Notions of Refinement for a Coordination Language for Gamma [*]

Michel Chaudron
Department of Informatics
Leiden University, The Netherlands
chaudron@cs.leidenuniv.nl

Revised: February 14, 1997

Abstract

Gamma has shown to be a powerful and expressive formalism that allows the basic computation of an algorithm to be expressed with a minimum of control. In a second stage of the design process, the highly nondeterministic behaviour of Gamma can be exploited to impose additional control using a separate coordination language.

Separating computation from coordination facilitates correctness proofs of the computation component, because operational details need not be taken into account. In this respect it is important that the coordination component does not invalidate the established correctness of the program.

In this paper we propose several notions of refinement that allow the coordination component of a Gamma program to be constructed through a correctness preserving derivation process. We show that the notions can be combined into a hybrid proof method that can be used to reason about different properties of the coordination component.

[*]A short version of this report appears as [6]

Contents

1 Introduction

From different areas of computer programming, the view has emerged that a program is made up from a computation and a coordination component. A computation component specifies the basic actions to be performed in order to solve a problem. A coordination component determines the way in which the computations are combined into a coherent ensemble [10].

In accordance to this view, we proposed in [8] a method for the design of parallel programs where computation and coordination are addressed separately. The main benefit of this approach, is that it leads to a uniform approach for parallel and distributed computing. This results in a greater ease of design, a high degree of portability (of the computation component) and supports collections of heterogeneous computing agents.

It is argued in [2] and [8] that the Gamma formalism is well suited for expressing the basic computations of an algorithm with a minimum of control. Hence Gamma programs are not biased towards any particular mode of execution. The abstraction of control-flow is an advantage is the design stage, because in proving the correctness of a program, one does not have to take (efficiency related) operational details into account. A method for the construction of correct Gamma programs, akin to the one advocated by [11], was described by Bânatre and Le Métayer in [1].

In a second phase of the design process, we can exploit the highly nondeterministic behaviour of Gamma programs to obtain efficiency. To this end, we proposed in [7], a language for coordinating the actions of Gamma programs. This language enables the programmer to determine operational details that are left unspecified by the Gamma program.

In this paper we focus on methods for the construction and verification of the coordination component of a Gamma program. We investigate several notions of refinement that allow the construction of a coordination component through a correctness preserving derivation process such that the correctness of the initial Gamma program is not invalidated.

The paper is organized as follows. In Section 2 we briefly describe the Gamma model and its coordination language. In Section 3 we examine various notions of refinement for schedules and illustrate their use in the derivation process. In Section 4 we illustrate how the notions of refinement can be combined into a method for the derivation of coordination strategies. We conclude with some final remarks and directions for future research in Section 5.

2 Gamma and its coordination language

We start with a brief introduction to the components that make up our programming model: the Gamma language, which is used to specify computations, and the coordination language which is used to control Gamma computations.

2.1 The Computation Language Gamma

The Gamma model [2] provides the multiset as the basic data structuring mechanism. Multisets can be formed over arbitrary domains of values, including integers, reals, booleans and tuples. The set of multisets is denoted $\mathbb{M}$. Gamma programs are built from conditional multiset rewrite rules, denoted $\overline{x} \mapsto m \Leftarrow b$. Here $\overline{x}$ denotes a sequence $x_1, \ldots, x_n$ of variables, m denotes a multiset expression, and b denotes a boolean expression, called the reaction condition. The free variables in m and b are taken from $x_1, \ldots, x_n$.

A rewrite rule may execute if there are values in the multiset that satisfy the rule's reaction condition. These values are then replaced by (transformed into) the elements that result from evaluating the multiset expression (for these values). Execution of a rule proceeds until there are no more matching elements in the multiset.

Example 2.0.1 *Consider the rewrite rule*

$$sieve \mathrel{\widehat{=}} x, y \mapsto x \Leftarrow y \, mod \, x = 0$$

Execution of this rule eliminates a number, y, that is a multiple of some other number, x, present in the multiset. When started with a multiset $M_0 = \{2, \ldots, n\}$, execution of the program sieve continues until it reaches a state that contains only the prime numbers up to n.

The Gamma program does not specify in which order numbers are deleted. Different numbers may be compared and removed in parallel, but this need not be the case. The program may behave as the well known algorithm of Eratosthenes as well as in some apparently chaotic way. $\square$

From individual rewrite-rules more complex programs can be built. Rules may be composed in parallel, forming so-called *simple* programs, using the "+" operator. The constituent rules of a simple program are executed in any order, possibly in parallel, until none of these rules can be applied. Simple programs can in turn be composed using the sequential operator, denoted " $\circ$ ". A program $P_2 \circ P_1$ first executes P_1 until completion, after which execution continues with P_2.

Formal definitions of Gamma's operational semantics are given in [12] and [7]. In [12] Hankin et al. use a single step transition system as opposed to [7] where a multistep transition system is used. We postpone a treatment of the formal semantics until Section 2.2.3, where we shall present an alternative definition in terms of our coordination language.

2.2 A Coordination Language for Gamma

Execution in Gamma proceeds in a highly nondeterministic fashion. Therefore, a Gamma program can be seen as the specification of a wide spectrum of (more deterministic) behaviours. The language, to be presented shortly, is designed as a kernel language for coordinating Gamma programs. It provides the means to specify execution plans according to which a Gamma program may be executed. Hence we usually refer to the terms from the coordination language as *schedules*.

2.2.1 Introduction to Schedules

The basic components of the coordination language are the multiset rewrite-rules from a Gamma program. The set $\mathbb{S}$ of schedules consists of all terms that can be built from these rules according to the following abstract syntax. Here s denotes a schedule, r a multiset rewrite-rule, c a boolean expression, S a schedule identifier, and $\overline{v}$ a sequence of values.

$$s \quad ::= \quad \mathsf{skip} \quad | \quad r \to s[s] \quad | \quad s; s \quad | \quad s \,\|\, s \quad | \quad c \rhd s[s] \quad | \quad !s \quad | \quad S(\overline{v})$$

skip denotes the empty schedule. The link between schedules and Gamma programs is made by the rule-conditional schedule $r \to s[t]$. This schedule first attempts to execute r. If this succeeds, then execution continues with s; otherwise, if r fails, execution continues with t. As a notational convention, we write $r \to s[\mathsf{skip}]$ as $r \to s$ and $r \to \mathsf{skip}$ as r. More complex schedules can be obtained using ";", which denotes sequential composition, and " $\|$ " which denotes parallel composition. The execution of $s \,\|\, t$ proceeds by a step performed by either s or t, or by a parallel step in which both s and t participate. We use $\Pi_{i=1}^{n} s_i$ to denote $s_1 \,\|\, \ldots \,\|\, s_n$ and write s^k, for $k \geq 0$, to denote k copies of s composed in parallel.

Execution of Gamma programs is such that the number of rules that may be executed in parallel varies dynamically with the number of elements available in the multiset. In order to describe this dynamic behaviour using schedules, the replication operator "!" is included. The schedule $!s$ denotes a dynamically varying number of copies of s executing in parallel.

Each occurrence of a schedule identifier, as in $S(\overline{v})$, has a corresponding schedule definition of the form $S(\overline{x}) \mathrel{\widehat{=}} s$ where the free variables in s are taken from $\overline{x}$. From a program design point of view, schedule definitions provide the means for specifying schedules in a modular fashion. From a theoretical point of view, it enables the construction of recursive schedules. Execution of $S(\overline{v})$ proceeds by execution of its body s, where the actual arguments $\overline{v}$ are substituted for the variables $\overline{x}$.

Recursive definitions are typically accompanied by the use of the conditional construct $c \rhd s[t]$, which serves as a guard to the recursion. Here c represents a boolean expression that is independent from the multiset. If c evaluates to *true*, then execution continues with s, otherwise t is executed. Analogously to the rule conditional, $c \rhd s[\mathsf{skip}]$ is written as $c \rhd s$.

Nondeterminism in Gamma arises at two levels:

1. at the selection of a rewrite-rule,

2. in selecting elements from the multiset.

The coordination language as introduced so far is only capable of resolving the first type of nondeterminism. The second type is resolved by *strengthening* the condition of a rewrite-rule. Consider a rule $r = \overline{x} \mapsto m \Leftarrow b$. Rather than scheduling r directly, we can schedule a rule $r' = \overline{x} \mapsto m \Leftarrow b'$, such that $b' \Rightarrow b$. Rule r' exhibits restricted behaviour compared to r, because there are fewer elements from the multiset that satisfy the stronger condition b'.

In general, a rewrite rule may not be replaced by an arbitrary strengthening of that rule, because this may invalidate computational correctness. In practice, strengthening is used to restrict the enabling condition of a rewrite rule such that it pin-points the elements to be selected for rewriting. The techniques for refinement, to be presented later on, will enable us to assert the correctness of the strengthenings used.

2.2.2 Operational Semantics of Schedules

Formally, execution of a rule $\overline{x} \mapsto m \Leftarrow b$ in a multiset M takes place if there are elements $\overline{v} \subseteq M$ such that the boolean expression with $\overline{v}$ substituted for $\overline{x}$, denoted $b[\overline{x} := \overline{v}]$, evaluates to *true*. Then the values $\overline{v}$ are removed, and the values from the multiset $M' = m[\overline{x} := \overline{v}]$ are inserted. The effect of a rewrite is formally represented by $M[\sigma]$ where σ denotes a multiset substitution $M'/\overline{v}$. More formally, given a multiset substitution $\sigma = N'/N$, then $M[\sigma] = (M \ominus N) \cup N'$. The operators $\ominus$ and $\cup$ denote multiset difference and union and are defined formally in Section 6. Two multiset substitutions σ_1 and σ_2 may be applied in the same multiset if they do not interfere. This is denoted $M \models \sigma_1 \bowtie \sigma_2$ and is defined in Section 6.

