*) (*σs* ! *j*);
*a* ∈ ($\bigcup$ *i*<*length As*. *ε* (*As* ! *i*) (*σs* ! *i*)) − *E*]\!] $\Longrightarrow$

$\exists! i.\ i < \mathit{length}\ \mathit{As} \wedge a \in \varepsilon\ (\mathit{As}\ !\ i)\ (\sigma s\ !\ i)$›
⟨*proof*⟩

**lemma** *τ-iterated-combine$_d$-Sync-general-form-when-indep*:
‹$\tau\ \langle\!\langle_d\bigotimes [\![E]\!]\ \mathit{As}\rangle\!\rangle\ \sigma s\ a =$
$(\mathit{if}\ \mathit{As} = []\ \mathit{then}\ \Diamond$
$\mathit{else}\quad \mathit{if}\ a \in (\bigcap i{<}\mathit{length}\ \mathit{As}.\ \varepsilon\ (\mathit{As}\ !\ i)\ (\sigma s\ !\ i))$
$\mathit{then}\ \lfloor \mathit{map2}\ (\lambda\sigma\ A.\ \lceil \tau\ A\ \sigma\ a\rceil)\ \sigma s\ \mathit{As}\rfloor$
$\mathit{else}\quad \mathit{if}\ a \in (\bigcup i{<}\mathit{length}\ \mathit{As}.\ \varepsilon\ (\mathit{As}\ !\ i)\ (\sigma s\ !\ i)) - E$
$\mathit{then}\ \mathit{let}\ i = \mathit{THE}\ i.\ i < \mathit{length}\ \mathit{As} \wedge\ a \in \varepsilon\ (\mathit{As}\ !\ i)\ (\sigma s\ !\ i)$
$\mathit{in}\ \lfloor \sigma s[i := \lceil \tau\ (\mathit{As}\ !\ i)\ (\sigma s\ !\ i)\ a\rceil]\rfloor$
$\mathit{else}\ \Diamond)$›
(**is** ‹- = ($\mathit{if}\ \mathit{As} = []\ \mathit{then}\ \Diamond\ \mathit{else}$
$\mathit{if}\ a \in ?I\ \mathit{As}\ \sigma s\ \mathit{then}\ \lfloor \mathit{map2}\ (\lambda\sigma\ A.\ \lceil \tau\ A\ \sigma\ a\rceil)\ \sigma s\ \mathit{As}\rfloor\ \mathit{else}$
$\mathit{if}\ a \in ?U\ \mathit{As}\ \sigma s - E\ \mathit{then}\ ?\mathit{upd}\ \mathit{As}\ \sigma s\ \mathit{else}\ \Diamond$)›)
**if** ‹$\mathit{length}\ \sigma s = \mathit{length}\ \mathit{As}$›
‹$\bigwedge i\ j.\ [\![i < \mathit{length}\ \mathit{As};\ j < \mathit{length}\ \mathit{As};\ i \neq j]\!] \Longrightarrow$
$\mathit{indep\text{-}enabl}\ (\mathit{As}\ !\ i)\ (\sigma s\ !\ i)\ E\ (\mathit{As}\ !\ j)\ (\sigma s\ !\ j)$›
⟨*proof*⟩

## 6.5 More on Events

**lemma** *events-of-MultiSync-P$_{SKIPS}$-nd* :
‹$\alpha([\![E]\!]\ (\sigma,\ A) \in\#\ \mathit{mset}\ (\mathit{zip}\ \sigma s\ \mathit{As}).\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma) =$
$(\mathit{if}\ \mathit{As} = []\ \mathit{then}\ \{\}\ \mathit{else}$
$\bigcup\ \sigma s' \in \mathcal{R}_{nd}\ \langle\!\langle_{nd}\bigotimes [\![E]\!]\ \mathit{As}\rangle\!\rangle\ \sigma s.\ (\bigcup i{<}\mathit{length}\ \mathit{As}.\ \varepsilon\ (\mathit{As}\ !\ i)\ (\sigma s'\ !\ i)) - E \cup$
$(\bigcap i{<}\mathit{length}\ \mathit{As}.\ \varepsilon\ (\mathit{As}\ !\ i)\ (\sigma s'\ !\ i)))$›
(**is** ‹- = ?rhs›) **if** ‹$\mathit{length}\ \sigma s = \mathit{length}\ \mathit{As}$›
‹$\bigwedge A.\ A \in \mathit{set}\ \mathit{As} \Longrightarrow \varrho\text{-}\mathit{disjoint}\text{-}\varepsilon\ A$› ‹$\bigwedge A.\ A \in \mathit{set}\ \mathit{As} \Longrightarrow \mathit{at\text{-}most\text{-}1\text{-}elem\text{-}term}$
$A$›
⟨*proof*⟩

**lemma** *events-of-MultiSync-P-nd* :
‹$\alpha([\![E]\!]\ (\sigma,\ A) \in\#\ \mathit{mset}\ (\mathit{zip}\ \sigma s\ \mathit{As}).\ P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma) =$
$(\mathit{if}\ \mathit{As} = []\ \mathit{then}\ \{\}\ \mathit{else}$
$\bigcup\ \sigma s' \in \mathcal{R}_{nd}\ \langle\!\langle_{nd}\bigotimes [\![E]\!]\ \mathit{As}\rangle\!\rangle\ \sigma s.\ (\bigcup i{<}\mathit{length}\ \mathit{As}.\ \varepsilon\ (\mathit{As}\ !\ i)\ (\sigma s'\ !\ i)) - E \cup$
$(\bigcap i{<}\mathit{length}\ \mathit{As}.\ \varepsilon\ (\mathit{As}\ !\ i)\ (\sigma s'\ !\ i)))$›
(**is** ‹- = ?rhs›) **if** ‹$\mathit{length}\ \sigma s = \mathit{length}\ \mathit{As}$›
⟨*proof*⟩

**lemma** *events-of-MultiSync-P$_{SKIPS}$-d* :
‹$\alpha([\![E]\!]\ (\sigma,\ A) \in\#\ \mathit{mset}\ (\mathit{zip}\ \sigma s\ \mathit{As}).\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{d}\ \sigma) =$
$(\mathit{if}\ \mathit{As} = []\ \mathit{then}\ \{\}\ \mathit{else}$
$\bigcup\ \sigma s' \in \mathcal{R}_{d}\ \langle\!\langle_{d}\bigotimes [\![E]\!]\ \mathit{As}\rangle\!\rangle\ \sigma s.\ (\bigcup i{<}\mathit{length}\ \mathit{As}.\ \varepsilon\ (\mathit{As}\ !\ i)\ (\sigma s'\ !\ i) - E \cup$
$(\bigcap i{<}\mathit{length}\ \mathit{As}.\ \varepsilon\ (\mathit{As}\ !\ i)\ (\sigma s'\ !\ i))))$›

(**is** ‹- = *?rhs*›) **if** ‹*length* $\sigma s$ = *length As*› ‹$\bigwedge A.\ A \in$ *set As* $\Longrightarrow$ $\varrho$-*disjoint*-$\varepsilon$ $A$›
‹$\bigwedge i\ j.\ [\![i <$ *length As*; $j <$ *length As*; $i \neq j]\!] \Longrightarrow$
*indep-enabl* (*As* ! $i$) ($\sigma s$ ! $i$) $E$ (*As* ! $j$) ($\sigma s$ ! $j$)›
⟨*proof*⟩

**lemma** *events-of-MultiSync-P-d* :
‹$\alpha([\![E]\!]\ (\sigma,\ A) \in\#$ *mset* (*zip* $\sigma s$ *As*). $P\langle\!\langle A\rangle\!\rangle_d\ \sigma) =$
(*if As* = [] *then* {} *else*
$\bigcup\ \sigma s' \in \mathcal{R}_d\ \langle\!\langle_d\bigotimes [\![E]\!]$ *As*$\rangle\!\rangle\ \sigma s.\ (\bigcup i<$*length As*. $\varepsilon$ (*As* ! $i$) ($\sigma s'$ ! $i$) $-\ E \cup$
$(\bigcap i<$*length As*. $\varepsilon$ (*As* ! $i$) ($\sigma s'$ ! $i$))))›
(**is** ‹- = *?rhs*›) **if** ‹*length* $\sigma s$ = *length As*›
**and** ‹$\bigwedge i\ j.\ [\![i <$ *length As*; $j <$ *length As*; $i \neq j]\!] \Longrightarrow$
*indep-enabl* (*As* ! $i$) ($\sigma s$ ! $i$) $E$ (*As* ! $j$) ($\sigma s$ ! $j$)›
⟨*proof*⟩

# Chapter 7

# Combining Automata for Generalized Synchronization Product

## 7.1 Definitions

### 7.1.1 Specializations

**definition** $combine_{dPairlist}$-$Sync_{ptick}$ ::
‹[('σ, 'e, 'r, 'α) $A_d$-scheme, 'e set, ('σ, 'e, 'r, 'β) $A_d$-scheme] ⇒ ('σ list, 'e, 'r list) $A_d$›
**where** ‹$combine_{dPairlist}$-$Sync_{ptick}$ $A_0$ $E$ $A_1$ ≡
$combine_d$-Sync $A_0$ $E$ $A_1$ hd (λσs. hd (tl σs)) (λs t. [s, t]) (λs t. ⌊[s, t]⌋)›
**definition** $combine_{ndPairlist}$-$Sync_{ptick}$ ::
‹[('σ, 'e, 'r, 'α) $A_{nd}$-scheme, 'e set, ('σ, 'e, 'r, 'β) $A_{nd}$-scheme] ⇒ ('σ list, 'e, 'r list) $A_{nd}$›
**where** ‹$combine_{ndPairlist}$-$Sync_{ptick}$ $A_0$ $E$ $A_1$ ≡ $combine_{nd}$-Sync $A_0$ $E$ $A_1$ hd (λσs. hd (tl σs)) (λs t. [s, t]) (λs t. ⌊[s, t]⌋)›

**definition** $combine_{dPair}$-$Sync_{ptick}$ ::
‹[('$\sigma_0$, 'e, '$r_0$, 'α) $A_d$-scheme, 'e set, ('$\sigma_1$, 'e, '$r_1$, 'β) $A_d$-scheme] ⇒ ('$\sigma_0$ × '$\sigma_1$, 'e, '$r_0$ × '$r_1$) $A_d$›
**where** ‹$combine_{dPair}$-$Sync_{ptick}$ $A_0$ $E$ $A_1$ ≡ $combine_d$-Sync $A_0$ $E$ $A_1$ fst snd Pair (λs r. ⌊(s, r)⌋)›
**definition** $combine_{ndPair}$-$Sync_{ptick}$ ::
‹[('$\sigma_0$, 'e, '$r_0$, 'α) $A_{nd}$-scheme, 'e set, ('$\sigma_1$, 'e, '$r_1$, 'β) $A_{nd}$-scheme] ⇒ ('$\sigma_0$ × '$\sigma_1$, 'e, '$r_0$ × '$r_1$) $A_{nd}$›
**where** ‹$combine_{ndPair}$-$Sync_{ptick}$ $A_0$ $E$ $A_1$ ≡ $combine_{nd}$-Sync $A_0$ $E$ $A_1$ fst snd Pair (λs r. ⌊(s, r)⌋)›

**definition** $combine_{dListslenL}$-$Sync_{ptick}$ ::
‹[('σ list, 'e, 'r list, 'α) $A_d$-scheme, nat, 'e set, ('σ list, 'e, 'r list, 'β) $A_d$-scheme] ⇒ ('σ list, 'e, 'r list) $A_d$›

**where** ‹$combine_{dListslenL}$-$Sync_{ptick}$ $A_0$ $len_0$ $E$ $A_1$ ≡ $combine_d$-$Sync$ $A_0$ $E$ $A_1$ ($take$ $len_0$) ($drop$ $len_0$) (@) ($\lambda s$ $r$. $\lfloor s$ @ $r\rfloor$)›
**definition** $combine_{ndListslenL}$-$Sync_{ptick}$ ::
‹[('$\sigma$ $list$, '$e$, '$r$ $list$, '$\alpha$) $A_{nd}$-$scheme$, $nat$, '$e$ $set$, ('$\sigma$ $list$, '$e$, '$r$ $list$, '$\beta$) $A_{nd}$-$scheme$] ⇒ ('$\sigma$ $list$, '$e$, '$r$ $list$) $A_{nd}$›
**where** ‹$combine_{ndListslenL}$-$Sync_{ptick}$ $A_0$ $len_0$ $E$ $A_1$ ≡ $combine_{nd}$-$Sync$ $A_0$ $E$ $A_1$ ($take$ $len_0$) ($drop$ $len_0$) (@) ($\lambda s$ $r$. $\lfloor s$ @ $r\rfloor$)›

**definition** $combine_{dRlist}$-$Sync_{ptick}$ ::
‹[('$\sigma$, '$e$, '$r$, '$\alpha$) $A_d$-$scheme$, '$e$ $set$, ('$\sigma$ $list$, '$e$, '$r$ $list$, '$\beta$) $A_d$-$scheme$] ⇒ ('$\sigma$ $list$, '$e$, '$r$ $list$) $A_d$›
**where** ‹$combine_{dRlist}$-$Sync_{ptick}$ $A_0$ $E$ $A_1$ ≡ $combine_d$-$Sync$ $A_0$ $E$ $A_1$ $hd$ $tl$ (#) ($\lambda s$ $r$. $\lfloor s$ # $r\rfloor$)›
**definition** $combine_{ndRlist}$-$Sync_{ptick}$ ::
‹[('$\sigma$, '$e$, '$r$, '$\alpha$) $A_{nd}$-$scheme$, '$e$ $set$, ('$\sigma$ $list$, '$e$, '$r$ $list$, '$\beta$) $A_{nd}$-$scheme$] ⇒ ('$\sigma$ $list$, '$e$, '$r$ $list$) $A_{nd}$›
**where** ‹$combine_{ndRlist}$-$Sync_{ptick}$ $A_0$ $E$ $A_1$ ≡ $combine_{nd}$-$Sync$ $A_0$ $E$ $A_1$ $hd$ $tl$ (#) ($\lambda s$ $r$. $\lfloor s$ # $r\rfloor$)›

**lemmas** $combine_{Pairlist}$-$Sync_{ptick}$-$defs$ = $combine_{dPairlist}$-$Sync_{ptick}$-$def$ $combine_{ndPairlist}$-$Sync_{ptick}$-
**and** $combine_{Pair}$-$Sync_{ptick}$-$defs$ = $combine_{dPair}$-$Sync_{ptick}$-$def$ $combine_{ndPair}$-$Sync_{ptick}$-$def$
**and** $combine_{ListslenL}$-$Sync_{ptick}$-$defs$ = $combine_{dListslenL}$-$Sync_{ptick}$-$def$ $combine_{ndListslenL}$-$Sync_{ptick}$-$def$
**and** $combine_{Rlist}$-$Sync_{ptick}$-$defs$ = $combine_{dRlist}$-$Sync_{ptick}$-$def$ $combine_{ndRlist}$-$Sync_{ptick}$-$def$

**lemmas** $combine$-$Sync_{ptick}$-$defs$ =
$combine_{Pairlist}$-$Sync_{ptick}$-$defs$ $combine_{Pair}$-$Sync_{ptick}$-$defs$ $combine_{ListslenL}$-$Sync_{ptick}$-$defs$ $combine_{Rlist}$-$Sync_{ptick}$-$defs$

**bundle** $combine_{nd}$-$Sync_{ptick}$-$syntax$ **begin**

**notation** $combine_{dPairlist}$-$Sync_{ptick}$ (‹⟪- $_d$⊗⟦-⟧$_{✓Pairlist}$ -⟫› [0, 0, 0])
**notation** $combine_{ndPairlist}$-$Sync_{ptick}$ (‹⟪- $_{nd}$⊗⟦-⟧$_{✓Pairlist}$ -⟫› [0, 0, 0])
**notation** $combine_{dPair}$-$Sync_{ptick}$ (‹⟪- $_d$⊗⟦-⟧$_{✓Pair}$ -⟫› [0, 0, 0])
**notation** $combine_{ndPair}$-$Sync_{ptick}$ (‹⟪- $_{nd}$⊗⟦-⟧$_{✓Pair}$ -⟫› [0, 0, 0])
**notation** $combine_{dListslenL}$-$Sync_{ptick}$ (‹⟪- $_d$⊗⟦-, -⟧$_{✓ListslenL}$ -⟫› [0, 0, 0, 0])
**notation** $combine_{ndListslenL}$-$Sync_{ptick}$ (‹⟪- $_{nd}$⊗⟦-, -⟧$_{✓ListslenL}$ -⟫› [0, 0, 0, 0])
**notation** $combine_{dRlist}$-$Sync_{ptick}$ (‹⟪- $_d$⊗⟦-⟧$_{✓Rlist}$ -⟫› [0, 0, 0])
**notation** $combine_{ndRlist}$-$Sync_{ptick}$ (‹⟪- $_{nd}$⊗⟦-⟧$_{✓Rlist}$ -⟫› [0, 0, 0])

**end**

**unbundle** $combine_{nd}$-$Sync_{ptick}$-$syntax$

## 7.2 First Properties

**lemma** $finite$-$trans$-$combine_{nd}$-$Sync_{ptick}$-$simps$ [$simp$] :
‹$finite$-$trans$ $A_0$ ⟹ $finite$-$trans$ $A_1$ ⟹ $finite$-$trans$ ⟪$A_0$ $_{nd}$⊗⟦$E$⟧$_{✓Pairlist}$ $A_1$⟫›

‹*finite-trans* $B_0 \Longrightarrow$ *finite-trans* $B_1 \Longrightarrow$ *finite-trans* $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pair}\ B_1\rangle\!\rangle$›
‹*finite-trans* $C_0 \Longrightarrow$ *finite-trans* $C_1 \Longrightarrow$ *finite-trans* $\langle\!\langle C_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ C_1\rangle\!\rangle$›
‹*finite-trans* $D_0 \Longrightarrow$ *finite-trans* $D_1 \Longrightarrow$ *finite-trans* $\langle\!\langle D_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}\ D_1\rangle\!\rangle$›
⟨*proof*⟩

**lemma** $\varepsilon$-*combine*$_{Pairlist}$-*Sync*$_{ptick}$:
‹$\varepsilon\ \langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle\ \sigma s =$ *combine-Sync-*$\varepsilon\ A_0\ E\ A_1\ hd\ (hd \circ tl)\ \sigma s$›
‹$\varepsilon\ \langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pairlist}\ B_1\rangle\!\rangle\ \sigma s =$ *combine-Sync-*$\varepsilon\ B_0\ E\ B_1\ hd\ (hd \circ tl)\ \sigma s$›
⟨*proof*⟩

**lemma** $\varepsilon$-*combine*$_{Pair}$-*Sync*$_{ptick}$:
‹$\varepsilon\ \langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle\ \sigma s =$ *combine-Sync-*$\varepsilon\ A_0\ E\ A_1\ fst\ snd\ \sigma s$›
‹$\varepsilon\ \langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pair}\ B_1\rangle\!\rangle\ \sigma s =$ *combine-Sync-*$\varepsilon\ B_0\ E\ B_1\ fst\ snd\ \sigma s$›
⟨*proof*⟩

**lemma** $\varepsilon$-*combine*$_{ListslenL}$-*Sync*$_{ptick}$:
‹$\varepsilon\ \langle\!\langle A_0\ {}_{d}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle\ \sigma s =$ *combine-Sync-*$\varepsilon\ A_0\ E\ A_1\ (take\ len_0)\ (drop\ len_0)\ \sigma s$›
‹$\varepsilon\ \langle\!\langle B_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ B_1\rangle\!\rangle\ \sigma s =$ *combine-Sync-*$\varepsilon\ B_0\ E\ B_1\ (take\ len_0)\ (drop\ len_0)\ \sigma s$›
⟨*proof*⟩

**lemma** $\varepsilon$-*combine*$_{Rlist}$-*Sync*$_{ptick}$:
‹$\varepsilon\ \langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle\ \sigma s =$ *combine-Sync-*$\varepsilon\ A_0\ E\ A_1\ hd\ tl\ \sigma s$›
‹$\varepsilon\ \langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}\ B_1\rangle\!\rangle\ \sigma s =$ *combine-Sync-*$\varepsilon\ B_0\ E\ B_1\ hd\ tl\ \sigma s$›
⟨*proof*⟩

**lemma** $\varrho$-*combine*$_{Pairlist}$-*Sync*$_{ptick}$:
‹$\varrho\ \langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle = \{\sigma s.\ hd\ \sigma s \in \varrho\ A_0 \wedge hd\ (tl\ \sigma s) \in \varrho\ A_1\}$›
‹$\varrho\ \langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pairlist}\ B_1\rangle\!\rangle = \{\sigma s.\ hd\ \sigma s \in \varrho\ B_0 \wedge hd\ (tl\ \sigma s) \in \varrho\ B_1\}$›
⟨*proof*⟩

**lemma** $\varrho$-*combine*$_{Pair}$-*Sync*$_{ptick}$:
‹$\varrho\ \langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle = \{(\sigma_0,\ \sigma_1).\ \sigma_0 \in \varrho\ A_0 \wedge \sigma_1 \in \varrho\ A_1\}$›
‹$\varrho\ \langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pair}\ B_1\rangle\!\rangle = \{(\sigma_0,\ \sigma_1).\ \sigma_0 \in \varrho\ B_0 \wedge \sigma_1 \in \varrho\ B_1\}$›
⟨*proof*⟩

**lemma** $\varrho$-*combine*$_{ListslenL}$-*Sync*$_{ptick}$:
‹$\varrho\ \langle\!\langle A_0\ {}_{d}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle = \{\sigma s.\ take\ len_0\ \sigma s \in \varrho\ A_0 \wedge drop\ len_0\ \sigma s \in \varrho\ A_1\}$›
‹$\varrho\ \langle\!\langle B_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ B_1\rangle\!\rangle = \{\sigma s.\ take\ len_0\ \sigma s \in \varrho\ B_0 \wedge drop\ len_0\ \sigma s \in \varrho\ B_1\}$›
⟨*proof*⟩

**lemma** $\varrho$-*combine*$_{Rlist}$-*Sync*$_{ptick}$:
‹$\varrho\ \langle\!\langle A_0\ {}_{d}\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle = \{\sigma s.\ hd\ \sigma s \in \varrho\ A_0 \wedge tl\ \sigma s \in \varrho\ A_1\}$›
‹$\varrho\ \langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}\ B_1\rangle\!\rangle = \{\sigma s.\ hd\ \sigma s \in \varrho\ B_0 \wedge tl\ \sigma s \in \varrho\ B_1\}$›

⟨*proof*⟩

## 7.3 Transitions are unchanged in the Generalization

In the generalization, only the $\omega$ function is modified.

**lemma** $\tau$-*combine*$_{Pairlist}$-*Sync*$_{ptick}$ :
‹$\tau$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle = \tau\ \langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pairlist}\ A_1\rangle\!\rangle$›
‹$\tau$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pairlist}\ B_1\rangle\!\rangle = \tau\ \langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Pairlist}\ B_1\rangle\!\rangle$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{Pair}$-*Sync*$_{ptick}$ :
‹$\tau$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle = \tau\ \langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{Pair}\ A_1\rangle\!\rangle$›
‹$\tau$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pair}\ B_1\rangle\!\rangle = \tau\ \langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{Pair}\ B_1\rangle\!\rangle$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{ListslenL}$-*Sync*$_{ptick}$ :
‹$\tau$ $\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle = \tau\ \langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{ListslenL}\ A_1\rangle\!\rangle$›
‹$\tau$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ B_1\rangle\!\rangle = \tau\ \langle\!\langle B_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{ListslenL}\ B_1\rangle\!\rangle$›
⟨*proof*⟩

$\tau$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle$ and $\tau$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}\ B_1\rangle\!\rangle$ cannot be obtained that easily because of the types of terminations.

## 7.4 Reachability

**lemma** $\mathcal{R}_d$-*combine*$_{dListslenL}$-*Sync*$_{ptick}$-*subset*:
‹$\mathcal{R}_d$ $\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle$ $(s_0\ @\ s_1) \subseteq \{t_0\ @\ t_1 |\ t_0\ t_1.\ t_0 \in \mathcal{R}_d\ A_0\ s_0 \wedge t_1 \in \mathcal{R}_d\ A_1\ s_1\}$› (**is** ‹$?S_A \subseteq$ -›)
**if** ‹$\bigwedge t_0.\ t_0 \in \mathcal{R}_d\ A_0\ s_0 \Longrightarrow length\ t_0 = len_0$›
⟨*proof*⟩

**lemma** $\mathcal{R}_{nd}$-*combine*$_{ndListslenL}$-*Sync*$_{ptick}$-*subset*:
‹$\mathcal{R}_{nd}$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ B_1\rangle\!\rangle$ $(s_0\ @\ s_1) \subseteq \{t_0\ @\ t_1 |\ t_0\ t_1.\ t_0 \in \mathcal{R}_{nd}\ B_0\ s_0 \wedge t_1 \in \mathcal{R}_{nd}\ B_1\ s_1\}$› (**is** ‹$?S_B \subseteq$ -›)
**if** ‹$\bigwedge t_0.\ t_0 \in \mathcal{R}_{nd}\ B_0\ s_0 \Longrightarrow length\ t_0 = len_0$›
⟨*proof*⟩

**lemma** $\mathcal{R}_d$-*combine*$_{dPairlist}$-*Sync*$_{ptick}$-*subset*:
‹$\mathcal{R}_d$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle$ $[s_0,\ s_1] \subseteq \{[t_0,\ t_1] |\ t_0\ t_1.\ t_0 \in \mathcal{R}_d\ A_0\ s_0 \wedge t_1 \in \mathcal{R}_d\ A_1\ s_1\}$› (**is** ‹$?S_A \subseteq$ -›)
**and** $\mathcal{R}_{nd}$-*combine*$_{ndPairlist}$-*Sync*$_{ptick}$-*subset*:
‹$\mathcal{R}_{nd}$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pairlist}\ B_1\rangle\!\rangle$ $[s_0,\ s_1] \subseteq \{[t_0,\ t_1] |\ t_0\ t_1.\ t_0 \in \mathcal{R}_{nd}\ B_0\ s_0 \wedge t_1 \in \mathcal{R}_{nd}\ B_1\ s_1\}$› (**is** ‹$?S_B \subseteq$ -›)
⟨*proof*⟩

**lemma** $\mathcal{R}_d$-*combine*$_{dPair}$-*Sync*$_{ptick}$-*subset*:
‹$\mathcal{R}_d$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle$ $(s_0,\ s_1) \subseteq \mathcal{R}_d\ A_0\ s_0 \times \mathcal{R}_d\ A_1\ s_1$› (**is** ‹$?S_A \subseteq$ -›)
**and** $\mathcal{R}_{nd}$-*combine*$_{ndPair}$-*Sync*$_{ptick}$-*subset*:
‹$\mathcal{R}_{nd}$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pair}\ B_1\rangle\!\rangle$ $(s_0,\ s_1) \subseteq \mathcal{R}_{nd}\ B_0\ s_0 \times \mathcal{R}_{nd}\ B_1\ s_1$› (**is** ‹$?S_B \subseteq$ -›)
⟨*proof*⟩

**lemma** $\mathcal{R}_d$-*combine*$_{dRlist}$-*Sync*$_{ptick}$-*subset*:
‹$\mathcal{R}_d$ $\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle$ $(s_0\ \#\ \sigma s) \subseteq \{t_0\ \#\ \sigma t|\ t_0\ \sigma t.\ t_0 \in \mathcal{R}_d\ A_0\ s_0 \wedge \sigma t \in \mathcal{R}_d\ A_1\ \sigma s\}$› (**is** ‹$?S_A \subseteq$ -›)
**and** $\mathcal{R}_{nd}$-*combine*$_{ndRlist}$-*Sync*$_{ptick}$-*subset*:
‹$\mathcal{R}_{nd}$ $\langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}\ B_1\rangle\!\rangle$ $(s_0\ \#\ \sigma s) \subseteq \{t_0\ \#\ \sigma t|\ t_0\ \sigma t.\ t_0 \in \mathcal{R}_{nd}\ B_0\ s_0 \wedge \sigma t \in \mathcal{R}_{nd}\ B_1\ \sigma s\}$› (**is** ‹$?S_B \subseteq$ -›)
⟨*proof*⟩

## 7.5 Normalization

**lemma** $\omega$-*combine*$_{Pairlist}$-*Sync*$_{ptick}$-*behaviour*:
‹$\omega$ $\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}$ $[s_0,\ s_1] = \omega$ $\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pairlist}$ $\langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle$ $[s_0,\ s_1]$›
⟨*proof*⟩

**lemma** $\omega$-*combine*$_{Pair}$-*Sync*$_{ptick}$-*behaviour*:
‹$\omega$ $\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}$ $(s_0,\ s_1) = \omega$ $\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pair}$ $\langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle$ $(s_0,\ s_1)$›
⟨*proof*⟩

**lemma** $\omega$-*combine*$_{ListslenL}$-*Sync*$_{ptick}$-*behaviour*:
‹$\omega$ $\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}$ $(\sigma s_0\ @\ \sigma s_1) = \omega$ $\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}$ $\langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle$ $(\sigma s_0\ @\ \sigma s_1)$›
⟨*proof*⟩

**lemma** $\omega$-*combine*$_{Rlist}$-*Sync*$_{ptick}$-*behaviour*:
‹$\omega$ $\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}$ $(s_0\ \#\ \sigma s_1) = \omega$ $\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}$ $\langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle$ $(s_0\ \#\ \sigma s_1)$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{Pairlist}$-*Sync*$_{ptick}$-*behaviour-when-indep*:
‹$\varepsilon\ A_0\ s_0 \cap \varepsilon\ A_1\ s_1 \subseteq E \Longrightarrow$
$\tau$ $\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}$ $[s_0,\ s_1]\ e = \tau$ $\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pairlist}$ $\langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle$ $[s_0,\ s_1]\ e$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{Pair}$-*Sync*$_{ptick}$-*behaviour-when-indep*:
‹$\varepsilon\ A_0\ s_0 \cap \varepsilon\ A_1\ s_1 \subseteq E \Longrightarrow$
$\tau$ $\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}$ $(s_0,\ s_1)\ e = \tau$ $\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pair}$ $\langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle$ $(s_0,\ s_1)\ e$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{ListslenL}$-*Sync*$_{ptick}$-*behaviour-when-indep*:
‹$\varepsilon\ A_0\ \sigma s_0 \cap \varepsilon\ A_1\ \sigma s_1 \subseteq E \Longrightarrow$ *length* $\sigma s_0 = len_0 \Longrightarrow$
$\tau\ \langle\!\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle\!\rangle_{d\hookrightarrow nd}\ (\sigma s_0\ @\ \sigma s_1)\ e = \tau\ \langle\!\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\ (\sigma s_0\ @\ \sigma s_1)\ e$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{Rlist}$-*Sync*$_{ptick}$-*behaviour-when-indep*:
‹$\varepsilon\ A_0\ s_0 \cap \varepsilon\ A_1\ \sigma s_1 \subseteq E \Longrightarrow$
$\tau\ \langle\!\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle\!\rangle_{d\hookrightarrow nd}\ (s_0\ \#\ \sigma s_1)\ e = \tau\ \langle\!\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\ (s_0\ \#\ \sigma s_1)\ e$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*combine*$_{Pairlist}$-*Sync*$_{ptick}$-*behaviour-when-indep*:
‹$P_{SKIPS}\langle\!\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle\!\rangle_d\ [s_0,\ s_1] = P_{SKIPS}\langle\!\langle\!\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pairlist}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\!\rangle_{nd}\ [s_0,\ s_1]$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ s_1$›
⟨*proof*⟩

**lemma** *P-combine*$_{Pairlist}$-*Sync*$_{ptick}$-*behaviour-when-indep*:
‹$P\langle\!\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle\!\rangle_d\ [s_0,\ s_1] = P\langle\!\langle\!\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pairlist}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\!\rangle_{nd}\ [s_0,\ s_1]$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ s_1$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*combine*$_{Pair}$-*Sync*$_{ptick}$-*behaviour-when-indep*:
‹$P_{SKIPS}\langle\!\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle\!\rangle_d\ (s_0,\ s_1) = P_{SKIPS}\langle\!\langle\!\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pair}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\!\rangle_{nd}\ (s_0,\ s_1)$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ s_1$›
⟨*proof*⟩

**lemma** *P-combine*$_{Pair}$-*Sync*$_{ptick}$-*behaviour-when-indep*:
‹$P\langle\!\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle\!\rangle_d\ (s_0,\ s_1) = P\langle\!\langle\!\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Pair}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\!\rangle_{nd}\ (s_0,\ s_1)$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ s_1$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*combine*$_{ListslenL}$-*Sync*$_{ptick}$-*behaviour-when-indep*:
‹$P_{SKIPS}\langle\!\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma s_0\ @\ \sigma s_1) = P_{SKIPS}\langle\!\langle\!\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\!\rangle_{nd}\ (\sigma s_0\ @\ \sigma s_1)$›
**if** ‹*indep-enabl* $A_0\ \sigma s_0\ E\ A_1\ \sigma s_1$› **and** ‹$\bigwedge \sigma t_0.\ \sigma t_0 \in \mathcal{R}_d\ A_0\ \sigma s_0 \Longrightarrow$ *length* $\sigma t_0 = len_0$›
⟨*proof*⟩

**lemma** *P-combine*$_{ListslenL}$-*Sync*$_{ptick}$-*behaviour-when-indep*:

‹$P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (\sigma s_0\ @\ \sigma s_1) = P\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![len_0,$
$E]\!]_{\checkmark ListslenL}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\rangle\!\rangle_{nd}\ (\sigma s_0\ @\ \sigma s_1)$›
**if** ‹*indep-enabl* $A_0\ \sigma s_0\ E\ A_1\ \sigma s_1$› **and** ‹$\bigwedge\sigma t_0.\ \sigma t_0 \in \mathcal{R}_d\ A_0\ \sigma s_0 \Longrightarrow$ *length* $\sigma t_0$
$= len_0$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*combine*$_{Rlist}$-*Sync*$_{ptick}$-*behaviour-when-indep*:
‹$P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (s_0\ \#\ \sigma s_1) = P_{SKIPS}\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}$
$\langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\rangle\!\rangle_{nd}\ (s_0\ \#\ \sigma s_1)$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ \sigma s_1$›
⟨*proof*⟩

**lemma** *P-combine*$_{Rlist}$-*Sync*$_{ptick}$-*behaviour-when-indep*:
‹$P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (s_0\ \#\ \sigma s_1) = P\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}\ \langle\!\langle A_1\rangle\!\rangle_{d\hookrightarrow nd}\rangle\!\rangle\rangle\!\rangle_{nd}$
$(s_0\ \#\ \sigma s_1)$›
**if** ‹*indep-enabl* $A_0\ s_0\ E\ A_1\ \sigma s_1$›
⟨*proof*⟩

# Chapter 8

# Compactification of Synchronization Product Generalized

## 8.1 Iterated Combine

### 8.1.1 Definitions

**fun** *iterated-combine*$_d$*-Sync*$_{ptick}$ :: ‹$'e$ *set* $\Rightarrow$ $('\sigma,\ 'e,\ 'r)\ A_d$ *list* $\Rightarrow$ $('\sigma$ *list*, $'e$, $'r$ *list*$)\ A_d$› (‹$\langle\!\langle_d\bigotimes [\![-]\!]_{\checkmark}\ -\rangle\!\rangle$› $[0,\ 0]$)
  **where** ‹$\langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}\ []\rangle\!\rangle = (\!|\tau = \lambda\sigma s\ a.\ \Diamond,\ \omega = \lambda\sigma s.\ \Diamond|\!)$›
  | ‹$\langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}\ [A_0]\rangle\!\rangle = {}_d\langle\!\langle A_0\rangle\!\rangle_{singl\hookrightarrow list}$›
  | ‹$\langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}\ A_0\ \#\ A_1\ \#\ As\rangle\!\rangle = \langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Rlist}\ \langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}\ A_1\ \#\ As\rangle\!\rangle\rangle\!\rangle$›

**fun** *iterated-combine*$_{nd}$*-Sync*$_{ptick}$ :: ‹$'e$ *set* $\Rightarrow$ $('\sigma,\ 'e,\ 'r)\ A_{nd}$ *list* $\Rightarrow$ $('\sigma$ *list*, $'e$, $'r$ *list*$)\ A_{nd}$› (‹$\langle\!\langle_{nd}\bigotimes [\![-]\!]_{\checkmark}\ -\rangle\!\rangle$› $[0,\ 0]$)
  **where** ‹$\langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}\ []\rangle\!\rangle = (\!|\tau = \lambda\sigma s\ a.\ \{\},\ \omega = \lambda\sigma s.\ \{\}|\!)$›
  | ‹$\langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}\ [A_0]\rangle\!\rangle = {}_{nd}\langle\!\langle A_0\rangle\!\rangle_{singl\hookrightarrow list}$›
  | ‹$\langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}\ A_0\ \#\ A_1\ \#\ As\rangle\!\rangle = \langle\!\langle A_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}\ \langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}\ A_1\ \#\ As\rangle\!\rangle\rangle\!\rangle$›

**lemma** *iterated-combine*$_d$*-Sync*$_{ptick}$*-simps-bis*: ‹$As \neq [] \Longrightarrow \langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}\ A_0\ \#\ As\rangle\!\rangle = \langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Rlist}\ \langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}\ As\rangle\!\rangle\rangle\!\rangle$›
  **and** *iterated-combine*$_{nd}$*-Sync*$_{ptick}$*-simps-bis*: ‹$Bs \neq [] \Longrightarrow \langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}\ B_0\ \#\ Bs\rangle\!\rangle = \langle\!\langle B_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}\ \langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}\ Bs\rangle\!\rangle\rangle\!\rangle$›
  ⟨*proof*⟩

### 8.1.2 First Results

**lemma** $\tau$*-iterated-combine-Sync*$_{ptick}$:
  ‹$\tau\ \langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}\ As\rangle\!\rangle = \tau\ \langle\!\langle_d\bigotimes [\![E]\!]\ As\rangle\!\rangle$› ‹$\tau\ \langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}\ Bs\rangle\!\rangle = \tau\ \langle\!\langle_{nd}\bigotimes [\![E]\!]\ Bs\rangle\!\rangle$›
  ⟨*proof*⟩

**corollary** $\varepsilon$*-iterated-combine-Sync*$_{ptick}$:

‹$\varepsilon$ $\langle\!\langle {}_d\bigotimes [\![E]\!]_{\checkmark}\ As\rangle\!\rangle\ \sigma s = \varepsilon\ \langle\!\langle {}_d\bigotimes [\![E]\!]\ As\rangle\!\rangle\ \sigma s$›
‹$\varepsilon$ $\langle\!\langle {}_{nd}\bigotimes [\![E]\!]_{\checkmark}\ Bs\rangle\!\rangle\ \sigma s = \varepsilon\ \langle\!\langle {}_{nd}\bigotimes [\![E]\!]\ Bs\rangle\!\rangle\ \sigma s$›
⟨*proof*⟩

**corollary** $\mathcal{R}$*-iterated-combine-Sync*$_{ptick}$:
‹$\mathcal{R}_d$ $\langle\!\langle {}_d\bigotimes [\![E]\!]_{\checkmark}\ As\rangle\!\rangle = \mathcal{R}_d\ \langle\!\langle {}_d\bigotimes [\![E]\!]\ As\rangle\!\rangle$› ‹$\mathcal{R}_{nd}$ $\langle\!\langle {}_{nd}\bigotimes [\![E]\!]_{\checkmark}\ Bs\rangle\!\rangle = \mathcal{R}_{nd}\ \langle\!\langle {}_{nd}\bigotimes [\![E]\!]\ Bs\rangle\!\rangle$›
⟨*proof*⟩