A configuration of a schedule s and a multiset M is written $\langle s, M\rangle$. The set $\mathbb{S} \times \mathbb{M}$ of configurations is denoted $\mathbb{C}$. The operational semantics of schedules is defined by the structural congruence relation in Figure 1 and a multistep transition relation $\xrightarrow{\lambda} \subseteq \mathbb{C} \times \mathbb{C}$ depicted in Figure 2. The label λ of a transition is either a multiset substitution or the special symbol ε which denotes a transition that does not change the multiset.

$$
\begin{array}{llcll}
(E1) & \mathsf{skip}; s & \equiv & s & \\
(E2) & s_1; (s_2; s_3) & \equiv & (s_1; s_2); s_3 & \\
(E3) & \mathsf{skip} \parallel s & \equiv & s & \\
(E4) & s_1 \parallel (s_2 \parallel s_3) & \equiv & (s_1 \parallel s_2) \parallel s_3 & \\
(E5) & s_1 \parallel s_2 & \equiv & s_2 \parallel s_1 & \\
(E6) & \mathit{true} \triangleright s[t] & \equiv & s & \\
(E7) & \mathit{false} \triangleright s[t] & \equiv & t & \\
(E8) & !\mathsf{skip} & \equiv & \mathsf{skip} & \\
(E9) & S(\overline{v}) & \equiv & \mathsf{skip} & \text{if } S(\overline{x}) \mathrel{\widehat{=}} s \text{ and } s[\overline{x} := \overline{v}] \equiv \mathsf{skip}
\end{array}
$$

Figure 1: Structural Congruences for Schedules

$$(N0) \qquad \langle r \to s[t], M\rangle \xrightarrow{\varepsilon} \langle t, M\rangle \qquad \text{if } \nexists \overline{v} \subseteq M : b[\overline{x} := \overline{v}]$$

$$(N1) \qquad \langle r \to s[t], M\rangle \xrightarrow{\sigma} \langle s, M[\sigma]\rangle \qquad \text{if } \overline{v} \subseteq M \wedge b[\overline{x} := \overline{v}] \text{ and } \sigma = m[\overline{x} := \overline{v}]/\overline{v}$$

$$(N2) \qquad \frac{\langle s_1, M\rangle \xrightarrow{\lambda} \langle s_1', M'\rangle}{\langle s_1 \parallel s_2, M\rangle \xrightarrow{\lambda} \langle s_1' \parallel s_2, M'\rangle}$$

$$(N3) \qquad \frac{\begin{array}{c}\langle s_1, M\rangle \xrightarrow{\lambda} \langle s_1', M'\rangle \\ \langle s_2, M\rangle \xrightarrow{\varepsilon} \langle s_2', M\rangle\end{array}}{\langle s_1 \parallel s_2, M\rangle \xrightarrow{\lambda} \langle s_1' \parallel s_2', M'\rangle}$$

$$(N4) \qquad \frac{\begin{array}{c}\langle s_1, M\rangle \xrightarrow{\sigma_1} \langle s_1', M_1\rangle \\ \langle s_2, M\rangle \xrightarrow{\sigma_2} \langle s_2', M_2\rangle\end{array}}{\langle s_1 \parallel s_2, M\rangle \xrightarrow{\sigma} \langle s_1' \parallel s_2', M[\sigma]\rangle} \qquad \begin{array}{ll}\text{where} & \sigma = \sigma_1 \cdot \sigma_2 \\ \text{and} & M \models \sigma_1 \bowtie \sigma_2\end{array}$$

$$(N5) \qquad \frac{\langle s_1, M\rangle \xrightarrow{\lambda} \langle s_1', M'\rangle}{\langle s_1; s_2, M\rangle \xrightarrow{\lambda} \langle s_1'; s_2, M'\rangle}$$

$$(N6) \qquad \frac{\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle}{\langle !s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle}$$

$$(N7) \qquad \frac{\langle s \parallel !s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle}{\langle !s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle}$$

$$(N8) \qquad \frac{\langle s[\overline{x} := \overline{v}], M\rangle \xrightarrow{\lambda} \langle s', M'\rangle}{\langle S(\overline{v}), M\rangle \xrightarrow{\lambda} \langle s', M'\rangle} \qquad \text{if } S(\overline{x}) \mathrel{\widehat{=}} s$$

$$(N9) \qquad \frac{\begin{array}{c} s \equiv t \\ \langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle \\ s' \equiv t' \end{array}}{\langle t, M\rangle \xrightarrow{\lambda} \langle t', M'\rangle}$$

Figure 2: Structured Operational Semantics of Schedules

Definition 1 *A transition* $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle$ *is called a* single step *transition if it is derived without inferences by rule* $(N3)$ *or rule* $(N4)$.

The single-step transitions are exactly those transitions that can be derived from the execution of a single rewrite rule. Inference rules $(N3)$ and $(N4)$ allow the parallel execution of multiple rewrite rules to be modelled by a single transition. Hence these inference rules are responsible for the multi-step character of the semantics.

The operational semantics describes behavioural aspects of our coordination language. A particular aspect of interest is the parallelism in the behaviour of coordination strategies. An important reason for using multi-step semantics is that it distinguishes parallel execution from interleaved execution which the single-step semantics does not.

A property of the operational semantics in Figure 2 is that any multi-step transition can be split into a sequence of single-step transitions which has the same effect on the multiset. This has as a consequence that sequential behaviour is a special case of parallel behaviour.

Lemma 2.1 *If* $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle$*, then there exist* $\lambda_1, \ldots, \lambda_n, n \geq 1$ *such that*

$$\langle s_0, M_0\rangle \xrightarrow{\lambda_1} \langle s_1, M_1\rangle \ldots \xrightarrow{\lambda_i} \ldots \langle s_{n-1}, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle s_n, M_n\rangle$$

where $\langle s, M\rangle = \langle s_0, M_0\rangle$ *and* $\langle s_n, M_n\rangle = \langle s', M'\rangle$ *and each* $\langle s_{i-1}, M_{i-1}\rangle \xrightarrow{\lambda_i} \langle s_i, M_i\rangle$, $1 \leq i \leq n$, *is a single-step transition*

Proof By transition induction. We consider the different ways in which the last step of the inference is done:

- By (N0), with $s \equiv r \to s_1[s_2]$ and $\lambda = \varepsilon$, from $\nexists v \subseteq M : b[\overline{x} := \overline{v}]$.
  This transition is clearly single-step.
- By (N1), with $s \equiv r \to s_1[s_2]$ and $\lambda = \sigma = m[\overline{x} := \overline{v}]/\overline{v}$, from $\exists v \subseteq M : b[\overline{x} := \overline{v}]$. This transition is clearly single-step.
- By (N2), with $s \equiv s_1 \parallel s_2$, from $\langle s_1, M\rangle \xrightarrow{\lambda} \langle s_1', M'\rangle$. Then by the induction hypothesis

$$\langle s_1, M\rangle \xrightarrow{\lambda_1} \langle s_{1,1}, M_1\rangle \ldots \xrightarrow{\lambda_i} \ldots \langle s_{1,n-1}, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle s_{1,n}, M_n\rangle$$

  where $\langle s_{1,n}, M_n\rangle = \langle s_1', M'\rangle$ and each transition is single-step.
  By repeated use of $(N2)$ we derive

$$\langle s_1 \parallel s_2, M\rangle \xrightarrow{\lambda_1} \langle s_{1,1} \parallel s_2, M_1\rangle \ldots \xrightarrow{\lambda_i} \ldots \langle s_{1,n-1} \parallel s_2, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle s_1' \parallel s_2, M'\rangle$$

- By (N2), with $s \equiv s_1 \parallel s_2$, from $\langle s_2, M\rangle \xrightarrow{\lambda} \langle s_2', M'\rangle$. Analogous to the previous case.
- By (N3), with $s \equiv s_1 \parallel s_2$, from $\langle s_1, M\rangle \xrightarrow{\varepsilon} \langle s_1', M'\rangle$ and $\langle s_2, M\rangle \xrightarrow{\lambda} \langle s_2', M\rangle$.
  The induction hypothesis applies to both of these transition. This gives

$$\langle s_1, M\rangle \xrightarrow{\varepsilon} \langle s_{1,1}, M\rangle \ldots \xrightarrow{\varepsilon} \ldots \langle s_{1,n-1}, M\rangle \xrightarrow{\varepsilon} \langle s_{1,n}, M\rangle$$

where $s_{1,n} = s'_1$ and each transition is single-step and

$$\langle s_2, M\rangle \xrightarrow{\lambda_1} \langle s_{2,1}, M_1\rangle \ldots \xrightarrow{\lambda_i} \ldots \langle s_{2,n-1}, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle s_{2,n}, M_n\rangle$$

where $\langle s_{2,n}, M_n\rangle = \langle s'_2, M'\rangle$ and each transition is single-step.
By repeated use of $(N2)$ we derive

$$\langle s_1 \parallel s_2, M\rangle \xrightarrow{\varepsilon} \langle s_{1,1} \parallel s_2, M\rangle \ldots \xrightarrow{\varepsilon} \ldots \langle s_{1,n-1} \parallel s_2, M\rangle \xrightarrow{\varepsilon} \langle s'_1 \parallel s_2, M\rangle$$

and

$$\langle s'_1 \parallel s_2, M\rangle \xrightarrow{\lambda_1} \langle s'_1 \parallel s_{2,1}, M_1\rangle \ldots \xrightarrow{\lambda_i} \ldots \langle s'_1 \parallel s_{2,n-1}, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle s'_1 \parallel s'_2, M'\rangle$$

The result follows by concatenating these sequences:

$$\langle s_1 \parallel s_2, M\rangle \xrightarrow{\varepsilon} \ldots \xrightarrow{\varepsilon} \langle s'_1 \parallel s_2, M\rangle \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_n} \langle s'_1 \parallel s'_2, M'\rangle$$

- By (N3), with $s \equiv s_1 \parallel s_2$, from $\langle s_1, M\rangle \xrightarrow{\lambda_1} \langle s'_1, M'\rangle$ and $\langle s_2, M\rangle \xrightarrow{\varepsilon} \langle s'_2, M\rangle$.
The proof is analogous to the previous case.

- By (N4), with $s \equiv s_1 \parallel s_2$, from $\langle s_1, M\rangle \xrightarrow{\sigma_1} \langle s'_1, M_1\rangle$ and $\langle s_2, M\rangle \xrightarrow{\sigma_2} \langle s'_2, M_2\rangle$ where $M \models \sigma_1 \bowtie \sigma_2$ and $\sigma = \sigma_1 \cdot \sigma_2$. The induction hypothesis applies to both, which gives

$$\langle s_1, M\rangle \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_{n1}} \langle s'_1, M_1\rangle \quad \text{and} \quad \langle s_2, M\rangle \xrightarrow{\lambda'_1} \ldots \xrightarrow{\lambda'_{n2}} \langle s'_2, M_2\rangle$$

where each transition is single-step. By repeated use of $(N2)$ we derive

$$\langle s_1 \parallel s_2, M\rangle \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_{n1}} \langle s'_1 \parallel s_2, M_1\rangle \quad \text{and} \quad \langle s'_1 \parallel s_2, M\rangle \xrightarrow{\lambda'_1} \ldots \xrightarrow{\lambda'_{n2}} \langle s'_1 \parallel s'_2, M_2\rangle$$

We concatenate these sequences into

$$\langle s_1 \parallel s_2, M\rangle \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_{n1}} \langle s'_1 \parallel s_2, M_1\rangle \xrightarrow{\lambda'_1} \ldots \xrightarrow{\lambda'_{n2}} \langle s'_1 \parallel s'_2, M'\rangle$$

for which the proposition clearly holds.

- By (N5) if $s \equiv s_1; s_2$. The proof is analogous to the previous case.

- By (N6), if $s \equiv !s$, from $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle$. From the induction hypothesis follows

$$\langle s_0, M_0\rangle \xrightarrow{\lambda_1} \langle s_1, M_1\rangle \ldots \xrightarrow{\lambda_i} \ldots \langle s_{n-1}, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle s_n, M_n\rangle$$

where $\langle s, M\rangle = \langle s_0, M_0\rangle$ and $\langle s_n, M_n\rangle = \langle s', M'\rangle$ and each transition is single-step.
For the first transition we use $(N6)$ to derive $\langle !s, M\rangle \xrightarrow{\lambda_1} \langle s_1, M_1\rangle$. By transitivity of $\longrightarrow$ follows $\langle !s, M\rangle \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_n} \langle s', M'\rangle$.

- By (N7), if $s \equiv !s$, from $\langle s \,\|\, !s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle$. From the induction hypothesis follows

  $$\langle s_0, M_0\rangle \xrightarrow{\lambda_1} \langle s_1, M_1\rangle \ldots \xrightarrow{\lambda_i} \ldots \langle s_{n-1}, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle s_n, M_n\rangle$$

  where $\langle s \,\|\, !s, M\rangle = \langle s_0, M_0\rangle$ and $\langle s_n, M_n\rangle = \langle s', M'\rangle$ and each transition is single-step. For the first transition we use $(N7)$ to infer $\langle !s, M\rangle \xrightarrow{\lambda_1} \langle s_1, M_1\rangle$. By transitivity of $\longrightarrow$ follows $\langle !s, M\rangle \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_n} \langle s', M'\rangle$.

- By (N8), for $s \equiv S(\overline{v})$, where $S(\overline{x}) \mathrel{\widehat{=}} s$:
  A transition can be derived, (only) by (N8), from $\langle s[\overline{x} := \overline{v}], M\rangle \xrightarrow{\lambda} \langle s', M'\rangle$.
  By the induction hypothesis

  $$\langle s_0, M_0\rangle \xrightarrow{\lambda_1} \langle s_1, M_1\rangle \ldots \xrightarrow{\lambda_i} \ldots \langle s_{n-1}, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle s_n, M_n\rangle$$

  where $\langle s_0, M_0\rangle = \langle s[\overline{x} := \overline{v}], M\rangle$ and $\langle s_n, M_n\rangle = \langle s', M'\rangle$ and each transition is single-step. By $(N8)$ we conclude $\langle S(\overline{v}), M\rangle \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_n} \langle s', M'\rangle$.

- By (N9), from $s \equiv t$, $s' \equiv t'$ and $\langle t, M\rangle \xrightarrow{\lambda} \langle t', M'\rangle$. By the induction hypothesis

  $$\langle t_0, M_0\rangle \xrightarrow{\lambda_1} \langle t_1, M_1\rangle \ldots \xrightarrow{\lambda_i} \ldots \langle t_{n-1}, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle t_n, M_n\rangle$$

  where $\langle t_0, M_0\rangle = \langle t, M\rangle$ and $\langle t_n, M_n\rangle = \langle t', M'\rangle$ and each transition is single-step. By (N9) we infer from the first and the last of this sequence of transitions $\langle s, M\rangle \xrightarrow{\lambda_1} \langle t_1, M_1\rangle$ and $\langle t_{n-1}, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle s', M'\rangle$. Hence by transitivity of $\longrightarrow$:

  $$\langle s, M\rangle \xrightarrow{\lambda_1} \langle t_1, M_1\rangle \ldots \xrightarrow{\lambda_i} \ldots \langle t_{n-1}, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle s', M'\rangle$$

□

2.2.3 The Most General Schedule

The schedule language allows us to specify behaviours from a wide spectrum of possibilities, ranging from the chaotic execution of a Gamma program to the completely deterministic behaviour of known algorithms. The former can be seen by constructing a schedule that comprises all possible behaviours of a Gamma program. We refer to this schedule as the *most general schedule*, and for a given program, it can be defined as follows.

Definition 2 *Let R be a simple Gamma program $r_1 + \ldots + r_n$ and let P_1 and P_2 be arbitrary Gamma programs. The most general schedules for R and $P_1 \circ P_2$ are defined as*

$$\begin{array}{lcl} \Gamma_R & \widehat{=} & !(r_1 \to \Gamma_R \,\|\, \ldots \,\|\, r_n \to \Gamma_R) \\ \Gamma_{P_1 \circ P_2} & \widehat{=} & \Gamma_{P_2}; \Gamma_{P_1} \end{array}$$

In [3] it was shown that a Gamma program P and its most general schedule Γ_P are equivalent with respect to their possible transition sequences. Therefore, Γ_P can be thought of as a model for the operational semantics of P.

Due to this property, the most general schedule plays a crucial rôle in the derivation process. The fact that the most general schedule explicitly represents the behaviour that is implicit in Gamma programs, makes it amenable to formal manipulation. Hence, the most general schedule may serve as starting point in the derivation process. A successive series of refinements of the most general schedule should enable the programmer to derive more deterministic execution strategies.

3 Notions of Refinement for Schedules

We make a separation between computation and coordination because they concern aspects of programming of a different nature. For both these aspects, a notion of refinement can be defined that suits the conceptual nature of that aspect.

A common notion of refinement that is suitable for the computational aspect, is subset inclusion of the set of possible outcomes. This approach has been used to investigate refinement of Gamma programs by formulating a capability function which captures the input-output relation of programs [12].

A coordination component is intended to capture the behavioural aspects of a program. Hence for the refinement of the coordination component, a notion should be used that addresses the issue of behaviour. In our setting, we consider a schedule to be a refinement of another schedule if it constitutes a more deterministic description of the behaviour. Limiting the behaviour may have as a side-effect (for nondeterministic programs) that the set of possible outcomes is reduced, hence also results in a refinement of the computation.

3.1 Statebased Refinement

The starting point for our investigations into refinement is the notion of bisimulation. This notion was successfully used for comparing behaviours of communicating (parallel) processes [13] and automaton [14]. In order to apply the theory of bisimulation to our setting, we need to make the following modifications.

In CCS [13], process and state are identified, suggesting that every process has a local state which can only be accessed by other processes through message-passing communication. An essential feature of the Gamma model is its use of a shared dataspace. In a shared dataspace, all processes may concurrently operate upon the current state, and any change is noticed by all processes. Clearly, the behaviour of schedules depends on the (global) state, therefore, we are concerned with the behaviour of configurations $\langle s, M \rangle$ rather than schedules in isolation.