**lemma** *combine*$_{ListslenL}$*-Sync*$_{ptick}$*-combine*$_{Rlist}$*-Sync*$_{ptick}$*-eq*:
‹$\varepsilon$ $\langle\!\langle {}_d\langle\!\langle A_0\rangle\!\rangle_{singl\hookrightarrow list}\ {}_d\otimes [\![1,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle\ \sigma s = \varepsilon\ \langle\!\langle A_0\ {}_d\otimes [\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle\ \sigma s$›
‹$\tau$ $\langle\!\langle {}_d\langle\!\langle A_0\rangle\!\rangle_{singl\hookrightarrow list}\ {}_d\otimes [\![1,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle\ (s_0\ \#\ \sigma s)\ e = \tau\ \langle\!\langle A_0\ {}_d\otimes [\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle\ (s_0\ \#\ \sigma s)\ e$›
‹$\varepsilon$ $\langle\!\langle {}_{nd}\langle\!\langle B_0\rangle\!\rangle_{singl\hookrightarrow list}\ {}_{nd}\otimes [\![1,\ E]\!]_{\checkmark ListslenL}\ B_1\rangle\!\rangle\ \sigma s = \varepsilon\ \langle\!\langle B_0\ {}_{nd}\otimes [\![E]\!]_{\checkmark Rlist}\ B_1\rangle\!\rangle\ \sigma s$›
‹$\tau$ $\langle\!\langle {}_{nd}\langle\!\langle B_0\rangle\!\rangle_{singl\hookrightarrow list}\ {}_{nd}\otimes [\![1,\ E]\!]_{\checkmark ListslenL}\ B_1\rangle\!\rangle\ (s_0\ \#\ \sigma s)\ e = \tau\ \langle\!\langle B_0\ {}_{nd}\otimes [\![E]\!]_{\checkmark Rlist}\ B_1\rangle\!\rangle\ (s_0\ \#\ \sigma s)\ e$›
⟨*proof*⟩

**lemma** *combine*$_{Pairlist}$*-Sync*$_{ptick}$*-and-iterated-combine*$_{nd}$*-Sync*$_{ptick}$*-eq*:
‹$\varepsilon$ $\langle\!\langle A_0\ {}_d\otimes [\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle\ [s_0,\ s_1] = \varepsilon\ \langle\!\langle {}_d\bigotimes [\![E]\!]_{\checkmark}\ [A_0,\ A_1]\rangle\!\rangle\ [s_0,\ s_1]$›
‹$\tau$ $\langle\!\langle A_0\ {}_d\otimes [\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle\ [s_0,\ s_1]\ e = \tau\ \langle\!\langle {}_d\bigotimes [\![E]\!]_{\checkmark}\ [A_0,\ A_1]\rangle\!\rangle\ [s_0,\ s_1]\ e$›
‹$\varepsilon$ $\langle\!\langle B_0\ {}_{nd}\otimes [\![E]\!]_{\checkmark Pairlist}\ B_1\rangle\!\rangle\ [s_0,\ s_1] = \varepsilon\ \langle\!\langle {}_{nd}\bigotimes [\![E]\!]_{\checkmark}\ [B_0,\ B_1]\rangle\!\rangle\ [s_0,\ s_1]$›
‹$\tau$ $\langle\!\langle B_0\ {}_{nd}\otimes [\![E]\!]_{\checkmark Pairlist}\ B_1\rangle\!\rangle\ [s_0,\ s_1]\ e = \tau\ \langle\!\langle {}_{nd}\bigotimes [\![E]\!]_{\checkmark}\ [B_0,\ B_1]\rangle\!\rangle\ [s_0,\ s_1]\ e$›
⟨*proof*⟩

**lemmas** *combine*$_{Pairlist}$*-Sync*$_{ptick}$*-and-combine*$_{Rlist}$*-Sync*$_{ptick}$*-eq* =
*combine*$_{Pairlist}$*-Sync*$_{ptick}$*-and-iterated-combine*$_{nd}$*-Sync*$_{ptick}$*-eq*[*simplified*]

### 8.1.3 Transmission of Properties

**lemma** *finite-trans-transmission-to-iterated-combine*$_{nd}$*-Sync*$_{ptick}$:
‹$(\bigwedge A.\ A \in set\ As \Longrightarrow finite\text{-}trans\ A) \Longrightarrow finite\text{-}trans\ \langle\!\langle {}_{nd}\bigotimes [\![E]\!]_{\checkmark}\ As\rangle\!\rangle$›
⟨*proof*⟩

**lemma** $\varrho$*-disjoint-*$\varepsilon$*-transmission-to-iterated-combine*$_d$*-Sync*$_{ptick}$:
‹$(\bigwedge A.\ A \in set\ As \Longrightarrow \varrho\text{-}disjoint\text{-}\varepsilon\ A) \Longrightarrow \varrho\text{-}disjoint\text{-}\varepsilon\ \langle\!\langle {}_d\bigotimes [\![E]\!]_{\checkmark}\ As\rangle\!\rangle$›
⟨*proof*⟩

**lemma** $\varrho$*-disjoint-*$\varepsilon$*-transmission-to-iterated-combine*$_{nd}$*-Sync*$_{ptick}$:
‹$\forall B \in set\ Bs.\ \varrho\text{-}disjoint\text{-}\varepsilon\ B \Longrightarrow \varrho\text{-}disjoint\text{-}\varepsilon\ \langle\!\langle {}_{nd}\bigotimes [\![E]\!]_{\checkmark}\ Bs\rangle\!\rangle$›
⟨*proof*⟩

**lemma** *same-length-indep-transmission-to-iterated-combine$_d$-Sync$_{ptick}$*:
‹*indep-enabl* $A_0$ $s_0$ $E$ $\langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}$ $As\rangle\!\rangle$ $\sigma s$›
**if** ‹*length* $\sigma s$ = *length* $As$›
‹$\bigwedge i\ j.\ [\![i \leq$ *length* $As$; $j \leq$ *length* $As$; $i \neq j]\!] \Longrightarrow$
*indep-enabl* $((A_0\ \#\ As)\ !\ i)\ ((s_0\ \#\ \sigma s)\ !\ i)\ E\ ((A_0\ \#\ As)\ !\ j)\ ((s_0\ \#\ \sigma s)$
$!\ j)$›
⟨*proof*⟩

**lemma** *$\omega$-iterated-combine$_d$-Sync$_{ptick}$* :
‹*length* $\sigma s$ = *length* $As \Longrightarrow$
$\omega$ $\langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}$ $As\rangle\!\rangle$ $\sigma s$ = (*if* $As$ = [] *then* $\Diamond$ *else those* (*map2* $\omega$ $As$ $\sigma s$))›
⟨*proof*⟩

**lemma** *$\omega$-iterated-combine$_{nd}$-Sync$_{ptick}$* :
‹*length* $\sigma s$ = *length* $As \Longrightarrow$
$\omega$ $\langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}$ $As\rangle\!\rangle$ $\sigma s$ =
(*if* $As$ = [] *then* {} *else* {$rs$. *length* $rs$ = *length* $As$ $\wedge$ ($\forall i <$ *length* $As$. $rs\ !\ i \in \omega$
$(As\ !\ i)\ (\sigma s\ !\ i)$)})›
⟨*proof*⟩

### 8.1.4 Normalization

**lemma** *$\omega$-iterated-combine$_{nd}$-Sync$_{ptick}$-det-ndet-conv*:
‹*length* $\sigma s$ = *length* $As \Longrightarrow$
$\omega$ $\langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}$ *map* ($\lambda A$. $\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}$) $As\rangle\!\rangle$ $\sigma s$ = $\omega$ $\langle\!\langle\langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}$ $As\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}$ $\sigma s$›
⟨*proof*⟩

**lemma** *$\tau$-iterated-combine$_{nd}$-Sync$_{ptick}$-behaviour-when-indep*:
‹*length* $\sigma s$ = *length* $As \Longrightarrow$
($\bigwedge i\ j.\ [\![i <$ *length* $As$; $j <$ *length* $As$; $i \neq j]\!]$
$\Longrightarrow$ *indep-enabl* $(As\ !\ i)\ (\sigma s\ !\ i)\ E\ (As\ !\ j)\ (\sigma s\ !\ j)) \Longrightarrow$
$\tau$ $\langle\!\langle\langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}$ $As\rangle\!\rangle\rangle\!\rangle_{d\hookrightarrow nd}$ $\sigma s$ $e$ = $\tau$ $\langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}$ *map* ($\lambda A$. $\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}$) $As\rangle\!\rangle$ $\sigma s$ $e$›
⟨*proof*⟩

**lemma** *$P_{SKIPS}$-iterated-combine$_{nd}$-Sync$_{ptick}$-behaviour-when-indep*:
**assumes** *same-length*: ‹*length* $\sigma s$ = *length* $As$›
**and** *indep*: ‹$\bigwedge i\ j.\ [\![i <$ *length* $As$; $j <$ *length* $As$; $i \neq j]\!] \Longrightarrow$
*indep-enabl* $(As\ !\ i)\ (\sigma s\ !\ i)\ E\ (As\ !\ j)\ (\sigma s\ !\ j)$›
**shows** ‹$P_{SKIPS}\langle\!\langle\langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}$ $As\rangle\!\rangle\rangle\!\rangle_d$ $\sigma s$ = $P_{SKIPS}\langle\!\langle\langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}$ *map* ($\lambda A$. $\langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd}$)
$As\rangle\!\rangle\rangle\!\rangle_{nd}$ $\sigma s$›
⟨*proof*⟩

**lemma** *P-d-iterated-combine$_{nd}$-Sync$_{ptick}$-behaviour-when-indep*:
**assumes** *same-length*: ‹*length* $\sigma s$ = *length* $As$›

**and** *indep*: ‹$\bigwedge i\ j.\ [\![i < length\ As;\ j < length\ As;\ i \neq j]\!] \Longrightarrow$
*indep-enabl* $(As\ !\ i)\ (\sigma s\ !\ i)\ E\ (As\ !\ j)\ (\sigma s\ !\ j)$›
**shows** ‹$P\langle\!\langle\langle\!\langle_d \bigotimes [\![E]\!]_{\checkmark}\ As\rangle\!\rangle\!\rangle_d\ \sigma s = P\langle\!\langle\langle\!\langle_{nd} \bigotimes [\![E]\!]_{\checkmark}\ map\ (\lambda A.\ \langle\!\langle A\rangle\!\rangle_{d\hookrightarrow nd})\ As\rangle\!\rangle\!\rangle_{nd}\ \sigma s$›
⟨*proof*⟩

## 8.2 Compactification Theorems

### 8.2.1 Binary

#### Pair

**theorem** $P_{SKIPS}$-*nd-combine*$_{Pair}$-*Sync*$_{ptick}$ :
**fixes** $E$ :: ‹$'a$ *set*›
**assumes** *ϱ-disjoint-ε* : ‹*ϱ-disjoint-ε* $A_0$› ‹*ϱ-disjoint-ε* $A_1$›
**defines** *A-def*: ‹$A \equiv \langle\!\langle A_0\ _{nd}\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle$›
**defines** *P-def*: ‹$P \equiv P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}$› **and** *Q-def*: ‹$Q \equiv P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_{nd}$› **and** *S-def*: ‹$S \equiv P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}$›
**shows** ‹$P\ \sigma_0\ [\![E]\!]_{\checkmark Pair}\ Q\ \sigma_1 = S\ (\sigma_0, \sigma_1)$›
⟨*proof*⟩

**corollary** *P-nd-combine*$_{Pair}$-*Sync*$_{ptick}$ :
‹$P\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ [\![E]\!]_{\checkmark Pair}\ P\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ _{nd}\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle\!\rangle_{nd}\ (\sigma_0, \sigma_1)$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$-*d-combine*$_{Pair}$-*Sync*$_{ptick}$:
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ [\![E]\!]_{\checkmark Pair}\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_d\ \sigma_1 = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ _d\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma_0, \sigma_1)$›
**if** ‹*ϱ-disjoint-ε* $A_0$› **and** ‹*ϱ-disjoint-ε* $A_1$› **and** ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma_1$›
⟨*proof*⟩

**corollary** *P-d-combine*$_{Pair}$-*Sync*$_{ptick}$:
‹$P\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ [\![E]\!]_{\checkmark Pair}\ P\langle\!\langle A_1\rangle\!\rangle_d\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ _d\otimes[\![E]\!]_{\checkmark Pair}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma_0, \sigma_1)$›
**if** ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma_1$›
⟨*proof*⟩

#### Pairlist

**theorem** $P_{SKIPS}$-*nd-combine*$_{Pairlist}$-*Sync*$_{ptick}$ :
**fixes** $E$ :: ‹$'a$ *set*›
**assumes** *ϱ-disjoint-ε* : ‹*ϱ-disjoint-ε* $A_0$› ‹*ϱ-disjoint-ε* $A_1$›
**shows** ‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ [\![E]\!]_{\checkmark Pairlist}\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma_1 = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ _{nd}\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle\!\rangle_{nd}\ [\sigma_0, \sigma_1]$›
⟨*proof*⟩

**corollary** *P-nd-combine*$_{Pairlist}$-*Sync*$_{ptick}$ :
‹$P\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ [\![E]\!]_{\checkmark Pairlist}\ P\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ _{nd}\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle\!\rangle_{nd}\ [\sigma_0, \sigma_1]$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$-*d-combine*$_{Pairlist}$-*Sync*$_{ptick}$ :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ [\![E]\!]_{\checkmark Pairlist}\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_d\ \sigma_1 = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle\!\rangle_d\ [\sigma_0,\ \sigma_1]$›
**if** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› **and** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$› **and** ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma_1$›
⟨*proof*⟩

**corollary** *P-d-combine*$_{Pairlist}$-*Sync*$_{ptick}$ :
‹$P\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ [\![E]\!]_{\checkmark Pairlist}\ P\langle\!\langle A_1\rangle\!\rangle_d\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Pairlist}\ A_1\rangle\!\rangle\!\rangle_d\ [\sigma_0,\ \sigma_1]$›
**if** ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma_1$›
⟨*proof*⟩

### 8.2.2 Rlist

**theorem** $P_{SKIPS}$-*nd-combine*$_{Rlist}$-*Sync*$_{ptick}$ :
**fixes** $E$ :: ‹*'a set*›
**assumes** $\varrho$-*disjoint*-$\varepsilon$ : ‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$›
**defines** *A-def*: ‹$A \equiv \langle\!\langle A_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle$›
**defines** *P-def*: ‹$P \equiv P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}$› **and** *Q-def*: ‹$Q \equiv P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_{nd}$› **and** *S-def*: ‹$S \equiv P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}$›
**shows** ‹$P\ \sigma_0\ [\![E]\!]_{\checkmark Rlist}\ Q\ \sigma s = S\ (\sigma_0\ \#\ \sigma s)$›
⟨*proof*⟩

**corollary** *P-nd-combine*$_{Rlist}$-*Sync*$_{ptick}$ :
‹$P\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ [\![E]\!]_{\checkmark Rlist}\ P\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma s = P\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle\!\rangle_{nd}\ (\sigma_0\ \#\ \sigma s)$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$-*d-combine*$_{Rlist}$-*Sync*$_{ptick}$:
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ [\![E]\!]_{\checkmark Rlist}\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_d\ \sigma s = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma_0\ \#\ \sigma s)$›
**if** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› **and** ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$› **and** ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma s$›
⟨*proof*⟩

**corollary** *P-d-combine*$_{Rlist}$-*Sync*$_{ptick}$:
‹$P\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ [\![E]\!]_{\checkmark Rlist}\ P\langle\!\langle A_1\rangle\!\rangle_d\ \sigma s = P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![E]\!]_{\checkmark Rlist}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma_0\ \#\ \sigma s)$›
**if** ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma s$›
⟨*proof*⟩

### 8.2.3 ListslenL

**theorem** $P_{SKIPS}$-*nd-combine*$_{ListslenL}$-*Sync*$_{ptick}$ :
**fixes** $E$ :: ‹*'a set*›
**assumes** *same-length-reach0* : ‹$\bigwedge\sigma_0'.\ \sigma_0' \in \mathcal{R}_{nd}\ A_0\ \sigma_0 \Longrightarrow length\ \sigma_0' = len_0$›
**and** *same-length-term0* : ‹$\bigwedge\sigma_0'\ rs.\ \sigma_0' \in \mathcal{R}_{nd}\ A_0\ \sigma_0 \Longrightarrow rs \in \omega\ A_0\ \sigma_0' \Longrightarrow length\ rs = len_0$›
**and** $\varrho$-*disjoint*-$\varepsilon$ : ‹$\varrho$-*disjoint*-$\varepsilon$ $A_0$› ‹$\varrho$-*disjoint*-$\varepsilon$ $A_1$›
**defines** *A-def*: ‹$A \equiv \langle\!\langle A_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle$›
**defines** *P-def*: ‹$P \equiv P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}$› **and** *Q-def*: ‹$Q \equiv P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_{nd}$› **and** *S-def*: ‹$S \equiv P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}$›
**shows** ‹$P\ \sigma_0\ {}_{len_0}[\![E]\!]_{\checkmark ListslenL}\ Q\ \sigma_1 = S\ (\sigma_0\ @\ \sigma_1)$›
⟨*proof*⟩

**corollary** *P-nd-combine*$_{ListslenL}$*-Sync*$_{ptick}$ :
‹$P\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ {}_{len_0}[\![E]\!]_{\checkmark ListslenL}\ P\langle\!\langle A_1\rangle\!\rangle_{nd}\ \sigma_1 = P\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle\!\rangle_{nd}\ (\sigma_0\ @\ \sigma_1)$›
**if** *same-length-reach0* : ‹$\bigwedge\sigma_0'.\ \sigma_0' \in \mathcal{R}_{nd}\ A_0\ \sigma_0 \Longrightarrow length\ \sigma_0' = len_0$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$*-d-combine*$_{ListslenL}$*-Sync*$_{ptick}$ :
**assumes** *same-length-reach0* : ‹$\bigwedge\sigma_0'.\ \sigma_0' \in \mathcal{R}_d\ A_0\ \sigma_0 \Longrightarrow length\ \sigma_0' = len_0$›
**and** *same-length-term0* : ‹$\bigwedge\sigma_0'.\ \sigma_0' \in \mathcal{R}_d\ A_0\ \sigma_0 \Longrightarrow \omega\ A_0\ \sigma_0' \neq \Diamond \Longrightarrow length\ \lceil\omega\ A_0\ \sigma_0'\rceil = len_0$›
**and** *ϱ-disjoint-ε* : ‹*ϱ-disjoint-ε* $A_0$› ‹*ϱ-disjoint-ε* $A_1$›
**and** *indep-enabl* : ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma_1$›
**shows** ‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ {}_{len_0}[\![E]\!]_{\checkmark ListslenL}\ P_{SKIPS}\langle\!\langle A_1\rangle\!\rangle_d\ \sigma_1 =$
$P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma_0\ @\ \sigma_1)$›
⟨*proof*⟩

**corollary** *P-d-combine*$_{ListslenL}$*-Sync*$_{ptick}$ :
**assumes** *same-length-reach0* : ‹$\bigwedge\sigma_0'.\ \sigma_0' \in \mathcal{R}_d\ A_0\ \sigma_0 \Longrightarrow length\ \sigma_0' = len_0$›
**and** *indep-enabl* : ‹*indep-enabl* $A_0\ \sigma_0\ E\ A_1\ \sigma_1$›
**shows** ‹$P\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ {}_{len_0}[\![E]\!]_{\checkmark ListslenL}\ P\langle\!\langle A_1\rangle\!\rangle_d\ \sigma_1 =$
$P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes[\![len_0,\ E]\!]_{\checkmark ListslenL}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma_0\ @\ \sigma_1)$›
⟨*proof*⟩

### 8.2.4 Multiple

**theorem** $P_{SKIPS}$*-nd-compactification-Sync*$_{ptick}$:
‹$length\ \sigma s = length\ As \Longrightarrow (\bigwedge A.\ A \in set\ As \Longrightarrow$ *ϱ-disjoint-ε* $A) \Longrightarrow$
$[\![E]\!]_{\checkmark}\ (\sigma,\ A) \in@\ zip\ \sigma s\ As.\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma = P_{SKIPS}\langle\!\langle\langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}\ As\rangle\!\rangle\!\rangle_{nd}\ \sigma s$›
⟨*proof*⟩

**corollary** *P-nd-compactification-Sync*$_{ptick}$:
‹$length\ \sigma s = length\ As \Longrightarrow [\![E]\!]_{\checkmark}\ (\sigma,\ A) \in@\ zip\ \sigma s\ As.\ P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma = P\langle\!\langle\langle\!\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}\ As\rangle\!\rangle\!\rangle_{nd}\ \sigma s$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$*-d-compactification-Sync*$_{ptick}$:
‹$[\![length\ \sigma s = length\ As;\ \bigwedge A.\ A \in set\ As \Longrightarrow$ *ϱ-disjoint-ε* $A;$
$\bigwedge i\ j.\ [\![i < length\ As;\ j < length\ As;\ i \neq j]\!] \Longrightarrow$
*indep-enabl* $(As\ !\ i)\ (\sigma s\ !\ i)\ E\ (As\ !\ j)\ (\sigma s\ !\ j)]\!] \Longrightarrow$
$[\![E]\!]_{\checkmark}\ (\sigma,\ A) \in@\ zip\ \sigma s\ As.\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma = P_{SKIPS}\langle\!\langle\langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}\ As\rangle\!\rangle\!\rangle_d\ \sigma s$›
⟨*proof*⟩