Additionally, bisimulation induces an equivalence relation, while we are interested in a partial ordering of refinements where s is considered to be a refinement of t, if s can be simulated by t, but not necessarily the other way around. Such an ordering can be obtained by breaking the symmetry of bisimulation which leads to the following characterization of refinement: s can be simulated by t, if every action of s can be matched by t. This characterization is not sufficient to preserve total correctness, because s may terminate prematurely (after having displayed only a prefix of the behaviour of t.) We adapt the definition of simulation in such a way that the modified version preserves termination. We achieve this by adding a clause which requires that a refining schedule may terminate only if the refined schedule may terminate. Thus we arrive at following definition. Recall that $\mathbb{C} = \mathbb{S} \times \mathbb{M}$ where $\mathbb{C}$ denotes the set of configurations, $\mathbb{S}$ denotes the set of schedules and $\mathbb{M}$ denotes the set of multisets.

Definition 3 *A binary relation on configurations $\mathcal{R} \subseteq \mathbb{C} \times \mathbb{C}$ is a* strong statebased simulation *if $(\langle s, M\rangle, \langle t, N\rangle) \in \mathcal{R}$ implies, for all λ,*

i. $N = M$

ii. $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle \Rightarrow \exists t' : \langle t, M\rangle \xrightarrow{\lambda} \langle t', M'\rangle$ *such that* $(\langle s', M'\rangle, \langle t', M'\rangle) \in \mathcal{R}$

iii. $s \equiv \mathsf{skip} \ \Rightarrow t \equiv \mathsf{skip}$

For a binary relation $\mathcal{R}$ on configurations, we sometimes write $\langle s, M\rangle \mathcal{R} \langle t, M\rangle$ to mean $(\langle s, M\rangle, \langle t, M\rangle) \in \mathcal{R}$.

In compliance with [13], this notion of simulation is called *strong* simulation because every single transition of the refining schedule can be mimicked by a single corresponding transition of the refined schedule. Below we shall relax on this property by introducing a weak notion of refinement. The adjective *statebased* is added, because the current state of a computation is taken into account – this in contrast to the stateless notion presented in the next section.

We show some basic properties of strong statebased simulation.

Lemma 3.1 *Let $\mathcal{R}_i$ for $i = 1, 2, \ldots$ be strong statebased simulations. Then the following are also strong statebased simulations*

1. *the identity relation on configurations $Id_{\mathbb{C}} = \{(\langle s, M\rangle, \langle s, M\rangle) \mid s \in \mathbb{S}, M \in \mathbb{M}\}$,*
2. *the composition: $\mathcal{R}_1\mathcal{R}_2$,*
3. *the union: $\bigcup_{i\in I} \mathcal{R}_i$.*

Proof

1. By reflexivity of $=$ and $\Rightarrow$.
2. Let $\mathcal{R} = \mathcal{R}_1\mathcal{R}_2$. Suppose $(\langle s_1, M\rangle, \langle s_2, N\rangle) \in \mathcal{R}$.
   Then for some t we have $(\langle s_1, M\rangle, \langle t, N'\rangle) \in \mathcal{R}_1$ and $(\langle t, N'\rangle, \langle s_2, N\rangle) \in \mathcal{R}_2$.
   Because $\mathcal{R}_1$ and $\mathcal{R}_2$ are strong statebased simulations, we have $M = N'$ and $N' = N$, hence $M = N' = N$.
   Now let $\langle s_1, M\rangle \xrightarrow{\lambda} \langle s_1', M'\rangle$.
   Because $(\langle s_1, M\rangle, \langle t, M\rangle) \in \mathcal{R}_1$ we have $\langle t, M\rangle \xrightarrow{\lambda} \langle t', M'\rangle$ and $(\langle s_1', M'\rangle, \langle t', M'\rangle) \in \mathcal{R}_1$.
   Because $(\langle t, M\rangle, \langle s_2, M\rangle) \in \mathcal{R}_2$, we have $\langle s_2, M\rangle \xrightarrow{\lambda} \langle s_2', M'\rangle$ and $(\langle t', M'\rangle, \langle s_2', M'\rangle) \in \mathcal{R}_2$.
   From $(\langle s_1', M'\rangle, \langle t', M'\rangle) \in \mathcal{R}_1$ and $(\langle t', M'\rangle, \langle s_2', M'\rangle) \in \mathcal{R}_2$ follows $(\langle s_1', M'\rangle, \langle s_2', M'\rangle) \in \mathcal{R}$.

If $s_1 \equiv \mathsf{skip}$ then, from $(\langle s_1, M\rangle, \langle t, M\rangle) \in \mathcal{R}_1$, we have $t \equiv \mathsf{skip}$. Then from $(\langle t, M\rangle, \langle s_2, M\rangle) \in \mathcal{R}_2$ we have $s_2 \equiv \mathsf{skip}$.

3. Let $\mathcal{R} = \bigcup_{i \in I} \mathcal{R}_i$. Suppose $(\langle s_1, M\rangle, \langle s_2, N\rangle) \in \mathcal{R}$.
   Then $(\langle s_1, M\rangle, \langle s_2, N\rangle) \in \mathcal{R}_i$ for some $i \in I$, hence $N = M$.
   If $\langle s_1, M\rangle \xrightarrow{\lambda} \langle s_1', M'\rangle$, then because $\mathcal{R}_i$ is a strong statebased simulation, we have $\langle s_2, M\rangle \xrightarrow{\lambda} \langle s_2', M'\rangle$ and $(\langle s_1', M'\rangle, \langle s_2', M'\rangle) \in \mathcal{R}_i$.
   Because $\mathcal{R}_i \subseteq \mathcal{R}$ also $(\langle s_1', M'\rangle, \langle s_2', M'\rangle) \in \mathcal{R}$.
   The case $s_1 \equiv \mathsf{skip}$ goes analogously.

$\square$

Let $\langle s, M\rangle$ and $\langle t, M\rangle$ be configurations. We say that $\langle s, M\rangle$ is a *strong statebased refinement* of $\langle t, M\rangle$, denoted $\langle s, M\rangle \leqq \langle t, M\rangle$, if $(\langle s, M\rangle, \langle t, M\rangle) \in \mathcal{R}$ for some strong statebased simulation $\mathcal{R}$. Hence, we define the strong statebased refinement relation as the maximal strong statebased simulation. Strong statebased equivalence is defined as the intersection of strong statebased refinement and its inverse.

Definition 4

1. $\leqq \; = \bigcup\{\mathcal{R} \mid \mathcal{R}$ *is a strong statebased simulation* $\}$
2. $\cong \; = \; \leqq \cap \leqq^{-1}$

Lemma 3.2

1. $\leqq$ *is the largest strong statebased simulation.*
2. $\leqq$ *is a partial order.*
3. $\cong$ *is an equivalence relation.*

Proof

1. By Lemma 3.1.3 $\leqq$ is a strong statebased simulation and by Definition 3.2 it includes any other such.
2. *Reflexivity*: By Lemma 3.1.1.
   Transitivity: By Lemma 3.1.2.
   Antisymmetry If $\langle s, M\rangle \leqq \langle t, M\rangle$ and $\langle t, M\rangle \leqq \langle s, M\rangle$, then by Definition 3.2.2 $\langle s, M\rangle \cong \langle t, M\rangle$.
3. $\cong$ is reflexive and transitive by Lemma 3.2.(1 and 2).
   Symmetry follows from Definition 3.2.2.

$\square$

Analogously to [13], we use some fixed-point theory (see e.g. [9]) to show that $\leqq$ defines the relation that contains precisely all strong statebased simulations.

Definition 5 *Define a function* $\mathbb{F} : \mathbb{C} \times \mathbb{C} \to \mathbb{C} \times \mathbb{C}$ *as follows:*
If $\mathcal{R} \subseteq \mathbb{C} \times \mathbb{C}$, *then* $(\langle s, M\rangle, \langle t, M\rangle) \in \mathbb{F}(\mathcal{R})$ *if and only if, for all* λ,

1. $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle \Rightarrow \langle t, M\rangle \xrightarrow{\lambda} \langle t', M'\rangle \wedge (\langle s', M'\rangle, \langle t', M'\rangle) \in \mathcal{R}$
2. $s \equiv \mathsf{skip} \Rightarrow t \equiv \mathsf{skip}$

Lemma 3.3

1. $\mathbb{F}$ *is monotonic; i.e. if* $\mathcal{R}_1 \subseteq \mathcal{R}_2$, *then* $\mathbb{F}(\mathcal{R}_1) \subseteq \mathbb{F}(\mathcal{R}_2)$.
2. $\mathcal{R}$ *is a* strong statebased simulation *if and only if* $\mathcal{R} \subseteq \mathbb{F}(\mathcal{R})$.

Proof

1. Follows directly from Definition 3.3 of $\mathbb{F}$.
2. Follows directly from Definition 3.3 of $\mathbb{F}$ and Definition 3.1 of strong statebased simulation.

□

Monotonicity says that $\mathbb{F}$ preserves the ordering $\subseteq$ on $\mathbb{P}(\mathbb{C}\times\mathbb{C} \to \mathbb{C}\times\mathbb{C})$. We call $\mathcal{R}$ a *fixed-point* of $\mathbb{F}$ if $\mathcal{R} = \mathbb{F}(\mathcal{R})$. Similarly, we say that $\mathcal{R}$ is a *pre-fixed-point* of $\mathbb{F}$ if $\mathcal{R} \subseteq \mathbb{F}(\mathcal{R})$. So statebased simulations are, by Lemma 3.3.2, exactly the pre-fixed-points of $\mathbb{F}$, and we wish to show that $\leqq$, which is, by Lemma 3.2, the largest pre-fixed-point, is a fixed-point of $\mathbb{F}$.