**corollary** *P-d-compactification-Sync*$_{ptick}$:
‹$[\![length\ \sigma s = length\ As;$
$\bigwedge i\ j.\ [\![i < length\ As;\ j < length\ As;\ i \neq j]\!] \Longrightarrow$
*indep-enabl* $(As\ !\ i)\ (\sigma s\ !\ i)\ E\ (As\ !\ j)\ (\sigma s\ !\ j)]\!] \Longrightarrow$
$[\![E]\!]_{\checkmark}\ (\sigma,\ A) \in@\ zip\ \sigma s\ As.\ P\langle\!\langle A\rangle\!\rangle_d\ \sigma = P\langle\!\langle\langle\!\langle_d\bigotimes [\![E]\!]_{\checkmark}\ As\rangle\!\rangle\!\rangle_d\ \sigma s$›

⟨*proof*⟩

## 8.3 Derived Versions

**lemma** $P_{SKIPS}$*-nd-compactification-Sync*$_{ptick}$*-upt-version*:
‹$[\![E]\!]_{\checkmark}$ $P$ ∈@ *map* $Q$ $[0..<n]$. $P = P_{SKIPS}\langle\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}$ *map* $A$ $[0..<n]\rangle\rangle_{nd}$ (*replicate* $n$ $0$)›
**if** ‹$\bigwedge i.\ i < n \Longrightarrow$ *ϱ-disjoint-ε* $(A\ i)$›
‹$\bigwedge i.\ i < n \Longrightarrow P_{SKIPS}\langle\langle A\ i\rangle\rangle_{nd}\ 0 = Q\ i$›
⟨*proof*⟩

**lemma** *P-nd-compactification-Sync*$_{ptick}$*-upt-version*:
‹$[\![E]\!]_{\checkmark}$ $P$ ∈@ *map* $Q$ $[0..<n]$. $P = P\langle\langle_{nd}\bigotimes [\![E]\!]_{\checkmark}$ *map* $A$ $[0..<n]\rangle\rangle_{nd}$ (*replicate* $n$ $0$)›
**if** ‹$\bigwedge i.\ i < n \Longrightarrow P\langle\langle A\ i\rangle\rangle_{nd}\ 0 = Q\ i$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-d-compactification-Sync*$_{ptick}$*-upt-version*:
‹$[\![E]\!]_{\checkmark}$ $P$ ∈@ *map* $Q$ $[0..<n]$. $P = P_{SKIPS}\langle\langle_{d}\bigotimes [\![E]\!]_{\checkmark}$ *map* $A$ $[0..<n]\rangle\rangle_{d}$ (*replicate* $n$ $0$)›
**if** ‹$\bigwedge i.\ i < n \Longrightarrow$ *ϱ-disjoint-ε* $(A\ i)$›
‹$\bigwedge i\ j.\ i < n \Longrightarrow j < n \Longrightarrow i \neq j \Longrightarrow$ *indep-enabl* $(A\ i)\ 0\ E\ (A\ j)\ 0$›
‹$\bigwedge i.\ i < n \Longrightarrow P_{SKIPS}\langle\langle A\ i\rangle\rangle_{d}\ 0 = Q\ i$›
⟨*proof*⟩

**lemma** *P-d-compactification-Sync*$_{ptick}$*-upt-version*:
‹$[\![E]\!]_{\checkmark}$ $P$ ∈@ *map* $Q$ $[0..<n]$. $P = P\langle\langle_{d}\bigotimes [\![E]\!]_{\checkmark}$ *map* $A$ $[0..<n]\rangle\rangle_{d}$ (*replicate* $n$ $0$)›
**if** ‹$\bigwedge i\ j.\ i < n \Longrightarrow j < n \Longrightarrow i \neq j \Longrightarrow$ *indep-enabl* $(A\ i)\ 0\ E\ (A\ j)\ 0$›
‹$\bigwedge i.\ i < n \Longrightarrow P\langle\langle A\ i\rangle\rangle_{d}\ 0 = Q\ i$›
⟨*proof*⟩

## 8.4 More on Iterated Combine and Events

Through $\tau$*-iterated-combine-Sync*$_{ptick}$ $\varepsilon$*-iterated-combine-Sync*$_{ptick}$ $\mathcal{R}$*-iterated-combine-Sync*$_{ptick}$, we immediately recover the results proven in *HOL−CSP-Proc−Omata.Compactification-Synchronization-P*

# Chapter 9

# Combining Automata for Sequential Composition Generalized

## 9.1 Definitions

## 9.2 General Patterns

**definition** *combine$_d$-Seq-ε* ::
‹[('$\sigma_0$, 'a, 'r, '$\alpha_0$) $A_d$-*scheme*, 'r ⇒ ('$\sigma_1$, 'a, 's, '$\alpha_1$) $A_d$-*scheme*, '$\sigma$ ⇒ '$\sigma_0$, '$\sigma$ ⇒ '$\sigma_1$, '$\sigma$] ⇒ 'a *set*›
**where** ‹*combine$_d$-Seq-ε* $A_0$ $A_1$ $i_0$ $i_1$ $\sigma s$ ≡
*if* $i_0$ $\sigma s$ ∈ $\varrho$ $A_0$
*then* *if* $i_1$ $\sigma s$ ∈ $\varrho$ ($A_1$ ⌈$\omega$ $A_0$ ($i_0$ $\sigma s$)⌉)
*then* {}
*else* $\varepsilon$ ($A_1$ ⌈$\omega$ $A_0$ ($i_0$ $\sigma s$)⌉) ($i_1$ $\sigma s$)
*else* $\varepsilon$ $A_0$ ($i_0$ $\sigma s$)›

**definition** *combine$_{nd}$-Seq-ε* ::
‹[('$\sigma_0$, 'a, 'r, '$\alpha_0$) $A_{nd}$-*scheme*, 'r ⇒ ('$\sigma_1$, 'a, 's, '$\alpha_1$) $A_{nd}$-*scheme*, '$\sigma$ ⇒ '$\sigma_0$, '$\sigma$ ⇒ '$\sigma_1$, '$\sigma$] ⇒ 'a *set*›
**where** ‹*combine$_{nd}$-Seq-ε* $A_0$ $A_1$ $i_0$ $i_1$ $\sigma s$ ≡
*if* $i_0$ $\sigma s$ ∈ $\varrho$ $A_0$
*then* *if* $i_1$ $\sigma s$ ∈ (⋃$r$∈$\omega$ $A_0$ ($i_0$ $\sigma s$). $\varrho$ ($A_1$ $r$))
*then* {}
*else* (⋃$r$ ∈ $\omega$ $A_0$ ($i_0$ $\sigma s$). $\varepsilon$ ($A_1$ $r$) ($i_1$ $\sigma s$))
*else* $\varepsilon$ $A_0$ ($i_0$ $\sigma s$)›

**lemmas** *combine-Seq-ε-defs* = *combine$_d$-Seq-ε-def* *combine$_{nd}$-Seq-ε-def*

**fun** *combine$_d$-Seq* ::
‹[('$\sigma_0$, 'e, 'r, '$\alpha_0$) $A_d$-*scheme*, 'r ⇒ ('$\sigma_1$, 'e, 's, '$\alpha_1$) $A_d$-*scheme*,

$'\sigma \Rightarrow '\sigma_0,\ '\sigma \Rightarrow '\sigma_1,\ '\sigma_0 \Rightarrow '\sigma_1 \Rightarrow '\sigma] \Rightarrow ('\sigma,\ 'e,\ 's)\ A_d$›
**and** *combine*$_{nd}$*-Seq* ::
‹$[('\sigma_0,\ 'e,\ 'r,\ '\alpha_0)\ A_{nd}$*-scheme*, $'r \Rightarrow ('\sigma_1,\ 'e,\ 's,\ '\alpha_1)\ A_{nd}$*-scheme*,
$'\sigma \Rightarrow '\sigma_0,\ '\sigma \Rightarrow '\sigma_1,\ '\sigma_0 \Rightarrow '\sigma_1 \Rightarrow '\sigma] \Rightarrow ('\sigma,\ 'e,\ 's)\ A_{nd}$›
**where** ‹*combine*$_d$*-Seq* $A_0\ A_1\ i_0\ i_1\ f =$
$(\!|\tau = \lambda\sigma s\ e.$ *if* $i_0\ \sigma s \in \varrho\ A_0$
*then* *if* $i_1\ \sigma s \in \varrho\ (A_1\ \lceil\omega\ A_0\ (i_0\ \sigma s)\rceil)$
*then* $\Diamond$
*else update-right* $(A_1\ \lceil\omega\ A_0\ (i_0\ \sigma s)\rceil)\ (i_0\ \sigma s)\ (i_1\ \sigma s)\ e$ (*f-up-opt* $f$) $(\lambda\sigma.\ \lfloor\sigma\rfloor)$
*else update-left* $A_0\ (i_0\ \sigma s)\ (i_1\ \sigma s)\ e$ (*f-up-opt* $f$) $(\lambda\sigma.\ \lfloor\sigma\rfloor)$,
$\omega = \lambda\sigma s.$ *case* $\omega\ A_0\ (i_0\ \sigma s)$ *of* $\Diamond \Rightarrow \Diamond \mid \lfloor r\rfloor \Rightarrow \omega\ (A_1\ r)\ (i_1\ \sigma s)|\!)$›
| ‹*combine*$_{nd}$*-Seq* $A_0\ A_1\ i_0\ i_1\ f =$
$(\!|\tau = \lambda\sigma s\ e.$ *if* $i_0\ \sigma s \in \varrho\ A_0$
*then* *if* $i_1\ \sigma s \in (\bigcup r{\in}\omega\ A_0\ (i_0\ \sigma s).\ \varrho\ (A_1\ r))$
*then* $\{\}$
*else* $(\bigcup r{\in}\omega\ A_0\ (i_0\ \sigma s).$ *update-right* $(A_1\ r)\ (i_0\ \sigma s)\ (i_1\ \sigma s)\ e$ (*f-up-set* $f$) $(\lambda\sigma.\ \{\sigma\}))$
*else* *update-left* $A_0\ (i_0\ \sigma s)\ (i_1\ \sigma s)\ e$ (*f-up-set* $f$) $(\lambda\sigma.\ \{\sigma\})$,
$\omega = \lambda\sigma s.\ \bigcup r{\in}\omega\ A_0\ (i_0\ \sigma s).\ \omega\ (A_1\ r)\ (i_1\ \sigma s)|\!)$›

## 9.3 Specializations

**definition** *combine*$_{dPairlist}$*-Seq*$_{ptick}$ ::
‹$[('\sigma,\ 'e,\ 'r,\ '\alpha)\ A_d$*-scheme*, $'r \Rightarrow ('\sigma,\ 'e,\ 's,\ '\beta)\ A_d$*-scheme*$] \Rightarrow ('\sigma$ *list*, $'e,\ 's)\ A_d$›
**where** ‹*combine*$_{dPairlist}$*-Seq*$_{ptick}$ $A_0\ A_1 \equiv$ *combine*$_d$*-Seq* $A_0\ A_1$ *hd* $(\lambda\sigma s.$ *hd* (*tl* $\sigma s)) (\lambda s\ t.\ [s,\ t])$›
**definition** *combine*$_{ndPairlist}$*-Seq*$_{ptick}$ ::
‹$[('\sigma,\ 'e,\ 'r,\ '\alpha)\ A_{nd}$*-scheme*, $'r \Rightarrow ('\sigma,\ 'e,\ 's,\ '\beta)\ A_{nd}$*-scheme*$] \Rightarrow ('\sigma$ *list*, $'e,\ 's)$ $A_{nd}$›
**where** ‹*combine*$_{ndPairlist}$*-Seq*$_{ptick}$ $A_0\ A_1 \equiv$ *combine*$_{nd}$*-Seq* $A_0\ A_1$ *hd* $(\lambda\sigma s.$ *hd* (*tl* $\sigma s)) (\lambda s\ t.\ [s,\ t])$›

**definition** *combine*$_{dPair}$*-Seq*$_{ptick}$ ::
‹$[('\sigma_0,\ 'e,\ 'r,\ '\alpha)\ A_d$*-scheme*, $'r \Rightarrow ('\sigma_1,\ 'e,\ 's,\ '\beta)\ A_d$*-scheme*$] \Rightarrow ('\sigma_0 \times '\sigma_1,\ 'e,$ $'s)\ A_d$›
**where** ‹*combine*$_{dPair}$*-Seq*$_{ptick}$ $A_0\ A_1 \equiv$ *combine*$_d$*-Seq* $A_0\ A_1$ *fst snd Pair*›
**definition** *combine*$_{ndPair}$*-Seq*$_{ptick}$ ::
‹$[('\sigma_0,\ 'e,\ 'r,\ '\alpha)\ A_{nd}$*-scheme*, $'r \Rightarrow ('\sigma_1,\ 'e,\ 's,\ '\beta)\ A_{nd}$*-scheme*$] \Rightarrow ('\sigma_0 \times '\sigma_1,$ $'e,\ 's)\ A_{nd}$›
**where** ‹*combine*$_{ndPair}$*-Seq*$_{ptick}$ $A_0\ A_1 \equiv$ *combine*$_{nd}$*-Seq* $A_0\ A_1$ *fst snd Pair*›

**definition** *combine*$_{dListslenL}$*-Seq*$_{ptick}$ ::
‹$[('\sigma$ *list*, $'e,\ 'r,\ '\alpha)\ A_d$*-scheme*, *nat*, $'r \Rightarrow ('\sigma$ *list*, $'e,\ 's,\ '\beta)\ A_d$*-scheme*$] \Rightarrow ('\sigma$ *list*, $'e,\ 's)\ A_d$›
**where** ‹*combine*$_{dListslenL}$*-Seq*$_{ptick}$ $A_0$ *len*$_0$ $A_1 \equiv$ *combine*$_d$*-Seq* $A_0\ A_1$ (*take* *len*$_0$) (*drop* *len*$_0$) (@)›
**definition** *combine*$_{ndListslenL}$*-Seq*$_{ptick}$ ::
‹$[('\sigma$ *list*, $'e,\ 'r,\ '\alpha)\ A_{nd}$*-scheme*, *nat*, $'r \Rightarrow ('\sigma$ *list*, $'e,\ 's,\ '\beta)\ A_{nd}$*-scheme*$] \Rightarrow ('\sigma$

*list*, *'e*, *'s*) $A_{nd}$›
**where** ‹$combine_{ndListslenL}$-$Seq_{ptick}$ $A_0$ $len_0$ $A_1$ ≡ $combine_{nd}$-*Seq* $A_0$ $A_1$ (*take* $len_0$) (*drop* $len_0$) (@)›

**definition** $combine_{dRlist}$-$Seq_{ptick}$ ::
‹[(*'σ*, *'e*, *'r*, *'α*) $A_d$-*scheme*, *'r* ⇒ (*'σ list*, *'e*, *'s*, *'β*) $A_d$-*scheme*] ⇒ (*'σ list*, *'e*, *'s*) $A_d$›
**where** ‹$combine_{dRlist}$-$Seq_{ptick}$ $A_0$ $A_1$ ≡ $combine_d$-*Seq* $A_0$ $A_1$ *hd tl* (#)›
**definition** $combine_{ndRlist}$-$Seq_{ptick}$ ::
‹[(*'σ*, *'e*, *'r*, *'α*) $A_{nd}$-*scheme*, *'r* ⇒ (*'σ list*, *'e*, *'s*, *'β*) $A_{nd}$-*scheme*] ⇒ (*'σ list*, *'e*, *'s*) $A_{nd}$›
**where** ‹$combine_{ndRlist}$-$Seq_{ptick}$ $A_0$ $A_1$ ≡ $combine_{nd}$-*Seq* $A_0$ $A_1$ *hd tl* (#)›

**lemmas** $combine_{Pairlist}$-$Seq_{ptick}$-*defs* = $combine_{dPairlist}$-$Seq_{ptick}$-*def* $combine_{ndPairlist}$-$Seq_{ptick}$-*def*
**and** $combine_{Pair}$-$Seq_{ptick}$-*defs* = $combine_{dPair}$-$Seq_{ptick}$-*def* $combine_{ndPair}$-$Seq_{ptick}$-*def*
**and** $combine_{ListslenL}$-$Seq_{ptick}$ = $combine_{dListslenL}$-$Seq_{ptick}$-*def* $combine_{ndListslenL}$-$Seq_{ptick}$-*def*
**and** $combine_{Rlist}$-$Seq_{ptick}$-*defs* = $combine_{dRlist}$-$Seq_{ptick}$-*def* $combine_{ndRlist}$-$Seq_{ptick}$-*def*

**lemmas** *combine-Seq-defs* =
$combine_{Pairlist}$-$Seq_{ptick}$-*defs* $combine_{Pair}$-$Seq_{ptick}$-*defs* $combine_{ListslenL}$-$Seq_{ptick}$ $combine_{Rlist}$-$Seq_{ptick}$-*defs*

**bundle** *combine-Seq-syntax* **begin**

**notation** $combine_{dPairlist}$-$Seq_{ptick}$ (‹⟪- $_d$⊗;✓$_{Pairlist}$ -⟫› [*0*, *0*])
**notation** $combine_{ndPairlist}$-$Seq_{ptick}$ (‹⟪- $_{nd}$⊗;✓$_{Pairlist}$ -⟫› [*0*, *0*])
**notation** $combine_{dPair}$-$Seq_{ptick}$ (‹⟪- $_d$⊗;✓$_{Pair}$ -⟫› [*0*, *0*])
**notation** $combine_{ndPair}$-$Seq_{ptick}$ (‹⟪- $_{nd}$⊗;✓$_{Pair}$ -⟫› [*0*, *0*])
**notation** $combine_{dListslenL}$-$Seq_{ptick}$ (‹⟪- $_d$⊗-;$_{ListslenL}$ -⟫› [*0*, *0*, *0*])
**notation** $combine_{ndListslenL}$-$Seq_{ptick}$ (‹⟪- $_{nd}$⊗-;$_{ListslenL}$ -⟫› [*0*, *0*, *0*])
**notation** $combine_{dRlist}$-$Seq_{ptick}$ (‹⟪- $_d$⊗;✓$_{Rlist}$ -⟫› [*0*, *0*])
**notation** $combine_{ndRlist}$-$Seq_{ptick}$ (‹⟪- $_{nd}$⊗;✓$_{Rlist}$ -⟫› [*0*, *0*])

**end**

**unbundle** *combine-Seq-syntax*

## 9.4 First Properties

**lemma** *ε*-$combine_{Pairlist}$-$Seq_{ptick}$ :
‹*ε* ⟪$A_0$ $_d$⊗;✓$_{Pairlist}$ $A_1$⟫ *σs* = $combine_d$-*Seq*-*ε* $A_0$ $A_1$ *hd* (*hd* ∘ *tl*) *σs*›
‹*ε* ⟪$B_0$ $_{nd}$⊗;✓$_{Pairlist}$ $B_1$⟫ *σs* = $combine_{nd}$-*Seq*-*ε* $B_0$ $B_1$ *hd* (*hd* ∘ *tl*) *σs*›
⟨*proof*⟩

**lemma** *ε*-$combine_{Pair}$-$Seq_{ptick}$ :
‹*ε* ⟪$A_0$ $_d$⊗;✓$_{Pair}$ $A_1$⟫ *σs* = $combine_d$-*Seq*-*ε* $A_0$ $A_1$ *fst snd σs*›

‹$\varepsilon$ $\langle\!\langle B_0\ {}_{nd}\otimes;_{\checkmark Pair}\ B_1\rangle\!\rangle\ \sigma s = combine_{nd}\text{-}Seq\text{-}\varepsilon\ B_0\ B_1\ fst\ snd\ \sigma s$›
⟨*proof*⟩

**lemma** $\varepsilon\text{-}combine_{ListslenL}\text{-}Seq_{ptick}$ :
‹$\varepsilon$ $\langle\!\langle A_0\ {}_{d}\otimes len_0;_{ListslenL}\ A_1\rangle\!\rangle\ \sigma s = combine_{d}\text{-}Seq\text{-}\varepsilon\ A_0\ A_1\ (take\ len_0)\ (drop\ len_0)\ \sigma s$›
‹$\varepsilon$ $\langle\!\langle B_0\ {}_{nd}\otimes len_0;_{ListslenL}\ B_1\rangle\!\rangle\ \sigma s = combine_{nd}\text{-}Seq\text{-}\varepsilon\ B_0\ B_1\ (take\ len_0)\ (drop\ len_0)\ \sigma s$›
⟨*proof*⟩

**lemma** $\varepsilon\text{-}combine_{Rlist}\text{-}Seq_{ptick}$ :
‹$\varepsilon$ $\langle\!\langle A_0\ {}_{d}\otimes;_{\checkmark Rlist}\ A_1\rangle\!\rangle\ \sigma s = combine_{d}\text{-}Seq\text{-}\varepsilon\ A_0\ A_1\ hd\ tl\ \sigma s$›
‹$\varepsilon$ $\langle\!\langle B_0\ {}_{nd}\otimes;_{\checkmark Rlist}\ B_1\rangle\!\rangle\ \sigma s = combine_{nd}\text{-}Seq\text{-}\varepsilon\ B_0\ B_1\ hd\ tl\ \sigma s$›
⟨*proof*⟩

### 9.4.1 Reachability

**lemma** $\mathcal{R}_d\text{-}combine_{dListslenL}\text{-}Seq_{ptick}\text{-}subset$:
‹$\mathcal{R}_d$ $\langle\!\langle A_0\ {}_{d}\otimes len_0;_{ListslenL}\ A_1\rangle\!\rangle\ (s_0\ @\ s_1) \subseteq \{t_0\ @\ t_1\ |t_0\ t_1.\ t_0 \in \mathcal{R}_d\ A_0\ s_0\}$› (**is** ‹$?S_A \subseteq$ -›)
**if** *same-length-*$\mathcal{R}_d$: ‹$\bigwedge t_0.\ t_0 \in \mathcal{R}_d\ A_0\ s_0 \Longrightarrow length\ t_0 = len_0$›
⟨*proof*⟩

**lemma** $\mathcal{R}_{nd}\text{-}combine_{ndListslenL}\text{-}Seq_{ptick}\text{-}subset$:
‹$\mathcal{R}_{nd}$ $\langle\!\langle A_0\ {}_{nd}\otimes len_0;_{ListslenL}\ A_1\rangle\!\rangle\ (s_0\ @\ s_1) \subseteq \{t_0\ @\ t_1\ |t_0\ t_1.\ t_0 \in \mathcal{R}_{nd}\ A_0\ s_0\}$› (**is** ‹$?S_A \subseteq$ -›)
**if** *same-length-*$\mathcal{R}_{nd}$: ‹$\bigwedge t_0.\ t_0 \in \mathcal{R}_{nd}\ A_0\ s_0 \Longrightarrow length\ t_0 = len_0$›
⟨*proof*⟩

**lemma** $\mathcal{R}_d\text{-}combine_{dPairlist}\text{-}Seq_{ptick}\text{-}subset$:
‹$\mathcal{R}_d$ $\langle\!\langle A_0\ {}_{d}\otimes;_{\checkmark Pairlist}\ A_1\rangle\!\rangle\ [s_0,\ s_1] \subseteq \{[t_0,\ t_1]\ |t_0\ t_1.\ t_0 \in \mathcal{R}_d\ A_0\ s_0\}$› (**is** ‹$?S_A \subseteq$ -›)
⟨*proof*⟩

**lemma** $\mathcal{R}_{nd}\text{-}combine_{ndPairlist}\text{-}Seq_{ptick}\text{-}subset$:
‹$\mathcal{R}_{nd}$ $\langle\!\langle A_0\ {}_{nd}\otimes;_{\checkmark Pairlist}\ A_1\rangle\!\rangle\ [s_0,\ s_1] \subseteq \{[t_0,\ t_1]\ |t_0\ t_1.\ t_0 \in \mathcal{R}_{nd}\ A_0\ s_0\}$› (**is** ‹$?S_A \subseteq$ -›)
⟨*proof*⟩

**lemma** $\mathcal{R}_d\text{-}combine_{dPair}\text{-}Seq_{ptick}\text{-}subset$:
‹$\mathcal{R}_d$ $\langle\!\langle A_0\ {}_{d}\otimes;_{\checkmark Pair}\ A_1\rangle\!\rangle\ (s_0,\ s_1) \subseteq \{(t_0,\ t_1)\ |t_0\ t_1.\ t_0 \in \mathcal{R}_d\ A_0\ s_0\}$› (**is** ‹$?S_A \subseteq$ -›)
⟨*proof*⟩

**lemma** $\mathcal{R}_{nd}\text{-}combine_{ndPair}\text{-}Seq_{ptick}\text{-}subset$:
‹$\mathcal{R}_{nd}$ $\langle\!\langle A_0\ {}_{nd}\otimes;_{\checkmark Pair}\ A_1\rangle\!\rangle\ (s_0,\ s_1) \subseteq \{(t_0,\ t_1)\ |t_0\ t_1.\ t_0 \in \mathcal{R}_{nd}\ A_0\ s_0\}$› (**is** ‹$?S_A \subseteq$ -›)

⟨*proof*⟩

**lemma** $\mathcal{R}_d$-*combine*$_{dRlist}$-*Seq*$_{ptick}$-*subset*:
‹$\mathcal{R}_d$ ⟪$A_0$ $_d\otimes;_{\checkmark Rlist}$ $A_1$⟫ $(s_0 \# s_1) \subseteq \{t_0 \# t_1 \mid t_0\ t_1.\ t_0 \in \mathcal{R}_d\ A_0\ s_0\}$› (**is** ‹$?S_A \subseteq$ -›)
⟨*proof*⟩

**lemma** $\mathcal{R}_{nd}$-*combine*$_{ndRlist}$-*Seq*$_{ptick}$-*subset*:
‹$\mathcal{R}_{nd}$ ⟪$A_0$ $_{nd}\otimes;_{\checkmark Rlist}$ $A_1$⟫ $(s_0 \# s_1) \subseteq \{t_0 \# t_1 \mid t_0\ t_1.\ t_0 \in \mathcal{R}_{nd}\ A_0\ s_0\}$› (**is** ‹$?S_A \subseteq$ -›)
⟨*proof*⟩

## 9.5 Normalization

**lemma** $\tau$-*combine*$_{Pairlist}$-*Seq*$_{ptick}$-*behaviour*:
‹$\tau$ ⟪⟪$A_0$ $_d\otimes;_{\checkmark Pairlist}$ $A_1$⟫⟫$_{d\hookrightarrow nd}$ $[s_0,\ s_1]$ $e$ = $\tau$ ⟪⟪$A_0$⟫$_{d\hookrightarrow nd}$ $_{nd}\otimes;_{\checkmark Pairlist}$ $(\lambda r.$ ⟪$A_1$ $r$⟫$_{d\hookrightarrow nd})$⟫ $[s_0,\ s_1]$ $e$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{Pair}$-*Seq*$_{ptick}$-*behaviour*:
‹$\tau$ ⟪⟪$A_0$ $_d\otimes;_{\checkmark Pair}$ $A_1$⟫⟫$_{d\hookrightarrow nd}$ $(s_0,\ s_1)$ $e$ = $\tau$ ⟪⟪$A_0$⟫$_{d\hookrightarrow nd}$ $_{nd}\otimes;_{\checkmark Pair}$ $(\lambda r.$ ⟪$A_1$ $r$⟫$_{d\hookrightarrow nd})$⟫ $(s_0,\ s_1)$ $e$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{ListslenL}$-*Seq*$_{ptick}$-*behaviour*:
‹$\tau$ ⟪⟪$A_0$ $_d\otimes len_0;_{ListslenL}$ $A_1$⟫⟫$_{d\hookrightarrow nd}$ $(\sigma s_0$ @ $\sigma s_1)$ $e$ = $\tau$ ⟪⟪$A_0$⟫$_{d\hookrightarrow nd}$ $_{nd}\otimes len_0;_{ListslenL}$ $(\lambda r.$ ⟪$A_1$ $r$⟫$_{d\hookrightarrow nd})$⟫ $(\sigma s_0$ @ $\sigma s_1)$ $e$›
⟨*proof*⟩

**lemma** $\tau$-*combine*$_{Rlist}$-*Seq*$_{ptick}$-*behaviour*:
‹$\tau$ ⟪⟪$A_0$ $_d\otimes;_{\checkmark Rlist}$ $A_1$⟫⟫$_{d\hookrightarrow nd}$ $(s_0 \# \sigma s_1)$ $e$ = $\tau$ ⟪⟪$A_0$⟫$_{d\hookrightarrow nd}$ $_{nd}\otimes;_{\checkmark Rlist}$ $(\lambda r.$ ⟪$A_1$ $r$⟫$_{d\hookrightarrow nd})$⟫ $(s_0 \# \sigma s_1)$ $e$›
⟨*proof*⟩

Behaviour of normalisations

**lemma** $P_{SKIPS}$-*combine*$_{Pairlist}$-*Seq*$_{ptick}$-*behaviour*:
‹$P_{SKIPS}$⟪⟪$A_0$ $_d\otimes;_{\checkmark Pairlist}$ $A_1$⟫⟫$_d$ $[s_0,\ s_1]$ = $P_{SKIPS}$⟪⟪⟪$A_0$⟫$_{d\hookrightarrow nd}$ $_{nd}\otimes;_{\checkmark Pairlist}$ $(\lambda r.$ ⟪$A_1$ $r$⟫$_{d\hookrightarrow nd})$⟫⟫$_{nd}$ $[s_0,\ s_1]$›
⟨*proof*⟩

**lemma** *P*-*combine*$_{Pairlist}$-*Seq*$_{ptick}$-*behaviour*:
‹$P$⟪⟪$A_0$ $_d\otimes;_{\checkmark Pairlist}$ $A_1$⟫⟫$_d$ $[s_0,\ s_1]$ = $P$⟪⟪⟪$A_0$⟫$_{d\hookrightarrow nd}$ $_{nd}\otimes;_{\checkmark Pairlist}$ $(\lambda r.$ ⟪$A_1$ $r$⟫$_{d\hookrightarrow nd})$⟫⟫$_{nd}$ $[s_0,\ s_1]$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*combine*$_{Pair}$-*Seq*$_{ptick}$-*behaviour*:

‹$P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes;_{\checkmark Pair}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (s_0,\ s_1) = P_{SKIPS}\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes;_{\checkmark Pair}\ (\lambda r.\ \langle\!\langle A_1\ r\rangle\!\rangle_{d\hookrightarrow nd})\rangle\!\rangle\rangle\!\rangle_{nd}\ (s_0,\ s_1)$›
⟨*proof*⟩

**lemma** *P-combine*$_{Pair}$*-Seq*$_{ptick}$*-behaviour*:
‹$P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes;_{\checkmark Pair}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (s_0,\ s_1) = P\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes;_{\checkmark Pair}\ (\lambda r.\ \langle\!\langle A_1\ r\rangle\!\rangle_{d\hookrightarrow nd})\rangle\!\rangle\rangle\!\rangle_{nd}\ (s_0,\ s_1)$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-combine*$_{Rlist}$*-Seq*$_{ptick}$*-behaviour*:
‹$P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes;_{\checkmark Rlist}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (s_0\ \#\ s_1) = P_{SKIPS}\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes;_{\checkmark Rlist}\ (\lambda r.\ \langle\!\langle A_1\ r\rangle\!\rangle_{d\hookrightarrow nd})\rangle\!\rangle\rangle\!\rangle_{nd}\ (s_0\ \#\ s_1)$›
⟨*proof*⟩

**lemma** *P-combine*$_{Rlist}$*-Seq*$_{ptick}$*-behaviour*:
‹$P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes;_{\checkmark Rlist}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (s_0\ \#\ s_1) = P\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes;_{\checkmark Rlist}\ (\lambda r.\ \langle\!\langle A_1\ r\rangle\!\rangle_{d\hookrightarrow nd})\rangle\!\rangle\rangle\!\rangle_{nd}\ (s_0\ \#\ s_1)$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$*-combine*$_{ListslenL}$*-Seq*$_{ptick}$*-behaviour*:
‹$P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes len_0;_{ListslenL}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (\sigma s_0\ @\ \sigma s_1) = P_{SKIPS}\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes len_0;_{ListslenL}\ (\lambda r.\ \langle\!\langle A_1\ r\rangle\!\rangle_{d\hookrightarrow nd})\rangle\!\rangle\rangle\!\rangle_{nd}\ (\sigma s_0\ @\ \sigma s_1)$›
**if** ‹$\bigwedge\sigma s_0'.\ \sigma s_0' \in \mathcal{R}_d\ A_0\ \sigma s_0 \Longrightarrow length\ \sigma s_0' = len_0$›
⟨*proof*⟩

**lemma** *P-combine*$_{ListslenL}$*-Seq*$_{ptick}$*-behaviour*:
‹$P\langle\!\langle\langle\!\langle A_0\ {}_d\otimes len_0;_{ListslenL}\ A_1\rangle\!\rangle\rangle\!\rangle_d\ (\sigma s_0\ @\ \sigma s_1) = P\langle\!\langle\langle\!\langle\langle\!\langle A_0\rangle\!\rangle_{d\hookrightarrow nd}\ {}_{nd}\otimes len_0;_{ListslenL}\ (\lambda r.\ \langle\!\langle A_1\ r\rangle\!\rangle_{d\hookrightarrow nd})\rangle\!\rangle\rangle\!\rangle_{nd}\ (\sigma s_0\ @\ \sigma s_1)$›
**if** ‹$\bigwedge\sigma t_0.\ \sigma t_0 \in \mathcal{R}_d\ A_0\ \sigma s_0 \Longrightarrow length\ \sigma t_0 = len_0$›
⟨*proof*⟩

# Chapter 10

# Compactification of Sequential Composition Generalized

## 10.1 Iterated Combine

### 10.1.1 Definitions

**fun** *iterated-combine*$_d$*-Seq*$_{ptick}$ :: ‹$[('r \Rightarrow ('\sigma, 'e, 'r)\ A_d)\ list, 'r] \Rightarrow ('\sigma\ list, 'e, 'r)\ A_d$› (‹$\langle\!\langle_d\bigotimes;_{\checkmark}$ -$\rangle\!\rangle$› [*0*])
  **where** ‹$\langle\!\langle_d\bigotimes;_{\checkmark}\ []\rangle\!\rangle\ r = (\!|\tau = \lambda\sigma s\ a.\ \Diamond, \omega = \lambda\sigma s.\ \lfloor r\rfloor|\!)$›
  | ‹$\langle\!\langle_d\bigotimes;_{\checkmark}\ [A_0]\rangle\!\rangle\ r = {}_d\langle\!\langle A_0\ r\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$›
  | ‹$\langle\!\langle_d\bigotimes;_{\checkmark}\ A_0\ \#\ A_1\ \#\ As\rangle\!\rangle\ r = \langle\!\langle A_0\ r\ {}_d\otimes;_{\checkmark Rlist}\ \langle\!\langle_d\bigotimes;_{\checkmark}\ A_1\ \#\ As\rangle\!\rangle\rangle\!\rangle$›

**fun** *iterated-combine*$_{nd}$*-Seq*$_{ptick}$ :: ‹$[('r \Rightarrow ('\sigma, 'e, 'r)\ A_{nd})\ list, 'r] \Rightarrow ('\sigma\ list, 'e, 'r)\ A_{nd}$› (‹$\langle\!\langle_{nd}\bigotimes;_{\checkmark}$ -$\rangle\!\rangle$› [*0*])
  **where** ‹$\langle\!\langle_{nd}\bigotimes;_{\checkmark}\ []\rangle\!\rangle\ r = (\!|\tau = \lambda\sigma s\ a.\ \{\}, \omega = \lambda\sigma s.\ \{r\}|\!)$›
  | ‹$\langle\!\langle_{nd}\bigotimes;_{\checkmark}\ [A_0]\rangle\!\rangle\ r = {}_{nd}\langle\!\langle A_0\ r\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}$›
  | ‹$\langle\!\langle_{nd}\bigotimes;_{\checkmark}\ A_0\ \#\ A_1\ \#\ As\rangle\!\rangle\ r = \langle\!\langle A_0\ r\ {}_{nd}\otimes;_{\checkmark Rlist}\ \langle\!\langle_{nd}\bigotimes;_{\checkmark}\ A_1\ \#\ As\rangle\!\rangle\rangle\!\rangle$›

**lemma** *iterated-combine*$_d$*-Seq*$_{ptick}$*-simps-bis*: ‹$As \neq [] \Longrightarrow \langle\!\langle_d\bigotimes;_{\checkmark}\ A_0\ \#\ As\rangle\!\rangle\ r = \langle\!\langle A_0\ r\ {}_d\otimes;_{\checkmark Rlist}\ \langle\!\langle_d\bigotimes;_{\checkmark}\ As\rangle\!\rangle\rangle\!\rangle$›
  **and** *iterated-combine*$_{nd}$*-Seq*$_{ptick}$*-simps-bis*: ‹$Bs \neq [] \Longrightarrow \langle\!\langle_{nd}\bigotimes;_{\checkmark}\ B_0\ \#\ Bs\rangle\!\rangle\ r = \langle\!\langle B_0\ r\ {}_{nd}\otimes;_{\checkmark Rlist}\ \langle\!\langle_{nd}\bigotimes;_{\checkmark}\ Bs\rangle\!\rangle\rangle\!\rangle$›
  ⟨*proof*⟩

### 10.1.2 First Results

**lemma** *combine*$_{ListslenL}$*-Seq*$_{ptick}$*-combine*$_{Rlist}$*-Seq*$_{ptick}$*-eq*:
  ‹$\varepsilon\ \langle\!\langle_d\langle\!\langle A_0\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ {}_d\otimes 1;_{ListslenL}\ A_1\rangle\!\rangle\ (s_0\ \#\ \sigma s) = \varepsilon\ \langle\!\langle A_0\ {}_d\otimes;_{\checkmark Rlist}\ A_1\rangle\!\rangle\ (s_0\ \#\ \sigma s)$›
  ‹$\tau\ \langle\!\langle_d\langle\!\langle A_0\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ {}_d\otimes 1;_{ListslenL}\ A_1\rangle\!\rangle\ (s_0\ \#\ \sigma s)\ e = \tau\ \langle\!\langle A_0\ {}_d\otimes;_{\checkmark Rlist}\ A_1\rangle\!\rangle\ (s_0\ \#\ \sigma s)\ e$›
  ‹$\varepsilon\ \langle\!\langle_{nd}\langle\!\langle B_0\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ {}_{nd}\otimes 1;_{ListslenL}\ B_1\rangle\!\rangle\ (s_0\ \#\ \sigma s) = \varepsilon\ \langle\!\langle B_0\ {}_{nd}\otimes;_{\checkmark Rlist}\ B_1\rangle\!\rangle\ (s_0\ \#$

$\sigma s)$›
‹$\tau$ $\langle\!\langle_{nd}\langle\!\langle B_0\rangle\!\rangle_{\sigma\hookrightarrow\sigma s}\ {}_{nd}\otimes 1;_{ListslenL}\ B_1\rangle\!\rangle$ $(s_0 \# \sigma s)$ $e = \tau$ $\langle\!\langle B_0\ {}_{nd}\otimes;_{\checkmark Rlist}\ B_1\rangle\!\rangle$ $(s_0 \# \sigma s)$ $e$›
⟨*proof*⟩

**lemma** *combine*$_{Pairlist}$-*Seq*$_{ptick}$-*and-iterated-combine-Seq*$_{ptick}$-*eq*:
‹$\varepsilon$ $\langle\!\langle A_0\ r\ {}_{d}\otimes;_{\checkmark Pairlist}\ A_1\rangle\!\rangle$ $[s_0,\ s_1] = \varepsilon$ $(\langle\!\langle_{d}\bigotimes;_{\checkmark}\ [A_0,\ A_1]\rangle\!\rangle\ r)$ $[s_0,\ s_1]$›
‹$\tau$ $\langle\!\langle A_0\ r\ {}_{d}\otimes;_{\checkmark Pairlist}\ A_1\rangle\!\rangle$ $[s_0,\ s_1]$ $e = \tau$ $(\langle\!\langle_{d}\bigotimes;_{\checkmark}\ [A_0,\ A_1]\rangle\!\rangle\ r)$ $[s_0,\ s_1]$ $e$›
‹$\varepsilon$ $\langle\!\langle B_0\ r\ {}_{nd}\otimes;_{\checkmark Pairlist}\ B_1\rangle\!\rangle$ $[s_0,\ s_1] = \varepsilon$ $(\langle\!\langle_{nd}\bigotimes;_{\checkmark}\ [B_0,\ B_1]\rangle\!\rangle\ r)$ $[s_0,\ s_1]$›
‹$\tau$ $\langle\!\langle B_0\ r\ {}_{nd}\otimes;_{\checkmark Pairlist}\ B_1\rangle\!\rangle$ $[s_0,\ s_1]$ $e = \tau$ $(\langle\!\langle_{nd}\bigotimes;_{\checkmark}\ [B_0,\ B_1]\rangle\!\rangle\ r)$ $[s_0,\ s_1]$ $e$›
⟨*proof*⟩