Lemma 3.4 $\leqq$ *is a largest fixed point of* $\mathbb{F}$.

Proof

- $\leqq\ \subseteq \mathbb{F}(\leqq)$: By Lemma 3.2, $\leqq$ is a strong statebased simulation. By Lemma 3.3.2 then follows $\leqq\ \subseteq \mathbb{F}(\leqq)$.
- $\mathbb{F}(\leqq) \subseteq\ \leqq$: Monotonicity of $\mathbb{F}$ implies $\mathbb{F}(\leqq) \subseteq \mathbb{F}(\mathbb{F}(\leqq))$; i.e. $\mathbb{F}(\leqq)$ is a pre-fixed point of $\mathbb{F}$. But because $\leqq$ is the largest pre-fixed point, it includes $\mathbb{F}(\leqq)$, i.e. $\mathbb{F}(\leqq) \subseteq\ \leqq$.

Moreover, $\leqq$ must be a largest fixed point of $\mathbb{F}$, because it is the largest pre-fixed point. □

Thus $\leqq$ is the largest relation that satisfies the definition of strong statebased simulation.

The *up-to* method (from [13]) for proving simulations can be carried over onto strong statebased refinement.

Definition 6 *A relation* $\mathcal{R} \subseteq \mathbb{C} \times \mathbb{C}$ *is a* strong statebased simulation up-to strong statebased refinement *if* $(\langle s, M\rangle, \langle t, M\rangle) \in \mathcal{R}$ *implies, for all* λ,

i. $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle \Rightarrow \exists t' : \langle t, M\rangle \xrightarrow{\lambda} \langle t', M'\rangle$ *and* $\langle s', M'\rangle \leqq \mathcal{R} \leqq \langle t', M'\rangle$
ii. $s \equiv \mathsf{skip} \Rightarrow t \equiv \mathsf{skip}$

Lemma 3.5 *If $\mathcal{R}$ is a strong statebased simulation up-to strong statebased refinement, then $\leqq \mathcal{R} \leqq$ is a strong statebased simulation.*

Proof Assume $\langle s, M\rangle \leqq \mathcal{R} \leqq \langle t, M\rangle$.
Then there are u and v such that $\langle s, M\rangle \leqq \langle u, M\rangle \mathcal{R} \langle v, M\rangle \leqq \langle t, M\rangle$.

transition
Assume $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle$.
Then by $\langle s, M\rangle \leqq \langle u, M\rangle$, $\langle u, M\rangle \xrightarrow{\lambda} \langle u', M'\rangle$ such that $\langle s', M'\rangle \leqq \langle u', M'\rangle$.
By $\langle u, M\rangle \mathcal{R} \langle v, M\rangle$ follows $\langle u, M\rangle \xrightarrow{\lambda} \langle v', M'\rangle$ such that $\langle u', M'\rangle \leqq \mathcal{R} \leqq \langle v', M'\rangle$.
By $\langle v, M\rangle \leqq \langle t, M\rangle$ follows $\langle t, M\rangle \xrightarrow{\lambda} \langle t', M'\rangle$ such that $\langle u', M'\rangle \leqq \langle t', M'\rangle$.
Hence $\langle s', M'\rangle \leqq\ \leqq \mathcal{R} \leqq\ \leqq \langle t', M'\rangle$. By transitivity of $\leqq$: $\langle s', M'\rangle \leqq \mathcal{R} \leqq \langle t', M'\rangle$.

termination
If $s \equiv \mathsf{skip}$, then by $\langle s, M\rangle \leqq \langle u, M\rangle$, $u \equiv \mathsf{skip}$. By $\langle u, M\rangle \mathcal{R} \langle v, M\rangle$ follows $v \equiv \mathsf{skip}$.
By $\langle v, M\rangle \leqq \langle t, M\rangle$ follows $t \equiv \mathsf{skip}$. □

Lemma 3.6
If $\mathcal{R}$ is a strong statebased simulation up-to strong statebased refinement, then $\mathcal{R} \subseteq\ \leqq$.

Proof
By Lemma 3.5, $\leqq \mathcal{R} \leqq$ is a strong statebased simulation, hence by Lemma 3.2, $\leqq \mathcal{R} \leqq\ \subseteq\ \leqq$.
From $Id_{\mathbb{C}} \subseteq\ \leqq$ follows $\mathcal{R} \subseteq\ \leqq$. □

An important feature of the current notion of refinement is its ability to exploit properties of the multiset. This is possible because the multiset is an explicit component of the simulation relation. The following example illustrates the idea.

Example 3.6.1 *Consider a Gamma program for computing the sum of a multiset of numbers:*

$$add \mathrel{\widehat{=}} x, y \mapsto x + y \Leftarrow true$$

The program operates by adding pairs of numbers from the initial multiset in any order - possibly in parallel. This behaviour is equivalently expressed by the program's most general schedule:

$$\Gamma_{add} \mathrel{\widehat{=}} !(add \to \Gamma_{add})$$

A more deterministic schedule may exploit the size, say n, of the initial multiset by performing exactly $n-1$ additions in sequence, e.g. by $Sum(n)$ where

$$Sum(i) \mathrel{\widehat{=}} i > 1 \triangleright (add; Sum(i-1))$$

Now we would expect Sum(n) *to be a refinement of* Γ_{add}*, i.e.* $\langle Sum(n), M\rangle \leqq \langle \Gamma_{add}, M\rangle$ *with* $\#M = n$*. However, the following counterexample disproves this refinement. Consider, for instance, the initial multiset* $\{1,5,3\}$*. Then Sum*(3) *may perform the following transition sequence.*

$$\langle Sum(3), \{1,5,3\}\rangle \xrightarrow{\{6\}/\{1,5\}} \langle Sum(2), \{6,3\}\rangle \xrightarrow{\{9\}/\{6,3\}} \langle \mathsf{skip}, \{9\}\rangle$$

The most general schedule Γ_{add} *can also make these transitions, but inevitably needs an additional* ε*-transition to detect termination, e.g.*

$$\langle \Gamma_{add}, \{1,5,3\}\rangle \xrightarrow{\{6\}/\{1,5\}} \langle \Gamma_{add}, \{6,3\}\rangle \xrightarrow{\{9\}/\{6,3\}} \langle \Gamma_{add}, \{9\}\rangle \xrightarrow{\varepsilon} \langle \mathsf{skip}, \{9\}\rangle$$

□

If the behaviour of a configuration $\langle s, M\rangle$ differs from that of another configuration $\langle t, M\rangle$ only by the fact that it makes a different number of ε-transitions, we still want to consider the former a refinement of the latter because ε transitions do not change the multiset, hence do not change the input-output behaviour. In the next section we propose a more liberal notion of refinement that supports this intuition.

3.2 Weak Statebased Simulation

In the Section 3.1 we observed that strong statebased refinement does not justify refinements where the only difference between configurations is the number of ε-steps they may make. From the semantic rules in Figure 2 we see that ε-transitions do not change the multiset. So adding or removing ε-transitions in a transition sequence cannot change the outcome of a computation. Analogously to [13] this brings us to define a weak notion of refinement, that is insensitive to ε-transitions.

We define the transition relation $\xrightarrow{\overline{\lambda}}{}^*$ as the reflexive transitive closure of the transition relation $\longrightarrow$ from Figure 2. The label $\overline{\lambda}$ denotes the sequence obtained by concatenating, in order, all individual labels of the constituent transitions. Furthermore, we use $\widehat{\lambda}$ to denote the sequence $\overline{\lambda}$ where all occurrences of ε have been removed.

Definition 7 *A relation* $\mathcal{R} \subseteq \mathbb{C} \times \mathbb{C}$ *is a* weak statebased simulation *if,*
for all $(\langle s, M\rangle, \langle t, N\rangle) \in \mathcal{R}$

i. $M = N$

ii. $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle \Rightarrow \exists t' : \langle t, M\rangle \xrightarrow{\overline{\lambda'}}{}^* \langle t', M'\rangle$ *where* $\widehat{\lambda'} = \widehat{\lambda}$ *and* $(\langle s', M'\rangle, \langle t', M'\rangle) \in \mathcal{R}$

iii. $s \equiv \mathsf{skip} \Rightarrow \langle t, M\rangle \xrightarrow{\overline{\lambda'}}{}^* \langle \mathsf{skip}, M\rangle$ *where* $\widehat{\lambda'} = \langle\ \rangle$

Lemma 3.7 *Let* $\mathcal{R}_i$ *for* $i = 1, 2, \ldots$ *be weak statebased simulations. Then the following are also weak statebased simulations*

1. *the identity* $Id_{\mathbb{C}} = \{(\langle s, M\rangle, \langle s, M\rangle) \mid s \in \mathbb{S}, M \in \mathbb{M}\}$,

2. the composition $\mathcal{R}_1\mathcal{R}_2$,

3. the union $\bigcup_{i\in I}\mathcal{R}_i$.