**lemma** *combine*$_{Pairlist}$-*Seq*$_{ptick}$-*and-combine*$_{Rlist}$-*Seq*$_{ptick}$-*eq* :
‹$\varepsilon$ $\langle\!\langle A_0\ {}_{d}\otimes;_{\checkmark Pairlist}\ A_1\rangle\!\rangle$ $[s_0,\ s_1] = \varepsilon$ $\langle\!\langle A_0\ {}_{d}\otimes;_{\checkmark Rlist}\ \langle\!\langle_{d}\bigotimes;_{\checkmark}\ [A_1]\rangle\!\rangle\rangle\!\rangle$ $[s_0,\ s_1]$›
‹$\tau$ $\langle\!\langle A_0\ {}_{d}\otimes;_{\checkmark Pairlist}\ A_1\rangle\!\rangle$ $[s_0,\ s_1]$ $e = \tau$ $\langle\!\langle A_0\ {}_{d}\otimes;_{\checkmark Rlist}\ \langle\!\langle_{d}\bigotimes;_{\checkmark}\ [A_1]\rangle\!\rangle\rangle\!\rangle$ $[s_0,\ s_1]$ $e$›
‹$\varepsilon$ $\langle\!\langle B_0\ {}_{nd}\otimes;_{\checkmark Pairlist}\ B_1\rangle\!\rangle$ $[s_0,\ s_1] = \varepsilon$ $\langle\!\langle B_0\ {}_{nd}\otimes;_{\checkmark Rlist}\ \langle\!\langle_{nd}\bigotimes;_{\checkmark}\ [B_1]\rangle\!\rangle\rangle\!\rangle$ $[s_0,\ s_1]$›
‹$\tau$ $\langle\!\langle B_0\ {}_{nd}\otimes;_{\checkmark Pairlist}\ B_1\rangle\!\rangle$ $[s_0,\ s_1]$ $e = \tau$ $\langle\!\langle B_0\ {}_{nd}\otimes;_{\checkmark Rlist}\ \langle\!\langle_{nd}\bigotimes;_{\checkmark}\ [B_1]\rangle\!\rangle\rangle\!\rangle$ $[s_0,\ s_1]$ $e$›
⟨*proof*⟩

### 10.1.3 Reachability

**lemma** *same-length-*$\tau$*-iterated-combine*$_d$-*Seq*$_{ptick}$ :
‹*length* $\sigma s$ = *length As* $\Longrightarrow$ $\lfloor\sigma t\rfloor = \tau$ $(\langle\!\langle_{d}\bigotimes;_{\checkmark}\ As\rangle\!\rangle\ r)$ $\sigma s$ $a$ $\Longrightarrow$ *length* $\sigma t$ = *length As*›
⟨*proof*⟩

**lemma** *same-length-*$\tau$*-iterated-combine*$_{nd}$-*Seq*$_{ptick}$ :
‹*length* $\sigma s$ = *length As* $\Longrightarrow$ $\sigma t \in \tau$ $(\langle\!\langle_{nd}\bigotimes;_{\checkmark}\ As\rangle\!\rangle\ r)$ $\sigma s$ $a$ $\Longrightarrow$ *length* $\sigma t$ = *length As*›
⟨*proof*⟩

**lemma** *same-length-*$\mathcal{R}_d$*-iterated-combine*$_d$-*Seq*$_{ptick}$ :
‹$\sigma t \in \mathcal{R}_d$ $(\langle\!\langle_{d}\bigotimes;_{\checkmark}\ As\rangle\!\rangle\ r)$ $\sigma s$ $\Longrightarrow$ *length* $\sigma s$ = *length As* $\Longrightarrow$ *length* $\sigma t$ = *length As*›
⟨*proof*⟩

**lemma** *same-length-*$\mathcal{R}_{nd}$*-iterated-combine*$_{nd}$-*Seq*$_{ptick}$ :
‹$\sigma t \in \mathcal{R}_{nd}$ $(\langle\!\langle_{nd}\bigotimes;_{\checkmark}\ As\rangle\!\rangle\ r)$ $\sigma s$ $\Longrightarrow$ *length* $\sigma s$ = *length As* $\Longrightarrow$ *length* $\sigma t$ = *length As*›
⟨*proof*⟩

### 10.1.4 Transmission of Properties

**lemma** $\varrho$*-disjoint-*$\varepsilon$*-transmission-to-iterated-combine-Seq*$_{ptick}$ :
‹*As* $\neq$ [] $\Longrightarrow$ $\varrho$*-disjoint-*$\varepsilon$ (*(last As)* $r$) $\Longrightarrow$ $\varrho$*-disjoint-*$\varepsilon$ $(\langle\!\langle_{d}\bigotimes;_{\checkmark}\ As\rangle\!\rangle\ r)$›

‹$Bs \neq [] \Longrightarrow \varrho$-*disjoint*-$\varepsilon$ ((*last Bs*) $r$) $\Longrightarrow \varrho$-*disjoint*-$\varepsilon$ ($\langle\!\langle_{nd}\bigotimes;_{\checkmark}$ $Bs\rangle\!\rangle$ $r$)›
⟨*proof*⟩

### 10.1.5 Normalization

**lemma** $\omega$-*iterated-combine-Seq*$_{ptick}$-*det-ndet-conv*:
‹$\omega$ ($\langle\!\langle_{nd}\bigotimes;_{\checkmark}$ *map* ($\lambda A$ $r$. $\langle\!\langle A\ r\rangle\!\rangle_{d\hookrightarrow nd}$) $As\rangle\!\rangle$ $r$) $\sigma s = \omega$ $\langle\!\langle\langle\!\langle_{d}\bigotimes;_{\checkmark}$ $As\rangle\!\rangle$ $r\rangle\!\rangle_{d\hookrightarrow nd}$ $\sigma s$›
⟨*proof*⟩

**lemma** $\varrho$-*iterated-combine-Seq*$_{ptick}$-*det-ndet-conv* :
‹$\varrho$ ($\langle\!\langle_{nd}\bigotimes;_{\checkmark}$ *map* ($\lambda A$ $r$. $\langle\!\langle A\ r\rangle\!\rangle_{d\hookrightarrow nd}$) $As\rangle\!\rangle$ $r$) = $\varrho$ $\langle\!\langle\langle\!\langle_{d}\bigotimes;_{\checkmark}$ $As\rangle\!\rangle$ $r\rangle\!\rangle_{d\hookrightarrow nd}$›
⟨*proof*⟩

**lemma** $\tau$-*iterated-combine-Seq*$_{ptick}$-*behaviour*:
‹*length* $\sigma s$ = *length* $As \Longrightarrow$
$\tau$ $\langle\!\langle\langle\!\langle_{d}\bigotimes;_{\checkmark}$ $As\rangle\!\rangle$ $r\rangle\!\rangle_{d\hookrightarrow nd}$ $\sigma s$ $e$ = $\tau$ ($\langle\!\langle_{nd}\bigotimes;_{\checkmark}$ *map* ($\lambda A$ $r$. $\langle\!\langle A\ r\rangle\!\rangle_{d\hookrightarrow nd}$) $As\rangle\!\rangle$ $r$) $\sigma s$ $e$›
⟨*proof*⟩

**lemma** $P_{SKIPS}$-*iterated-combine-Seq*$_{ptick}$-*behaviour*:
**assumes** *same-length*: ‹*length* $\sigma s$ = *length* $As$›
**shows** ‹$P_{SKIPS}\langle\!\langle\langle\!\langle_{d}\bigotimes;_{\checkmark}$ $As\rangle\!\rangle$ $r\rangle\!\rangle_{d}$ $\sigma s$ = $P_{SKIPS}\langle\!\langle\langle\!\langle_{nd}\bigotimes;_{\checkmark}$ *map* ($\lambda A$ $r$. $\langle\!\langle A\ r\rangle\!\rangle_{d\hookrightarrow nd}$) $As\rangle\!\rangle$ $r\rangle\!\rangle_{nd}$ $\sigma s$›
⟨*proof*⟩

**lemma** *P-iterated-combine-Seq*$_{ptick}$-*behaviour*:
**assumes** *same-length*: ‹*length* $\sigma s$ = *length* $As$›
**shows** ‹$P\langle\!\langle\langle\!\langle_{d}\bigotimes;_{\checkmark}$ $As\rangle\!\rangle$ $r\rangle\!\rangle_{d}$ $\sigma s$ = $P\langle\!\langle\langle\!\langle_{nd}\bigotimes;_{\checkmark}$ *map* ($\lambda A$ $r$. $\langle\!\langle A\ r\rangle\!\rangle_{d\hookrightarrow nd}$) $As\rangle\!\rangle$ $r\rangle\!\rangle_{nd}$ $\sigma s$›
⟨*proof*⟩

## 10.2 Compactification Theorems

### 10.2.1 Binary

#### Pair

**lemma** $P_{SKIPS}$-*nd-combine*$_{Pair}$-*Seq*$_{ptick}$ :
**fixes** $A_0$ $A_1$
**assumes** *at-most-1-elem-term* : ‹*at-most-1-elem-term* $A_0$›
— This assumption is necessary in the new setup, otherwise the result is not always a Procomaton (for example if $\omega$ $A_0$ $\sigma_0$ = *UNIV*, we have $P$ $\sigma_0$ $;_{\checkmark}$ $Q$ $\sigma_1$ = *GlobalNdet UNIV* ($Q$ $\sigma_1$)).

**defines** *A-def*: ‹$A \equiv \langle\!\langle A_0\ {}_{nd}\otimes;_{\checkmark Pair}\ A_1\rangle\!\rangle$›
**defines** *P-def*: ‹$P \equiv P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}$›
**and** *Q-def*: ‹$Q \equiv \lambda\sigma_1$ $r$. $P_{SKIPS}\langle\!\langle A_1\ r\rangle\!\rangle_{nd}$ $\sigma_1$›

**and** *S-def*: ‹$S \equiv P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}$›
**shows** ‹$P\ \sigma_0\ ;_{\checkmark}\ Q\ \sigma_1 = S\ (\sigma_0,\ \sigma_1)$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$-*d-combine*$_{Pair}$-*Seq*$_{ptick}$ :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ ;_{\checkmark}\ (\lambda r.\ P_{SKIPS}\langle\!\langle A_1\ r\rangle\!\rangle_d\ \sigma_1) = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes;_{\checkmark Pair}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma_0,\ \sigma_1)$›
⟨*proof*⟩

### Pairlist

**lemma** $P_{SKIPS}$-*nd-combine*$_{Pairlist}$-*Seq*$_{ptick}$ :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ ;_{\checkmark}\ (\lambda r.\ P_{SKIPS}\langle\!\langle A_1\ r\rangle\!\rangle_{nd}\ \sigma_1) = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes;_{\checkmark Pairlist}\ A_1\rangle\!\rangle\!\rangle_{nd}\ [\sigma_0,\ \sigma_1]$›
**if** ‹*at-most-1-elem-term* $A_0$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$-*d-combine*$_{Pairlist}$-*Seq*$_{ptick}$ :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ ;_{\checkmark}\ (\lambda r.\ P_{SKIPS}\langle\!\langle A_1\ r\rangle\!\rangle_d\ \sigma_1) = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes;_{\checkmark Pairlist}\ A_1\rangle\!\rangle\!\rangle_d\ [\sigma_0,\ \sigma_1]$›
⟨*proof*⟩

### Rlist

**lemma** $P_{SKIPS}$-*nd-combine*$_{Rlist}$-*Seq*$_{ptick}$ :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma_0\ ;_{\checkmark}\ (\lambda r.\ P_{SKIPS}\langle\!\langle A_1\ r\rangle\!\rangle_{nd}\ \sigma s) = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes;_{\checkmark Rlist}\ A_1\rangle\!\rangle\!\rangle_{nd}\ (\sigma_0\ \#\ \sigma s)$›
**if** ‹*at-most-1-elem-term* $A_0$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$-*d-combine*$_{Rlist}$-*Seq*$_{ptick}$ :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_d\ \sigma_0\ ;_{\checkmark}\ (\lambda r.\ P_{SKIPS}\langle\!\langle A_1\ r\rangle\!\rangle_d\ \sigma s) = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes;_{\checkmark Rlist}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma_0\ \#\ \sigma s)$›
⟨*proof*⟩

## 10.2.2 ListslenL

**lemma** $P_{SKIPS}$-*nd-combine*$_{ListslenL}$-*Seq*$_{ptick}$ :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_{nd}\ \sigma s_0\ ;_{\checkmark}\ (\lambda r.\ P_{SKIPS}\langle\!\langle A_1\ r\rangle\!\rangle_{nd}\ \sigma s_1) = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_{nd}\otimes len_0;_{ListslenL}\ A_1\rangle\!\rangle\!\rangle_{nd}\ (\sigma s_0\ @\ \sigma s_1)$›
**if** *same-length-reach* : ‹$\bigwedge\sigma s_0'.\ \sigma s_0' \in \mathcal{R}_{nd}\ A_0\ \sigma s_0 \Longrightarrow length\ \sigma s_0' = len_0$›
**and** ‹*at-most-1-elem-term* $A_0$›
⟨*proof*⟩

**corollary** $P_{SKIPS}$-*d-combine*$_{ListslenL}$-*Seq*$_{ptick}$ :
‹$P_{SKIPS}\langle\!\langle A_0\rangle\!\rangle_d\ \sigma s_0\ ;_{\checkmark}\ (\lambda r.\ P_{SKIPS}\langle\!\langle A_1\ r\rangle\!\rangle_d\ \sigma s_1) = P_{SKIPS}\langle\!\langle\langle\!\langle A_0\ {}_d\otimes len_0;_{ListslenL}\ A_1\rangle\!\rangle\!\rangle_d\ (\sigma s_0\ @\ \sigma s_1)$›
**if** *same-length-reach* : ‹$\bigwedge\sigma s_0'.\ \sigma s_0' \in \mathcal{R}_d\ A_0\ \sigma s_0 \Longrightarrow length\ \sigma s_0' = len_0$›
⟨*proof*⟩

### 10.2.3 Multiple

**theorem** $P_{SKIPS}$*-nd-compactification-Seq*$_{ptick}$:
‹$[\![$*length* $\sigma s$ = *length As*; $\bigwedge A\ r.\ A \in$ *set* (*butlast As*) $\Longrightarrow$ *at-most-1-elem-term* ($A$ $r$)$]\!] \Longrightarrow$
*SEQ*$_{\checkmark}$ ($\sigma$, $A$) $\in$@ *zip* $\sigma s$ *As*. ($\lambda r$. $P_{SKIPS}\langle\!\langle A\ r\rangle\!\rangle_{nd}\ \sigma$) = ($\lambda r$. $P_{SKIPS}\langle\!\langle\langle\!\langle_{nd}\bigotimes;_{\checkmark}$ *As*$\rangle\!\rangle$ $r\rangle\!\rangle_{nd}\ \sigma s$)›
⟨*proof*⟩

**corollary** $P_{SKIPS}$*-d-compactification-Seq*$_{ptick}$:
‹*length* $\sigma s$ = *length As* $\Longrightarrow$
*SEQ*$_{\checkmark}$ ($\sigma$, $A$) $\in$@ *zip* $\sigma s$ *As*. ($\lambda r$. $P_{SKIPS}\langle\!\langle A\ r\rangle\!\rangle_{d}\ \sigma$) = ($\lambda r$. $P_{SKIPS}\langle\!\langle\langle\!\langle_{d}\bigotimes;_{\checkmark}$ *As*$\rangle\!\rangle$ $r\rangle\!\rangle_{d}\ \sigma s$)›
⟨*proof*⟩

# Chapter 11

# Application : May Philosophers dine ?

## 11.1 Preliminaries

### 11.1.1 Preliminary lemmas for proof automation

**lemma** *Suc-mod*: ‹$n > 1 \Longrightarrow i \neq Suc\ i\ mod\ n$›
⟨*proof*⟩

**lemmas** *suc-mods* = *Suc-mod Suc-mod*[*symmetric*]

**lemma** *l-suc*: ‹$n > 1 \Longrightarrow \neg\ n \leq Suc\ 0$›
⟨*proof*⟩

**lemma** *minus-suc*: ‹$n > 0 \Longrightarrow n - Suc\ 0 \neq n$›
⟨*proof*⟩

**declare** *Un-insert-right*[*simp del*] *Un-insert-left*[*simp del*]

## 11.2 The dining processes definition

**context** *DiningPhilosophers* **begin**

**lemma** *RPHIL-restriction-fix-def*:
‹$RPHIL\ i = (\upsilon\ X.\ picks\ i\ i \rightarrow picks\ i\ ((i - 1)\ mod\ N) \rightarrow$
$putsdown\ i\ ((i - 1)\ mod\ N) \rightarrow putsdown\ i\ i \rightarrow X)$›
⟨*proof*⟩

**lemma** *LPHIL0-restriction-fix-def*:
‹$LPHIL0 = (\upsilon\ X.\ picks\ 0\ (N - 1) \rightarrow picks\ 0\ 0 \rightarrow$
$putsdown\ 0\ 0 \rightarrow putsdown\ 0\ (N - 1) \rightarrow X)$›
⟨*proof*⟩

**lemma** *FORK-restriction-fix-def*:
‹$FORK\ i = (\upsilon\ X.\ (picks\ i\ i \rightarrow putsdown\ i\ i \rightarrow X)\ \Box$
$(picks\ ((i + 1)\ mod\ N)\ i \rightarrow putsdown\ ((i + 1)\ mod\ N)\ i \rightarrow X))$›
⟨*proof*⟩

### 11.2.1 Unfolding rules

**lemmas** *RPHIL-rec* = *cont-process-rec*[*OF RPHIL-def*[*THEN meta-eq-to-obj-eq*], *simplified*]
**and** *LPHIL0-rec* = *cont-process-rec*[*OF LPHIL0-def*[*THEN meta-eq-to-obj-eq*], *simplified*]
**and** *FORK-rec* = *cont-process-rec*[*OF FORK-def*[*THEN meta-eq-to-obj-eq*], *simplified*]

## 11.3 Translation into normal form

**lemma** *N-pos*[*simp*]: ‹$N > 0$›
⟨*proof*⟩

**lemmas** *N-pos-simps*[*simp*] = *suc-mods*[*OF N-g1*] *l-suc*[*OF N-g1*] *minus-suc*[*OF N-pos*]

### 11.3.1 *FORK*, *LPHIL0* and *RPHIL* are normalizable

Definition of one *fork* and one *philosopher* automata

**type-synonym** $id_{fork} = nat$
**type-synonym** $\sigma_{fork} = nat$
**type-synonym** $id_{phil} = nat$
**type-synonym** $\sigma_{phil} = nat$

**definition** *fork-A* :: ‹$id_{fork} \Rightarrow (\sigma_{fork},\ dining\text{-}event,\ unit)\ A_d$› (‹$A_f$›)
**where** ‹$A_f\ i \equiv recursive\text{-}constructor\text{-}A_d$
$[((0,\ picks\ i\ i),\ \lfloor 1 \rfloor),\ ((0,\ picks\ ((i + 1)\ mod\ N)\ i),\ \lfloor 2 \rfloor),$
$((1,\ putsdown\ i\ i),\ \lfloor 0 \rfloor),\ ((2,\ putsdown\ ((i + 1)\ mod\ N)\ i),\ \lfloor 0 \rfloor)]$
$(\lambda\sigma.\ \Diamond)$›

**definition** *rphil-A* :: ‹$id_{phil} \Rightarrow (\sigma_{phil},\ dining\text{-}event,\ unit)\ A_d$› (‹$A_{rp}$›)
**where** ‹$A_{rp}\ i \equiv recursive\text{-}constructor\text{-}A_d$
$[((0,\ picks\ i\ i),\ \lfloor 1 \rfloor),\ ((1,\ picks\ i\ ((i-1)\ mod\ N)),\ \lfloor 2 \rfloor),$
$((2,\ putsdown\ i\ ((i-1)\ mod\ N)),\ \lfloor 3 \rfloor),\ ((3,\ putsdown\ i\ i),\ \lfloor 0 \rfloor)]$
$(\lambda\sigma.\ \Diamond)$›

**definition** *lphil0-A* :: ‹$(\sigma_{phil},\ dining\text{-}event,\ unit)\ A_d$› (‹$A_{lp}$›)
**where** ‹$A_{lp} \equiv recursive\text{-}constructor\text{-}A_d$

$[((0,\ picks\ 0\ (N-1)),\ \lfloor 1 \rfloor),\ ((1,\ picks\ 0\ 0),\ \lfloor 2 \rfloor),$
$((2,\ putsdown\ 0\ 0),\ \lfloor 3 \rfloor),\ ((3,\ putsdown\ 0\ (N-1)),\ \lfloor 0 \rfloor)]\ (\lambda\sigma.\ \Diamond)$›

Definition and first properties of associated normal processes

**definition** *fork-P-d* :: ‹$id_{fork} \Rightarrow \sigma_{fork} \Rightarrow$ *dining-event process*› **where** ‹*fork-P-d* $i \equiv P\langle\!\langle A_f\ i \rangle\!\rangle_d$›
**definition** *rphil-P-d* :: ‹$id_{phil} \Rightarrow \sigma_{phil} \Rightarrow$ *dining-event process*› **where** ‹*rphil-P-d* $i \equiv P\langle\!\langle A_{rp}\ i \rangle\!\rangle_d$›
**definition** *lphil0-P-d* :: ‹$\sigma_{phil} \Rightarrow$ *dining-event process*› **where** ‹*lphil0-P-d* $\equiv P\langle\!\langle A_{lp} \rangle\!\rangle_d$›

**lemmas** *fork-P-d-rec* = *P-d-rec*[*of* ‹$A_f$ -›, *folded fork-P-d-def*]
**and** *rphil-P-d-rec* = *P-d-rec*[*of* ‹$A_{rp}$ -›, *folded rphil-P-d-def*]
**and** *lphil0-P-d-rec* = *P-d-rec*[*of* ‹$A_{lp}$›, *folded lphil0-P-d-def*]

**schematic-goal** *fork-ε*: ‹$\varepsilon\ (A_f\ i)\ \sigma = ?S$›
**and** *rphil-ε*: ‹$\varepsilon\ (A_{rp}\ i)\ \sigma = ?T$›
**and** *lphil0-ε*: ‹$\varepsilon\ A_{lp}\ \sigma = ?U$›
⟨*proof*⟩

**schematic-goal** *fork-τ*: ‹$\tau\ (A_f\ i)\ \sigma = ?S$›
**and** *rphil-τ*: ‹$\tau\ (A_{rp}\ i)\ \sigma = ?T$›
**and** *lphil0-τ*: ‹$\tau\ A_{lp}\ \sigma = ?U$›
⟨*proof*⟩

**corollary** *ev-id*$_{fork}x$: ‹$e \in \varepsilon\ (A_f\ i)\ \sigma \Longrightarrow fork\ e = i$›
**and** *rphil-phil*: ‹$e \in \varepsilon\ (A_{rp}\ i)\ \sigma \Longrightarrow phil\ e = i$›
**and** *lphil0-phil*: ‹$e \in \varepsilon\ A_{lp}\ \sigma \Longrightarrow phil\ e = 0$›
⟨*proof*⟩

**corollary** *ev-id*$_{phil}x$: ‹$i < n \Longrightarrow \sigma \in \varepsilon\ ((A_{lp}\ \#\ map\ A_{rp}\ [1..< n])\ !\ i)\ s \Longrightarrow phil\ \sigma = i$›
⟨*proof*⟩

**lemma** *indep-forks*: ‹$i \neq j \Longrightarrow \varepsilon\ (A_f\ i)\ \sigma \cap \varepsilon\ (A_f\ j)\ \sigma' = \{\}$›
**and** *indep-phils*: ‹$i \neq 0 \Longrightarrow \varepsilon\ A_{lp}\ \sigma \cap \varepsilon\ (A_{rp}\ i)\ \sigma' = \{\}$›
‹$i \neq j \Longrightarrow \varepsilon\ (A_{rp}\ i)\ \sigma \cap \varepsilon\ (A_{rp}\ j)\ \sigma' = \{\}$›
⟨*proof*⟩

Equalities between *FORK*, *RPHIL*, *LPHIL0* and respectively *fork-P-d*, *rphil-P-d*, *lphil0-P-d*

**lemma** *FORK-is-fork-P-d*: ‹*FORK i* = *fork-P-d i 0*›
⟨*proof*⟩

**lemma** *RPHIL-is-rphil-P-d*: ‹*RPHIL i = rphil-P-d i 0*›
⟨*proof*⟩

**lemma** *LPHIL0-is-lphil0-P-d*: ‹*LPHIL0 = lphil0-P-d 0*›
⟨*proof*⟩

### 11.3.2 *FORKS* is normalizable

Definition of the all-forks automaton

**type-synonym** $\sigma_{forks}$ = ‹*nat list*›

**definition** *forks-A* :: ‹($\sigma_{forks}$, *dining-event*, *unit*) $A_d$› (‹$A_F$›) **where** ‹$A_F \equiv \langle\!\langle_d\bigotimes [\![\{\}]\!]$ *map* $A_f$ [*0*..<*N*]⟫›

Definition and first properties of the associated normal process

**definition** *forks-P-d*:: ‹$\sigma_{forks}$ ⇒ *dining-event process*› **where** ‹*forks-P-d* ≡ $P\langle\!\langle A_F\rangle\!\rangle_d$›

**lemma** *forks-ε*: ‹*length fs = N* ⟹ $\varepsilon$ $A_F$ *fs* = (⋃*i*<*N*. $\varepsilon$ ($A_f$ *i*) (*fs* ! *i*))›
⟨*proof*⟩

Equality between *FORKS* and *forks-P-d*

**lemma** *NFORKS-is-forks-P-d*: ‹*FORKS = forks-P-d* (*replicate N 0*)›
⟨*proof*⟩

### 11.3.3 *PHILS* is normalizable

Definition of the all-philosophers automaton

**type-synonym** $\sigma_{phils}$ = ‹*nat list*›

**definition** *phils-A* :: ‹($\sigma_{phils}$, *dining-event*, *unit*) $A_d$› (‹$A_P$›) **where** ‹$A_P \equiv \langle\!\langle_d\bigotimes [\![\{\}]\!]$ $A_{lp}$ # *map* $A_{rp}$ [*1*..< *N*]⟫›

**lemma** *phils-A-def-bis*: ‹$A_P = \langle\!\langle_d\bigotimes [\![\{\}]\!]$ *map* (λ*i*. *if i = 0 then* $A_{lp}$ *else* $A_{rp}$ *i*) [*0*..<*N*]⟫›
⟨*proof*⟩

Definition and first properties of the associated normal process

**definition** *phils-P-d*:: ‹$\sigma_{phils}$ ⇒ *dining-event process*› **where** ‹*phils-P-d* ≡ $P\langle\!\langle A_P\rangle\!\rangle_d$›

**lemma** *phils-ε*: ‹*length ps = N* ⟹ $\varepsilon$ $A_P$ *ps* = $\varepsilon$ $A_{lp}$ (*ps* ! *0*) ∪ (⋃*i*∈{*1*..<*N*}. $\varepsilon$ ($A_{rp}$ *i*) (*ps* ! *i*))›

⟨*proof*⟩

Equality between *PHILS* and *phils-P-d*

**lemma** *NPHILS-is-phils-P-d*: ‹$PHILS = phils\text{-}P\text{-}d\ (replicate\ N\ 0)$›
⟨*proof*⟩

### 11.3.4 The complete process *DINING* is normalizable

Definition of the dining automaton

**definition** *dining-A* :: ‹$(\sigma_{phils} \times \sigma_{forks},\ dining\text{-}event,\ unit)\ A_d$› (‹$A_D$›) **where** ‹$A_D \equiv \langle\!\langle A_P\ {}_d\otimes [\![UNIV]\!]_{Pair}\ A_F \rangle\!\rangle$›

Definition and first properties of the associated normal process

**definition** *dining-P-d*:: ‹$\sigma_{phils} \times \sigma_{forks} \Rightarrow dining\text{-}event\ process$› **where** ‹$dining\text{-}P\text{-}d \equiv P\langle\!\langle A_D \rangle\!\rangle_d$›

**lemma** *dining-ε*:
‹$length\ ps = N \Longrightarrow length\ fs = N \Longrightarrow$
$\varepsilon\ A_D\ (ps, fs) = (\varepsilon\ A_{lp}\ (ps\ !\ 0) \cup (\bigcup i{\in}\{1..{<}N\}.\ \varepsilon\ (A_{rp}\ i)\ (ps\ !\ i))) \cap (\bigcup i{<}N.\ \varepsilon\ (A_f\ i)\ (fs\ !\ i))$›
⟨*proof*⟩

Equality between *DINING* and *dining-P-d*

**lemma** *DINING-is-dining-P-d*: ‹$DINING = dining\text{-}P\text{-}d\ (replicate\ N\ 0,\ replicate\ N\ 0)$›
⟨*proof*⟩

## 11.4 And finally: Philosophers may dine ! Always !

**method** *ε-sets-simp* **uses** *opt* = (*simp-all split*: *if-split-asm*)?,
*simp-all add*: *fork-ε lphil0-ε rphil-ε opt split*: *if-splits*
**method** *A-defs-simp* **uses** *opt* = (*simp-all split*: *if-split-asm*)?,
*simp-all add*: *fork-A-def lphil0-A-def rphil-A-def opt split*: *if-splits*

### 11.4.1 Construction of an invariant for the dining automaton

**definition** ‹$inv\text{-}dining\ ps\ fs \equiv$
$length\ fs = N \wedge length\ ps = N$
$\wedge\ (\forall i < N.\ fs\ !\ i = 0 \vee fs\ !\ i = 1 \vee fs\ !\ i = 2)$
$\wedge\ (\forall i < N.\ ps\ !\ i = 0 \vee ps\ !\ i = 1 \vee ps\ !\ i = 2 \vee ps\ !\ i = 3)$
$\wedge\ (\forall i.\ Suc\ i < N \longrightarrow ((fs\ !\ Suc\ i = 1) \longleftrightarrow ps\ !\ Suc\ i \neq 0)) \wedge (fs\ !\ (N - 1) = 2 \longleftrightarrow ps\ !\ 0 \neq 0)$
$\wedge\ (\forall i < N - 1.\ \ fs\ !\ i = 2 \longleftrightarrow ps\ !\ Suc\ i = 2) \wedge (fs\ !\ 0 = 1 \longleftrightarrow ps\ !\ 0 = 2)$›

**lemma** *show-inv-dining*:
‹*length fs* = *N* ∧ *length ps* = *N* ⟹
(∀ *i* < *N*. *fs* ! *i* = *0* ∨ *fs* ! *i* = *1* ∨ *fs* ! *i* = *2*) ⟹
(∀ *i* < *N*. *ps* ! *i* = *0* ∨ *ps* ! *i* = *1* ∨ *ps* ! *i* = *2* ∨ *ps* ! *i* = *3*) ⟹
(∀ *i*. *Suc i* < *N* ⟶ (*fs* ! *Suc i* = *1* ⟷ *ps* ! *Suc i* ≠ *0*)) ⟹ (*fs* ! (*N* − *1*) = *2* ⟷ *ps* ! *0* ≠ *0*) ⟹
(∀ *i* < *N* − *1*. *fs* ! *i* = *2* ⟷ *ps* ! *Suc i* = *2*) ⟹ (*fs* ! *0* = *1* ⟷ *ps* ! *0* = *2*) ⟹
*inv-dining ps fs*›
⟨*proof*⟩

**lemma** *inv-DINING*: ‹*s* ∈ $\mathcal{R}_d$ $A_D$ (*replicate N 0*, *replicate N 0*) ⟹ *inv-dining* (*fst s*) (*snd s*)›
⟨*proof*⟩

### 11.4.2 The invariant *inv-dining* implies that *DINING* is *deadlock-free*

**method** *nonempty-Int-by-common-element* **for** *x* = *rule-tac ex-in-conv*[*THEN iffD1*, *OF exI*, *OF IntI*, *of x*]

**lemma** *inv-implies-DF*: ‹ε $A_D$ (*ps*, *fs*) ≠ {}› **if** *hyp-inv*: ‹*inv-dining ps fs*›
⟨*proof*⟩

### 11.4.3 Conclusion

**corollary** *deadlock-free-DINING*: ‹*deadlock-free DINING*›
⟨*proof*⟩

## 11.5 Alternative version with only right-handed philosophers (in order to show that it's not *deadlock-free*)

### 11.5.1 Setup

**definition** ‹*RPHILS* ≡ ||| *P* ∈# *mset* (*map RPHIL* [*0*..< *N*]). *P*›

**corollary** ‹*N* = *3* ⟹ *RPHILS* = (*RPHIL 0* ||| *RPHIL 1* ||| *RPHIL 2*)›
⟨*proof*⟩

**definition** *RDINING* :: ‹*dining-event process*›
**where** ‹*RDINING* = (*FORKS* || *RPHILS*)›

### 11.5.2 Normalization

**definition** *rphils-A* :: ‹$(\sigma_{phils}$, *dining-event*, *unit*) $A_d$› (‹$A_{RP}$›) **where** ‹$A_{RP} \equiv \langle\!\langle_d \bigotimes [\![\{\}]\!]$ *map* $A_{rp}$ $[0..< N]\rangle\!\rangle$›

**definition** *rphils-P-d*:: ‹$\sigma_{phils} \Rightarrow$ *dining-event process*› **where** ‹*rphils-P-d* $\equiv P\langle\!\langle A_{RP}\rangle\!\rangle_d$›

**definition** *rdining-A* :: ‹$(\sigma_{phils} \times \sigma_{forks}$, *dining-event*, *unit*) $A_d$› (‹$A_{RD}$›) **where** ‹$A_{RD} \equiv \langle\!\langle A_{RP}\ {}_d\otimes[\![UNIV]\!]_{Pair}\ A_F\rangle\!\rangle$›

**definition** *rdining-P-d*:: ‹$\sigma_{phils} \times \sigma_{forks} \Rightarrow$ *dining-event process*› **where** ‹*rdining-P-d* $\equiv P\langle\!\langle A_{RD}\rangle\!\rangle_d$›

### 11.5.3 Correspondance between our normalized processes and the previous definitions

**lemma** *rphils-ε*: ‹*length ps* $= N \Longrightarrow \varepsilon\ A_{RP}\ ps = (\bigcup i{\in}\{0..{<}N\}.\ \varepsilon\ (A_{rp}\ i)\ (ps\ !\ i))$›
⟨*proof*⟩

**lemma** *NRPHILS-is-rphils-P-d*: ‹*RPHILS* = *rphils-P-d* (*replicate N 0*)›
⟨*proof*⟩

**lemma** *rdining-ε*:
‹*length ps* $= N \Longrightarrow$ *length fs* $= N \Longrightarrow$
$\varepsilon\ A_{RD}\ (ps, fs) = (\bigcup i{\in}\{0..{<}N\}.\ \varepsilon\ (A_{rp}\ i)\ (ps\ !\ i)) \cap (\bigcup i{<}N.\ \varepsilon\ (A_f\ i)\ (fs\ !\ i))$›
⟨*proof*⟩

**lemma** *RDINING-is-rdining-P-d*: ‹*RDINING* = *rdining-P-d* (*replicate N 0*, *replicate N 0*)›
⟨*proof*⟩

### 11.5.4 Proof that we have a deadlock in the state (*replicate N 1*, *replicate N 1*)

**lemma** *empty-enabl-replicate1*: ‹$\varepsilon\ A_{RD}$ (*replicate N 1*, *replicate N 1*) = {}›
⟨*proof*⟩

**corollary** *non-dealock-free-rdining*: ‹¬ *deadlock-free* (*rdining-P-d* (*replicate N 1*, *replicate N 1*))›
⟨*proof*⟩

### 11.5.5 Proof that this state is reachable from our initial state, i.e. (*replicate N 1*, *replicate N 1*) $\in \mathcal{R}_d\ A_{RD}$ (*replicate N 0*, *replicate N 0*)

**lemma** *rdining-τ*: ‹*length ps* $= N \Longrightarrow$ *length fs* $= N \Longrightarrow e \in \varepsilon\ A_{RD}\ (ps, fs) \Longrightarrow$
$\tau\ A_{RD}\ (ps, fs)\ e = \lfloor(\lceil\tau\ A_{RP}\ ps\ e\rceil,\ \lceil\tau\ A_F\ fs\ e\rceil)\rfloor$›
⟨*proof*⟩

**lemma** *replicate1-reachable-from-replicate0-prelim*:
‹$n \leq N \Longrightarrow$ (*replicate n 1* @ *replicate* ($N - n$) *0*, *replicate n 1* @ *replicate* ($N - n$) *0*) $\in \mathcal{R}_d$ $A_{RD}$ (*replicate N 0*, *replicate N 0*)›
⟨*proof*⟩

**corollary** *replicate1-reachable-from-replicate0*: ‹(*replicate N 1*, *replicate N 1*) $\in \mathcal{R}_d$ $A_{RD}$ (*replicate N 0*, *replicate N 0*)›
⟨*proof*⟩

**theorem** *not-deadlock-free-RDINING*: ‹$\neg$ *deadlock-free RDINING*›
⟨*proof*⟩

**end**

# Chapter 12

# Other Results similar to Compactification

Unlike *Sync* and (**;**), some operators like *Det* do not enjoy a compactification result. Nevertheless, we still can prove some useful lemmas.