Proof

1. Follows by $\longrightarrow\ \subseteq\ \longrightarrow^*$ and reflexivity of $\Rightarrow$ and $=$.

2. Let $\mathcal{R}=\mathcal{R}_1\mathcal{R}_2$. Suppose $(\langle s_1,M\rangle,\langle s_2,N\rangle)\in\mathcal{R}$, then for some t we have $(\langle s_1,M\rangle,\langle t,N'\rangle)\in\mathcal{R}_1$ and $(\langle t,N'\rangle,\langle s_2,N\rangle)\in\mathcal{R}_2$. Because $\mathcal{R}_1$ and $\mathcal{R}_2$ are weak statebased simulations, we have $M=N'=N$.
   Now let $\langle s_1,M\rangle\xrightarrow{\lambda}\langle s_1',M'\rangle$. Because $(\langle s_1,M\rangle,\langle t,M\rangle)\in\mathcal{R}_1$ we have $\langle t,M\rangle\xrightarrow{\overline{\lambda'}}{}^*\langle t',M'\rangle$ where $\widehat{\lambda}=\widehat{\lambda'}$ and $(\langle s_1',M'\rangle,\langle t',M'\rangle)\in\mathcal{R}_1$. By definition of $\longrightarrow^*$, $\lambda'=\langle\,\lambda_1',\dots,\lambda_n'\,\rangle$ such that
   $$\langle t_0,M_0\rangle\xrightarrow{\lambda_1'}\langle t_1,M_1\rangle\xrightarrow{\lambda_2'}\dots\xrightarrow{\lambda_{n-1}'}\langle t_{n-1},M_{n-1}\rangle\xrightarrow{\lambda_n'}\langle t_n,M_n\rangle$$
   where $t=t_0, M=M_0, t'=t_n$ and $M'=M_n$. Because $(\langle t_0,M\rangle,\langle s_2,M\rangle)\in\mathcal{R}_2$, we have, from the first transition, $\langle s_2,M\rangle\xrightarrow{\overline{\lambda_1''}}{}^*\langle s_{2,1},M_1\rangle$ where $\widehat{\lambda_1'}=\widehat{\lambda_1''}$ and $(\langle t_1,M_1\rangle,\langle s_{2,1},M_1\rangle)\in\mathcal{R}_2$. By induction on n it can be shown that
   $$\langle s_{2,0},M_0\rangle\xrightarrow{\overline{\lambda_1''}}{}^*\langle s_{2,1},M_1\rangle\xrightarrow{\overline{\lambda_2''}}{}^*\dots\xrightarrow{\overline{\lambda_{n-1}''}}{}^*\langle s_{2,n-1},M_{n-1}\rangle\xrightarrow{\overline{\lambda_n''}}{}^*\langle s_{2,n},M_n\rangle$$
   where $\widehat{\lambda_i'}=\widehat{\lambda_i''}$ and $(\langle t_i,M_i\rangle,\langle s_{2,i},M_i\rangle)\in\mathcal{R}_2$ for all $1\le i\le n$ and $s_2=s_{2,0}, M=M_0$, $s_2'=s_{2,n}$ and $M'=M_n$. Hence $\langle s_2,M\rangle\xrightarrow{\overline{\lambda''}}{}^*\langle s_2',M'\rangle$ where $\widehat{\lambda''}=\widehat{\lambda'}$ and $(\langle t',M'\rangle,\langle s_2',M'\rangle)\in\mathcal{R}_2$. From $(\langle s_1',M'\rangle,\langle t',M'\rangle)\in\mathcal{R}_1$ and $(\langle t',M'\rangle,\langle s_2',M'\rangle)\in\mathcal{R}_2$ follows $(\langle s_1',M'\rangle,\langle s_2',M'\rangle)\in\mathcal{R}$.

   If $s_1\equiv\mathsf{skip}$ then, from $(\langle s_1,M\rangle,\langle t,M\rangle)\in\mathcal{R}_1$, we have $\langle t,M\rangle\xrightarrow{\overline{\lambda}}{}^*\langle\mathsf{skip},M\rangle$ where $\widehat{\lambda}=\langle\,\rangle$. Then from $(\langle t,M\rangle,\langle s_2,M\rangle)\in\mathcal{R}_2$ we have, analogous to the previous case, $\langle s_2,M\rangle\xrightarrow{\overline{\lambda'}}{}^*\langle\mathsf{skip},M\rangle$ where $\widehat{\lambda'}=\langle\,\rangle$.

3. Let $\mathcal{R}=\bigcup_{i\in I}\mathcal{R}_i$. Suppose $(\langle s_1,M\rangle,\langle s_2,N\rangle)\in\mathcal{R}$.
   Then $(\langle s_1,M\rangle,\langle s_2,N\rangle)\in\mathcal{R}_i$ for some $i\in I$, hence $M=N$.
   If $\langle s_1,M\rangle\xrightarrow{\lambda}\langle s_1',M'\rangle$, then because $\mathcal{R}_i$ is a weak statebased simulation, we have
   $\langle s_2,M\rangle\xrightarrow{\overline{\lambda'}}{}^*\langle s_2',M'\rangle$ where $\widehat{\lambda'}=\widehat{\lambda}$ and $(\langle s_1',M'\rangle,\langle s_2',M'\rangle)\in\mathcal{R}_i$.
   Because $\mathcal{R}_i\subseteq\mathcal{R}$ also $(\langle s_1',M'\rangle,\langle s_2',M'\rangle)\in\mathcal{R}$.
   The case $s_1\equiv\mathsf{skip}$ goes analogously.

$\square$

Let $\langle s,M\rangle$ and $\langle t,M\rangle$ be configurations. We say that $\langle s,M\rangle$ is a *weak statebased refinement* of $\langle t,M\rangle$, denoted $\langle s,M\rangle\precapprox\langle t,M\rangle$, if $(\langle s,M\rangle,\langle t,M\rangle)\in\mathcal{R}$ for some weak statebased simulation $\mathcal{R}$. As is standard, weak statebased equivalence is defined as the kernel of weak statebased refinement.

Definition 8

1. $\precapprox \; = \bigcup\{\mathcal{R} \mid \mathcal{R}$ *is a weak statebased simulation* $\}$

2. $\approx \; = \; \precapprox \cap \precapprox^{-1}$

Lemma 3.8

1. $\precapprox$ *is the largest weak statebased simulation.*

2. $\precapprox$ *is a partial order.*

3. $\approx$ *is an equivalence relation.*

Proof Analogous to the proofs of Lemma 3.2. □

Analogously to strong statebased refinement in the previous section, it can be shown that $\precapprox$ defines the relation that contains precisely all weak statebased simulations.

Definition 9 *Define a function* $\mathbb{F} : \mathbb{C} \times \mathbb{C} \to \mathbb{C} \times \mathbb{C}$ *as follows:*
If $\mathcal{R} \subseteq \mathbb{C} \times \mathbb{C}$*, then* $(\langle s, M\rangle, \langle t, M\rangle) \in \mathbb{F}(\mathcal{R})$ *if and only if, for all* λ*,*

$$\begin{array}{ll} i. & \langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle \Rightarrow \exists t' : \langle t, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t', M'\rangle \text{ where } \widehat{\lambda'} = \widehat{\lambda} \text{ and } (\langle s', M'\rangle, \langle t', M'\rangle) \in \mathcal{R} \\ ii. & s \equiv \mathsf{skip} \;\Rightarrow \langle t, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle \mathsf{skip}, M\rangle \text{ where } \widehat{\lambda'} = \langle\ \rangle \end{array}$$

Lemma 3.9 $\precapprox$ *is a largest fixed point of* $\mathbb{F}$.

Proof Analogous to the proof of Lemma 3.4. □

Next, we develop the *up-to* technique (from [13]) for proving weak statebased refinement.

Definition 10 *A relation* $\mathcal{R} \subseteq \mathbb{C} \times \mathbb{C}$ *is a* weak statebased simulation up-to weak statebased refinement *if, for all* $(\langle s, M\rangle, \langle t, M\rangle) \in \mathcal{R}$

$$\begin{array}{ll} i. & \langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle \Rightarrow \exists t' : \langle t, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t', M'\rangle \text{ where } \widehat{\lambda'} = \widehat{\lambda} \text{ and } \langle s', M'\rangle \precapprox \mathcal{R} \precapprox \langle t', M'\rangle \\ ii. & s \equiv \mathsf{skip} \;\Rightarrow \langle t, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle \mathsf{skip}, M\rangle \text{ where } \widehat{\lambda'} = \langle\ \rangle \end{array}$$

Lemma 3.10 *If* $\mathcal{R}$ *is a weak statebased simulation up-to weak statebased refinement, then* $\precapprox \mathcal{R} \precapprox$ *is a weak statebased simulation.*

Proof Assume $\langle s, M\rangle \precapprox \mathcal{R} \precapprox \langle t, M\rangle$.
Then there are u and v such that $\langle s, M\rangle \precapprox \langle u, M\rangle \mathcal{R} \langle v, M\rangle \precapprox \langle t, M\rangle$.

transition
Assume $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle$.
Then by $\langle s, M\rangle \precapprox \langle u, M\rangle$, $\langle u, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle u', M'\rangle$ such that $\widehat{\lambda'} = \widehat{\lambda}$ and $\langle s', M'\rangle \precapprox \langle u', M'\rangle$.