## 12.1 Some preliminary Results

**lemma** *Mprefix-Det-Mprefix-bis* :
‹$(\Box a \in A \rightarrow P\ a)\ \Box\ (\Box b \in B \rightarrow Q\ b) =$
$(\Box x \in (A \cap B) \rightarrow P\ x \sqcap Q\ x)\ \Box\ (\Box a \in (A - B) \rightarrow P\ a)\ \Box\ (\Box b \in (B - A) \rightarrow Q\ b)$›
(**is** ‹*?lhs* = *?rhs*›)
⟨*proof*⟩

**lemma** *GlobalNdet-Ndet-GlobalNdet*:
‹$A \neq \{\} \Longrightarrow B \neq \{\} \Longrightarrow (\sqcap a \in A.\ P\ a) \sqcap (\sqcap b \in B.\ Q\ b) =$
$\sqcap x \in (A \cup B).\ (\mathit{if}\ x \in A \cap B\ \mathit{then}\ P\ x \sqcap Q\ x\ \mathit{else}\ \mathit{if}\ x \in A\ \mathit{then}\ P\ x\ \mathit{else}\ Q\ x)$›
⟨*proof*⟩

**lemma** *GlobalNdet-Ndet-GlobalNdet-bis*:
‹$A \cap B \neq \{\} \Longrightarrow A - B \neq \{\} \Longrightarrow B - A \neq \{\} \Longrightarrow$
$(\sqcap a \in A.\ P\ a) \sqcap (\sqcap b \in B.\ Q\ b) =$
$(\sqcap x \in (A \cap B).\ P\ x \sqcap Q\ x) \sqcap (\sqcap a \in (A - B).\ P\ a) \sqcap (\sqcap b \in (B - A).\ Q\ b)$›
⟨*proof*⟩

**lemma** *GlobalNdet-GlobalNdet*:
‹$(\sqcap a \in A.\ \sqcap b \in B\ a.\ P\ b) =$
$(\mathit{if}\ \forall a \in A.\ B\ a \neq \{\}\ \mathit{then}\ \sqcap b \in (\bigcup a \in A.\ B\ a).\ P\ b\ \mathit{else}\ (\sqcap b \in (\bigcup a \in A.\ B\ a).\ P\ b) \sqcap \mathit{STOP})$›
⟨*proof*⟩

## 12.2 Results for *Det*

**lemma** *P-nd-set-almost-compactification-Det* :
‹$(\Box\ (s,\ A) \in$ *s-A-set*. $P\langle\!\langle A\rangle\!\rangle_{nd}\ s) =$
$\Box e \in (\bigcup (s,\ A) \in$ *s-A-set*. $\varepsilon\ A\ s) \rightarrow$
$\sqcap (s,\ A) \in \{(s,\ A) \in$ *s-A-set*. $e \in \varepsilon\ A\ s\}$.
$\sqcap s' \in \tau\ A\ s\ e.\ P\langle\!\langle A\rangle\!\rangle_{nd}\ s'$› (**is** ‹*?lhs* = *?rhs*›)
⟨*proof*⟩

**lemma** *P-nd-set-almost-compactification-Det-bis* :
‹$(\Box\ (s,\ A) \in$ *s-A-set*. $P\langle\!\langle A\rangle\!\rangle_{nd}\ s) =$
$\Box e \in (\bigcup (s,\ A) \in$ *s-A-set*. $\varepsilon\ A\ s) \rightarrow$
$\sqcap (s',\ A) \in \{(s',\ A) |\ s'\ s\ A.\ (s,\ A) \in$ *s-A-set* $\wedge\ e \in \varepsilon\ A\ s \wedge s' \in \tau\ A\ s\ e\}.\ P\langle\!\langle A\rangle\!\rangle_{nd}$ $s'$›
(**is** ‹- = *?rhs*›)
⟨*proof*⟩

**lemma** *P-d-set-almost-compactification-Det*:
**shows** ‹$(\Box\ (s,\ A) \in$ *s-A-set*. $P\langle\!\langle A\rangle\!\rangle_{d}\ s) =$
$\Box e \in (\bigcup (s,\ A) \in$ *s-A-set*. $\varepsilon\ A\ s) \rightarrow$
$\sqcap (s,\ A) \in \{(s,\ A) \in$ *s-A-set*. $e \in \varepsilon\ A\ s\}.\ P\langle\!\langle A\rangle\!\rangle_{d}\ \lceil \tau\ A\ s\ e\rceil$› (**is** ‹*?lhs* = *?rhs*›)
⟨*proof*⟩

**lemma** *P-d-set-almost-compactification-Det-bis*:
**shows** ‹$(\Box\ (s,\ A) \in$ *s-A-set*. $P\langle\!\langle A\rangle\!\rangle_{d}\ s) =$
$\Box e \in (\bigcup (s,\ A) \in$ *s-A-set*. $\varepsilon\ A\ s) \rightarrow$
$\sqcap (s',\ A) \in \{(\lceil \tau\ A\ s\ e\rceil,\ A) |\ s\ A.\ (s,\ A) \in$ *s-A-set* $\wedge\ e \in \varepsilon\ A\ s\}.\ P\langle\!\langle A\rangle\!\rangle_{d}\ s'$›
⟨*proof*⟩

## 12.3 Results for *Ndet*

## 12.4 Other Operators

### 12.4.1 *initials*

**lemma** *initials-*$P_{SKIPS}$*-nd* :
‹$(P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma)^0 = (\mathit{if}\ \sigma \in \varrho\ A\ \mathit{then}\ \mathit{tick}\ `\ \omega\ A\ \sigma\ \mathit{else}\ \mathit{ev}\ `\ \varepsilon\ A\ \sigma)$›
⟨*proof*⟩

**lemma** *initials-*$P_{SKIPS}$*-d* :

‹$(P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma)^0 = (\mathit{if}\ \sigma \in \varrho\ A\ \mathit{then}\ \{\checkmark(\lceil\omega\ A\ \sigma\rceil)\}\ \mathit{else}\ \mathit{ev}\ `\ \varepsilon\ A\ \sigma)$›
⟨*proof*⟩

**lemma** *initials-P-nd* : ‹$(P\langle\!\langle A\rangle\!\rangle_{nd}\ s)^0 = \mathit{ev}\ `\ \varepsilon\ A\ s$›
⟨*proof*⟩

**lemma** *initials-P-d* : ‹$(P\langle\!\langle A\rangle\!\rangle_d\ s)^0 = \mathit{ev}\ `\ \varepsilon\ A\ s$›
⟨*proof*⟩

### 12.4.2 *Throw*

**lemma** *Throw-$P_{SKIPS}$-nd* :
‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma\ \Theta\ b \in B.\ Q\ b =$
$(\mathit{if}\ \sigma \in \varrho\ A\ \mathit{then}\ \mathit{SKIPS}\ (\omega\ A\ \sigma)\ \mathit{else}$
$\Box a{\in}\varepsilon\ A\ \sigma \rightarrow (\mathit{if}\ a \in B\ \mathit{then}\ Q\ a\ \mathit{else}\ \sqcap\sigma' \in \tau\ A\ \sigma\ a.\ (P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma'\ \Theta\ b \in B.\ Q\ b)))$›
⟨*proof*⟩

**lemma** *Throw-$P_{SKIPS}$-d* :
‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma\ \Theta\ b \in B.\ Q\ b =$
$(\mathit{if}\ \sigma \in \varrho\ A\ \mathit{then}\ \mathit{SKIP}\ \lceil\omega\ A\ \sigma\rceil\ \mathit{else}$
$\Box a{\in}\varepsilon\ A\ \sigma \rightarrow (\mathit{if}\ a \in B\ \mathit{then}\ Q\ a\ \mathit{else}\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \lceil\tau\ A\ \sigma\ a\rceil\ \Theta\ b \in B.\ Q\ b))$›
⟨*proof*⟩

**lemma** *Throw-P-nd* :
‹$P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma\ \Theta\ b \in B.\ Q\ b =$
$\Box a{\in}\varepsilon\ A\ \sigma \rightarrow (\mathit{if}\ a \in B\ \mathit{then}\ Q\ a\ \mathit{else}\ \sqcap\sigma' \in \tau\ A\ \sigma\ a.\ (P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma'\ \Theta\ b \in B.\ Q\ b))$›
⟨*proof*⟩

**lemma** *Throw-P-d* :
‹$P\langle\!\langle A\rangle\!\rangle_d\ \sigma\ \Theta\ b \in B.\ Q\ b =$
$\Box a{\in}\varepsilon\ A\ \sigma \rightarrow (\mathit{if}\ a \in B\ \mathit{then}\ Q\ a\ \mathit{else}\ P\langle\!\langle A\rangle\!\rangle_d\ \lceil\tau\ A\ \sigma\ a\rceil\ \Theta\ b \in B.\ Q\ b)$›
⟨*proof*⟩

### 12.4.3 $(\triangle)$

**lemma** *SKIPS-Interrupt-is-SKIPS-Det* :
‹$\mathit{SKIPS}\ R\ \triangle\ P = \mathit{SKIPS}\ R\ \Box\ P$›
⟨*proof*⟩

**lemma** *Interrupt-$P_{SKIPS}$-nd* :
‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma\ \triangle\ Q =$
$Q\ \Box\ (\mathit{if}\ \sigma \in \varrho\ A\ \mathit{then}\ \mathit{SKIPS}\ (\omega\ A\ \sigma)\ \mathit{else}\ \Box a \in \varepsilon\ A\ \sigma \rightarrow \sqcap\sigma' \in \tau\ A\ \sigma\ a.\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma'\ \triangle\ Q)$›
⟨*proof*⟩

**lemma** *Interrupt-$P_{SKIPS}$-d* :
‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma\ \triangle\ Q =$

$Q \,\Box\, (\mathit{if}\ \sigma \in \varrho\ A\ \mathit{then}\ \mathit{SKIP}\ \lceil\omega\ A\ \sigma\rceil\ \mathit{else}\ \Box a \in \varepsilon\ A\ \sigma \rightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \lceil\tau\ A\ \sigma\ a\rceil \triangle Q)$›
⟨*proof*⟩

**lemma** *Interrupt-P-nd* :
‹$P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma \triangle Q = Q \,\Box\, (\Box a \in \varepsilon\ A\ \sigma \rightarrow \sqcap\sigma' \in \tau\ A\ \sigma\ a.\ P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma' \triangle Q)$›
⟨*proof*⟩

**lemma** *Interrupt-P-d* :
‹$P\langle\!\langle A\rangle\!\rangle_d\ \sigma \triangle Q = Q \,\Box\, (\Box a \in \varepsilon\ A\ \sigma \rightarrow P\langle\!\langle A\rangle\!\rangle_d\ \lceil\tau\ A\ \sigma\ a\rceil \triangle Q)$›
⟨*proof*⟩

### 12.4.4 After

**context** *After*
**begin**

**lemma** *After-SKIPS* : ‹$\mathit{SKIPS}\ R\ \mathit{after}\ a = \Psi\ (\mathit{SKIPS}\ R)\ a$›
⟨*proof*⟩

**lemma** *After-$P_{SKIPS}$-nd* :
‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma\ \mathit{after}\ a =$
$(\mathit{if}\ \sigma \in \varrho\ A\ \mathit{then}\ \Psi\ (\mathit{SKIPS}\ (\omega\ A\ \sigma))\ a\ \mathit{else}$
$\mathit{if}\ a \in \varepsilon\ A\ \sigma\ \mathit{then}\ \sqcap\sigma' \in \tau\ A\ \sigma\ a.\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma'\ \mathit{else}\ \Psi\ (P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma)\ a)$›
⟨*proof*⟩

**lemma** *After-$P_{SKIPS}$-d* :
‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma\ \mathit{after}\ a =$
$(\mathit{if}\ \sigma \in \varrho\ A\ \mathit{then}\ \Psi\ (\mathit{SKIP}\ \lceil\omega\ A\ \sigma\rceil)\ a\ \mathit{else}$
$\mathit{if}\ a \in \varepsilon\ A\ \sigma\ \mathit{then}\ P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \lceil\tau\ A\ \sigma\ a\rceil\ \mathit{else}\ \Psi\ (P_{SKIPS}\langle\!\langle A\rangle\!\rangle_d\ \sigma)\ a)$›
⟨*proof*⟩

**lemma** *After-P-nd* :
‹$P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma\ \mathit{after}\ a = (\mathit{if}\ a \in \varepsilon\ A\ \sigma\ \mathit{then}\ \sqcap\sigma' \in \tau\ A\ \sigma\ a.\ P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma'\ \mathit{else}\ \Psi\ (P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma)\ a)$›
⟨*proof*⟩

**lemma** *After-P-d* :
‹$P\langle\!\langle A\rangle\!\rangle_d\ \sigma\ \mathit{after}\ a = (\mathit{if}\ a \in \varepsilon\ A\ \sigma\ \mathit{then}\ P\langle\!\langle A\rangle\!\rangle_d\ \lceil\tau\ A\ \sigma\ a\rceil\ \mathit{else}\ \Psi\ (P\langle\!\langle A\rangle\!\rangle_d\ \sigma)\ a)$›
⟨*proof*⟩

**end**

**context** *AfterExt*
**begin**

**lemma** *After$_{tick}$-SKIPS* :

‹*SKIPS R after*$_{\checkmark}$ *e* = (*case e of ev a* ⇒ Ψ (*SKIPS R*) *a* | ✓(*r*) ⇒ Ω (*SKIPS R*) *r*)›
⟨*proof*⟩

**lemma** *After*$_{tick}$-$P_{SKIPS}$-*nd* :
‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}$ σ *after*$_{\checkmark}$ *e* =
(*case e of ev a* ⇒ *if* σ ∈ ϱ *A then* Ψ (*SKIPS* (ω *A* σ)) *a else*
*if a* ∈ ε *A* σ *then* ⊓σ′ ∈ τ *A* σ *a*. $P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}$ σ′ *else* Ψ ($P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}$ σ) *a*
| ✓(*r*) ⇒ *if* σ ∈ ϱ *A then* Ω (*SKIPS* (ω *A* σ)) *r else* Ω ($P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}$ σ) *r*)›
⟨*proof*⟩

**lemma** *After*$_{tick}$-$P_{SKIPS}$-*d* :
‹$P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{d}$ σ *after*$_{\checkmark}$ *e* =
(*case e of ev a* ⇒ *if* σ ∈ ϱ *A then* Ψ (*SKIP* ⌈ω *A* σ⌉) *a else*
*if a* ∈ ε *A* σ *then* $P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{d}$ ⌈τ *A* σ *a*⌉ *else* Ψ ($P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{d}$ σ) *a*
| ✓(*r*) ⇒ *if* σ ∈ ϱ *A then* Ω (*SKIP* ⌈ω *A* σ⌉) *r else* Ω ($P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{d}$ σ) *r*)›
⟨*proof*⟩

**lemma** *After*$_{tick}$-*P*-*nd* :
‹$P\langle\!\langle A\rangle\!\rangle_{nd}$ σ *after*$_{\checkmark}$ *e* =
(*case e of ev a* ⇒ *if a* ∈ ε *A* σ *then* ⊓σ′ ∈ τ *A* σ *a*. $P\langle\!\langle A\rangle\!\rangle_{nd}$ σ′ *else* Ψ ($P\langle\!\langle A\rangle\!\rangle_{nd}$ σ) *a*
| ✓(*r*) ⇒ Ω ($P\langle\!\langle A\rangle\!\rangle_{nd}$ σ) *r*)›
⟨*proof*⟩

**lemma** *After*$_{tick}$-*P*-*d* :
‹$P\langle\!\langle A\rangle\!\rangle_{d}$ σ *after*$_{\checkmark}$ *e* =
(*case e of ev a* ⇒ *if a* ∈ ε *A* σ *then* $P\langle\!\langle A\rangle\!\rangle_{d}$ ⌈τ *A* σ *a*⌉ *else* Ψ ($P\langle\!\langle A\rangle\!\rangle_{d}$ σ) *a*
| ✓(*r*) ⇒ Ω ($P\langle\!\langle A\rangle\!\rangle_{d}$ σ) *r*)›
⟨*proof*⟩

**end**

## 12.5 OpSem

**context** *OpSemTransitions*
**begin**

**lemma** *SKIPS*-τ-*trans*-*SKIP* : ‹*r* ∈ *R* ⟹ *SKIPS R* $\rightsquigarrow_{\tau}$ *SKIP r*›
⟨*proof*⟩

In the ProcOmata, we will absorb the τ transitions that appear when we unfold the fixed-point operator.

**lemma** *$\tau$-trans-$P_{SKIPS}$-nd* :
‹$r \in \omega\ A\ \sigma \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma \rightsquigarrow_{\tau} SKIP\ r$›
⟨*proof*⟩

**lemma** *$\tau$-trans-$P_{SKIPS}$-d* :
‹$\sigma \in \varrho\ A \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{d}\ \sigma \rightsquigarrow\checkmark\lceil\omega\ A\ \sigma\rceil\ \Omega\ (SKIP\ \lceil\omega\ A\ \sigma\rceil)\ \lceil\omega\ A\ \sigma\rceil$›
⟨*proof*⟩

**lemma** *ev-trans-$P_{SKIPS}$-nd* :
‹$\sigma \notin \varrho\ A \Longrightarrow \sigma' \in \tau\ A\ \sigma\ a \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma \rightsquigarrow_{a} P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma'$›
⟨*proof*⟩

**lemma** *ev-trans-$P_{SKIPS}$-d* :
‹$\sigma \notin \varrho\ A \Longrightarrow a \in \varepsilon\ A\ \sigma \Longrightarrow P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{d}\ \sigma \rightsquigarrow_{a} P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{d}\ \lceil\tau\ A\ \sigma\ a\rceil$›
⟨*proof*⟩

**lemma** *ev-trans-P-nd* :
‹$\sigma' \in \tau\ A\ \sigma\ a \Longrightarrow P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma \rightsquigarrow_{a} P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma'$›
⟨*proof*⟩

**lemma** *ev-trans-P-d* :
‹$a \in \varepsilon\ A\ \sigma \Longrightarrow P\langle\!\langle A\rangle\!\rangle_{d}\ \sigma \rightsquigarrow_{a} P\langle\!\langle A\rangle\!\rangle_{d}\ \lceil\tau\ A\ \sigma\ a\rceil$›
⟨*proof*⟩

**end**

# Chapter 13

# Conclusion

## 13.1 Entry Point

This is where `HOL-CSP_Proc-Omata` should be imported from.

## 13.2 Conclusion

In this entry we have developed the Proc-Omata framework on top of `HOL-CSP` and its extensions. Starting from functional automata, we introduced Proc-Omata in four variants: deterministic, terminating deterministic, non-deterministic, and terminating non-deterministic. They enjoy strong structural properties, for example deadlocks can be characterized directly and established by invariant reasoning:

$$\frac{\textit{deadlock-free}\ (P\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma) = (\forall\, \sigma' \in \mathcal{R}_{nd}\ A\ \sigma.\ \varepsilon\ A\ \sigma' \neq \emptyset) \qquad \varrho\text{-}\textit{disjoint-}\varepsilon\ A}{\textit{deadlock-free}_{SKIPS}\ (P_{SKIPS}\langle\!\langle A\rangle\!\rangle_{nd}\ \sigma) = (\forall\, \sigma' \in \mathcal{R}_{nd}\ A\ \sigma.\ \sigma' \in \varrho\ A \vee \varepsilon\ A\ \sigma' \neq \emptyset)}$$

We then lifted sequential composition and synchronization product to the automata level, by defining suitable combination functions and proving their correctness. A major generalization of our development is the treatment of parameterized termination. For sequential composition we worked directly with the generalized operator ($;_{\checkmark}$), since the standard one ($;$) is easily recovered (indeed $P\ ;_{\checkmark}\ (\lambda r.\ Q) = P\ ;\ Q$). In contrast, for synchronization product we had to provide two distinct versions, as the handling of ticks prevents any straightforward reduction from $P\ [\![A]\!]\ Q$ to $P\ [\![A]\!]_{\checkmark}\ Q$.

Another central ingredient is the library `Restriction_Spaces` [1]. Proc-Omata are indeed defined as fixed points of endofunctions which, in the non-deterministic case, are not always continuous due to global non-deterministic

choice. While deterministic prefix choice does not suffice to restore continuity under composition, it does guarantee constructiveness, allowing us to rely on the fixed-point operator $\upsilon\ x.\ f\ x$ in all cases.

The resulting framework yields compactification theorems that support invariant-based reasoning over large process architectures:

$$\frac{|\sigma s| = |As|}{[\![E]\!]\ (\sigma,\ A) \in \#mset\ (zip\ \sigma s\ As).\ P\langle\!\langle A \rangle\!\rangle_{nd}\ \sigma = P\langle\!\langle\!\langle_{nd} \bigotimes [\![E]\!]\ As \rangle\!\rangle\!\rangle_{nd}\ \sigma s}$$

Finally, we demonstrated the applicability of our approach with the Dining Philosophers case study, where Proc-Omata compactification enables proofs that scale to an arbitrary number of participants in this parameterized process architecture.

# Bibliography


[1] B. Ballenghien, B. Puyobro, and B. Wolff. Restriction spaces: a fixed-point theory. *Archive of Formal Proofs*, May 2025. https://isa-afp.org/entries/Restriction_Spaces.html, Formal proof development.

[2] B. Ballenghien, S. Taha, and B. Wolff. Hol-cspm - architectural operators for hol-csp. *Archive of Formal Proofs*, December 2023. https://isa-afp.org/entries/HOL-CSPM.html, Formal proof development.

[3] B. Ballenghien, S. Taha, B. Wolff, and L. Ye. Hol-csp version 2.0. *Archive of Formal Proofs*, April 2019. https://isa-afp.org/entries/HOL-CSP.html, Formal proof development.

[4] B. Ballenghien and B. Wolff. Operational semantics formally proven in hol-csp. *Archive of Formal Proofs*, December 2023. https://isa-afp.org/entries/HOL-CSP_OpSem.html, Formal proof development.

[5] B. Ballenghien and B. Wolff. A theory of proc-omataand proof methods for process architectures. In *Theoretical Aspects of Computing ICTAC 2024: 21st International Colloquium, Bangkok, Thailand, November 2529, 2024, Proceedings*, page 272289, Berlin, Heidelberg, 2024. Springer-Verlag.

[6] B. Ballenghien and B. Wolff. Csp semantics over restriction spaces. *Archive of Formal Proofs*, May 2025. https://isa-afp.org/entries/HOL-CSP_RS.html, Formal proof development.

[7] A. Roscoe. *Theory and Practice of Concurrency.* Prentice Hall, 1997.

[8] S. Taha, B. Wolff, and L. Ye. Philosophers may dine - definitively! In *Integrated Formal Methods: 16th International Conference, IFM 2020, Lugano, Switzerland, November 1620, 2020, Proceedings*, page 419439, Berlin, Heidelberg, 2020. Springer-Verlag.