By $\langle u,M\rangle\mathcal{R}\langle v,M\rangle$ follows $\langle u,M\rangle \xrightarrow{\overline{\lambda''}}{}^{*} \langle v',M'\rangle$ such that $\widehat{\lambda''}=\widehat{\lambda'}$ and $\langle u',M'\rangle \underset{\approx}{\precsim} \mathcal{R} \underset{\approx}{\precsim} \langle v',M'\rangle$.
By $\langle v,M\rangle \underset{\approx}{\precsim} \langle t,M\rangle$ follows $\langle t,M\rangle \xrightarrow{\overline{\lambda'''}}{}^{*} \langle t',M'\rangle$ such that $\widehat{\lambda'''}=\widehat{\lambda''}$ and $\langle u',M'\rangle \underset{\approx}{\precsim} \langle t',M'\rangle$.
Hence $\widehat{\lambda'''}=\widehat{\lambda}$ and $\langle s',M'\rangle \underset{\approx}{\precsim}\ \underset{\approx}{\precsim} \mathcal{R} \underset{\approx}{\precsim}\ \underset{\approx}{\precsim} \langle t',M'\rangle$.
By transitivity of $\underset{\approx}{\precsim}$: $\langle s',M'\rangle \underset{\approx}{\precsim} \mathcal{R} \underset{\approx}{\precsim} \langle t',M'\rangle$.

termination
If $s \equiv \mathsf{skip}$, then by $\langle s,M\rangle \underset{\approx}{\precsim} \langle u,M\rangle$, $\langle u,M\rangle \xrightarrow{\overline{\lambda}}{}^{*} \langle \mathsf{skip},M\rangle$ such that $\widehat{\lambda}=\langle\ \rangle$.
By $\langle u,M\rangle\mathcal{R}\langle v,M\rangle$ follows $\langle v,M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle \mathsf{skip},M\rangle$ such that $\widehat{\lambda'}=\langle\ \rangle$.
By $\langle v,M\rangle \underset{\approx}{\precsim} \langle t,M\rangle$ follows $\langle t,M\rangle \xrightarrow{\overline{\lambda''}}{}^{*} \langle \mathsf{skip},M\rangle$ such that $\widehat{\lambda''}=\langle\ \rangle$. □

Lemma 3.11 *If $\mathcal{R}$ is a weak statebased simulation up-to weak statebased refinement, then $\mathcal{R} \subseteq \underset{\approx}{\precsim}$.*

Proof
By Lemma 3.10, $\underset{\approx}{\precsim} \mathcal{R} \underset{\approx}{\precsim}$ is a weak statebased simulation, hence by Lemma 3.8, $\underset{\approx}{\precsim} \mathcal{R} \underset{\approx}{\precsim}\ \subseteq\ \underset{\approx}{\precsim}$.
From $Id_{\mathbb{C}} \subseteq \underset{\approx}{\precsim}$ follows $\mathcal{R} \subseteq \underset{\approx}{\precsim}$. □

Using the weak notion of simulation we are now able to prove the refinement from Example 3.1.

Example 3.11.1 *Let $\mathcal{R} = \{(\langle Sum(n),M\rangle, \langle \Gamma_{add},M\rangle) \mid \#M = n, n \geq 0\}$.*
We prove that $\mathcal{R}$ is a weak statebased simulation by induction on n.

- *$n \leq 1$: then $Sum(0) \equiv \mathsf{skip}$. Because $\#M \leq 2$ we derive by $(N0)$, $(N6)$ and $(N9)$, $\langle \Gamma_{add},M\rangle \xrightarrow{\varepsilon} \langle \mathsf{skip},M\rangle$.*
  By definition of $\longrightarrow^{*}$ follows $\langle \Gamma_{add},M\rangle \xrightarrow{\langle\,\varepsilon\,\rangle}{}^{*} \langle \mathsf{skip},M\rangle$. Clearly $\widehat{\langle\,\varepsilon\,\rangle} = \langle\ \rangle$.
- *$n > 1$ and $\langle Sum(n),M\rangle \xrightarrow{\sigma} \langle Sum(n-1),M'\rangle$ where $\#M' = n-1$.*
  Then, by $(N1)$, $(N6)$ and $(N9)$, $\langle \Gamma_{add},M\rangle \xrightarrow{\sigma} \langle \Gamma_{add},M'\rangle$.
  By definition of $\longrightarrow^{*}$ $\langle \Gamma_{add},M\rangle \xrightarrow{\langle\,\sigma\,\rangle}{}^{*} \langle \Gamma_{add},M'\rangle$.
  By induction we have $(\langle Sum(n-1),M'\rangle, \langle \Gamma_{add},M'\rangle) \in \mathcal{R}$. □

The method of statebased simulation in principle suffices for proving any (valid) refinement. However, the operational reasoning using simulations that is required for proving refinements is considered to be rather complex and therefore error-prone.

A common approach, followed for instance by Milner [13], is to define an equivalence relation over programs (in terms of their semantics) and show that this relation is a congruence over program terms. The congruence property makes it possible to use program equivalences as equational laws to reason about programs in a modular (or compositional) fashion. Equational reasoning facilitates formal calculation and avoids the complexity of operational details.

If we want to follow this approach for the refinement of our coordination language, we have to show that our notions of refinement are precongruences for the set of schedules. Unfortunately, we run into the problem that our notions of refinement, $\leqq$ and $\underset{\approx}{\precsim}$, are relations over the set of configurations rather than over the set of schedules. This makes the notion of congruence meaningless because the composition operators ";" and " $\|$ " etc. over which we want our equivalence to be congruent operate on schedules.

The fact that the statebased notions of refinement are not precongruences, means that refinement cannot be applied in a modular fashion. Hence schedules need to be considered as a whole, which may result in complex proofs. In Section 3.4 we present results that facilitate proving refinements.

3.3 Relating Refinement and Capability

In [12] Hankin et al. define a so-called *capability* function which models the input-output behaviour of Gamma programs. Subsequently, they use the relational ordering (subset inclusion) of the set of possible outcomes as the basis of a calculus of refinement. In this section we show that the statebased notions of refinement preserve the relational ordering on schedules for Gamma programs.

The capability of a configuration is defined as the set of possible multisets it may produce, plus the special symbol $\perp$ if the configuration may never terminate.

Definition 11 *We define the divergence predicate* $\uparrow$ *on configurations:*
$\langle s, M\rangle\uparrow$ *if and only if*
$\langle s, M\rangle = \langle s_0, M_0\rangle$ *and for all* $i \geq 0$ *there exists a* λ_i *such that* $\langle s_i, M_i\rangle \xrightarrow{\lambda_i} \langle s_{i+1}, M_{i+1}\rangle$

Definition 12 *The capability function* $\mathcal{C} : \mathbb{S} \times \mathbb{M} \to \mathcal{P}(\mathbb{M}) \cup \{\perp\}$ *for schedules, is defined as*

$$\begin{aligned} \mathcal{C}(s, M) = \; & \{\perp \mid \langle s, M\rangle\uparrow\} \cup \\ & \{M' \mid \langle s, M\rangle \xrightarrow{\overline{\lambda}}{}^{*} \langle \mathsf{skip}, M'\rangle\} \end{aligned}$$

Theorem 3.12 *If* $\langle s, M\rangle$ *and* $\langle t, M\rangle$ *are configurations such that* $\langle s, M\rangle \leqq \langle t, M\rangle$*, then* $\mathcal{C}(s, M) \subseteq \mathcal{C}(t, M)$*.*

Proof Let $x \in \mathcal{C}(s, M)$, we have to show that $x \in \mathcal{C}(t, M)$.
Consider the following cases:

- $x = \perp$:
  Hence if $\langle s, M\rangle = \langle s_0, M_0\rangle$, then for all $i \geq 0$ there exists a λ_i such that $\langle s_i, M_i\rangle \xrightarrow{\lambda_i} \langle s_{i+1}, M_{i+1}\rangle$. By $\langle s, M\rangle \leqq \langle t, M\rangle$ follows $\langle t, M\rangle = \langle t_0, M_0\rangle$ and, by induction on the length of the transition sequence, for all $i \geq 0$ there exists a λ_i such that $\langle t_i, M_i\rangle \xrightarrow{\lambda_i} \langle t_{i+1}, M_{i+1}\rangle$. Hence $\perp \in \mathcal{C}(t, M)$.

- $x = M'$:
  Hence $\langle s, M\rangle \xrightarrow{\overline{\lambda}}{}^{*} \langle \mathsf{skip}, M'\rangle$. By $\langle s, M\rangle \leqq \langle t, M\rangle$ and induction on the length of the transition sequence, follows $\langle t, M\rangle \xrightarrow{\overline{\lambda}}{}^{*} \langle \mathsf{skip}, M'\rangle$. Hence $M' \in \mathcal{C}(t, M)$.

□

The power of weak refinement is that it is insensitive to a differing number of ε transitions. However, this has the undesirable consequence that weak refinement does not preserve total correctness. Because *fail* is weakly equivalent to skip, we may introduce an arbitrary number of failing transitions. In particular, we may introduce an infinite number of failing transitions which invalidates total correctness.

Example 3.12.1 *Let $F \mathrel{\widehat{=}} fail; F$. It is straightforward to prove that F is a weak refinement of* skip*; i.e. $\langle F, M\rangle \precapprox \langle \mathsf{skip}, M\rangle$ for any M. However replacing* skip *by F introduces a infinite sequence of ε transitions. In terms of the capability function, we have, for all M, $\mathcal{C}(F, M) = \{\bot\}$ and $\mathcal{C}(\mathsf{skip}, M) = \{M\}$. So while F is a refinement of* skip*, we have $\bot \in \mathcal{C}(F, M)$, while $\bot \notin \mathcal{C}(\mathsf{skip}, M)$.*

In [13] (pp. 147-149) Milner runs into a similar problem. We share the opinion that in a theory where actions without effect may be discarded, it is natural to allow an infinite number of these actions. When using weak refinement, one should realise that this does not guarantee termination of the refining schedule. However, weak refinement does preserve *partial correctness*; i.e. if the refining schedule does terminate, then the resulting state is a final state of the refined schedule.

Theorem 3.13 *Let $\langle s, M\rangle$ and $\langle t, M\rangle$ be configurations such that $\langle s, M\rangle \precapprox \langle t, M\rangle$.*
Then $(\mathcal{C}(s, M) - \{\bot\}) \subseteq \mathcal{C}(t, M)$.

Proof Let $M' \in (\mathcal{C}(s, M) - \{\bot\})$. Hence $\langle s, M\rangle \xrightarrow{\overline{\lambda}}{}^{*} \langle \mathsf{skip}, M'\rangle$ for some $\overline{\lambda}$. From $\langle s, M\rangle \precapprox \langle t, M\rangle$ follows, by induction on the length of the transition sequence, that $\langle t, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle \mathsf{skip}, M'\rangle$ where $\widehat{\lambda} = \widehat{\lambda'}$. Hence $M' \in \mathcal{C}(t, M)$. □

3.4 Stateless Refinement

In this section we develop a notion of refinement that allows an algebraic approach to refinement. To this end we define a refinement relation over *schedules* rather than over *configuration* (which was the case for statebased refinement) that is a precongruence. By dropping the multiset component from a simulation relation, we effectively require that transitions can be matched for *any* multiset [4]. Hence we refer to this type of simulation as "stateless simulation". First consider the strong variant, later we look at the weak variant.

Definition 13 *A relation $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$ is a* strong stateless simulation *if,*
for all $(s, t) \in \mathcal{R}$, for all $M \in \mathbb{M}$

i. $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle \Rightarrow \exists t' : \langle t, M\rangle \xrightarrow{\lambda} \langle t', M'\rangle$ *such that* $(s', t') \in \mathcal{R}$

ii. $s \equiv \mathsf{skip} \Rightarrow t \equiv \mathsf{skip}$

As before, we define strong stateless refinement as the largest strong stateless simulation relation. We consider a pair of schedules to be (strongly) *equivalent* if the refinement relation holds in both directions.

Definition 14

1. $\leqslant \; = \bigcup\{\mathcal{R} \mid \mathcal{R}$ *is a strong stateless simulation* $\}$
2. $\simeq \; = \; \leqslant \cap \leqslant^{-1}$

The *up-to* method for strong stateless refinement is developed in [4].

We see that using stateless simulation, s is a refinement of t, if t can match the transitions by s, independent of the multiset. This relation cannot be invalidated by some (demonic) modification of the multiset by the environment. This has the beneficial consequence that stateless refinement is a precongruence.

Theorem 3.14 *Let* s_1, s_2, t_1 *and* t_2 *be schedules such that* $s_1 \leqslant t_1$ *and* $s_2 \leqslant t_2$. *Then* $r \to s_1[t_1] \leqslant r \to s_2[t_2]$, $s_1; s_2 \leqslant t_1; t_2$, $s_1 \parallel s_2 \leqslant t_1 \parallel t_2$, $!s_1 \leqslant !t_1$ *and* $c \triangleright s_1[s_2] \leqslant c \triangleright t_1[t_2]$.

Proof In [4]. □

The notion of stateless simulation gives rise to a number of interesting refinement laws. These laws can be applied in an algebraic style of reasoning about schedules, whereas precongruence of stateless refinement enables a modular approach. We present the laws grouped per operator, starting with sequential and parallel composition. The proofs follow either from the structural congruence in Figure 1 or by determining appropriate simulations [4].

1.	$\mathsf{skip}; s \cong s$	5.	$\mathsf{skip} \parallel s \cong s$
2.	$s; \mathsf{skip} \cong s$	6.	$s_1 \parallel s_2 \cong s_2 \parallel s_1$
3.	$s_1; (s_2; s_3) \cong (s_1; s_2); s_3$	7.	$s_1 \parallel (s_2 \parallel s_3) \cong (s_1 \parallel s_2) \parallel s_3$
4.	$r \to (s_1; t)[s_2; t] \cong (r \to s_1[s_2]); t$	8.	$r \to (s_1 \parallel t)[s_2 \parallel t] \leqslant (r \to s_1[s_2]) \parallel t$

Sequential and parallel composition are related by the following law.

$$9. \quad (s_1 \parallel s_3); (s_2 \parallel s_4) \leqslant (s_1; s_2) \parallel (s_3; s_4)$$

In essence law 9 states that parallel composition may be refined by sequential composition, as exemplified by the derived property $s_1; s_2 \leqslant s_1 \parallel s_2$, which is a special case of law 9.

Finally, we can derive some basic laws for replication, among which its idempotency. Note that laws 12 and 13 below imply that $!s$ may be refined by s^k, for any $k \geq 1$.

10.	$!\mathsf{skip} \cong \mathsf{skip}$	12.	$s \leqslant !s$	14.	$!(s_1 \parallel s_2) \leqslant !s_1 \parallel !s_2$
11.	$!(!s) \cong !s$	13.	$s \parallel !s \leqslant !s$		

A number of basic laws involving the conditional operator follow easily from structural congruence. They can be found in [8] and are omitted here.

3.5 Weak Stateless Refinement

In Section 3.2 we exploited that failing rewrites (corresponding to ε-labelled transitions) are irrelevant to the outcome of a computation. We shall do the same here by extending the notion of stateless simulation to its weak variant, which is indifferent to ε-transitions.

Definition 15 *A relation $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$ is a* weak stateless simulation *if,*
for all $(s,t) \in \mathcal{R}$, for all $M \in \mathbb{M}$

i. $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle \Rightarrow \exists t' : \langle t, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t', M'\rangle$ *where* $\widehat{\lambda'} = \widehat{\lambda}$ *and* $(s',t') \in \mathcal{R}$

ii. $s \equiv \mathsf{skip} \;\Rightarrow \langle t, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle \mathsf{skip}, M\rangle$ *where* $\widehat{\lambda'} = \langle\,\rangle$

We show some basic properties of weak stateless simulation.

Lemma 3.15 *Let $\mathcal{R}_i$ for $i = 1, 2, \ldots$ be weak stateless simulations. Then the following are also weak stateless simulations*

1. *the identity relation on schedules $Id_{\mathbb{S}} = \{(s,s) \mid s \in \mathbb{S}\}$,*
2. *the composition: $\mathcal{R}_1\mathcal{R}_2$,*
3. *the union: $\bigcup_{i\in I} \mathcal{R}_i$.*

Proof Analogous to the proofs of Lemma 3.7. □

Definition 16

1. $\precsim \; = \bigcup\{\mathcal{R} \mid \mathcal{R}$ *is a weak stateless simulation* $\}$
2. $\sim \; = \; \precsim \cap \precsim^{-1}$

Lemma 3.16

1. $\precsim$ *is the largest weak stateless simulation.*
2. $\precsim$ *is a partial order.*
3. $\sim$ *is an equivalence relation.*

Proof Analogous to the proofs of Lemma 3.8. □

Analogously to the statebased refinements, it can be shown that $\precsim$ defines the relation that contains precisely all weak stateless simulations.

Definition 17 *Define a function $\mathbb{F} : \mathbb{S} \times \mathbb{S} \to \mathbb{S} \times \mathbb{S}$ as follows:*
If $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$, then $(s,t) \in \mathbb{F}(\mathcal{R})$ if and only if, for all λ, for all M

$i.\quad \langle s,M\rangle \xrightarrow{\lambda} \langle s',M'\rangle \Rightarrow \exists t' : \langle t,M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t',M'\rangle$ *where* $\widehat{\lambda'} = \widehat{\lambda}$ *and* $(s',t') \in \mathcal{R}$

$ii.\quad s \equiv \mathsf{skip} \Rightarrow \langle t,M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle \mathsf{skip},M\rangle$ *where* $\widehat{\lambda'} = \langle\,\rangle$

Lemma 3.17 $\precsim$ *is a largest fixed point of* $\mathbb{F}$.

Proof Analogous to the proof of Lemma 3.9. □

The *up-to* technique (from [13]) can be adapted for weak stateless refinement.

Definition 18 *A relation* $\mathcal{R} \subseteq \mathbb{S}\times\mathbb{S}$ *is a* weak stateless simulation up-to weak stateless refinement *if, for all* $(s,t) \in \mathcal{R}$, *for all* M,

$i.\quad \langle s,M\rangle \xrightarrow{\lambda} \langle s',M'\rangle \Rightarrow \exists t' : \langle t,M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t',M'\rangle$ *where* $\widehat{\lambda'} = \widehat{\lambda}$ *and* $s' \precsim \mathcal{R} \precsim t'$

$ii.\quad s \equiv \mathsf{skip} \Rightarrow \langle t,M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle \mathsf{skip},M\rangle$ *where* $\widehat{\lambda'} = \langle\,\rangle$

Lemma 3.18 *If* $\mathcal{R}$ *is a weak stateless simulation up-to weak stateless refinement, then* $\precsim \mathcal{R} \precsim$ *is a weak stateless simulation.*

Proof Assume $s \precsim \mathcal{R} \precsim t$. Then there are u and v such that $s \precsim u\mathcal{R}v \precsim t$.

transition
Assume $\langle s,M\rangle \xrightarrow{\lambda} \langle s',M'\rangle$.
Then by $s \precsim u$, $\langle u,M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle u',M'\rangle$ such that $\widehat{\lambda'} = \widehat{\lambda}$ and $s' \precsim u'$.
By $u\mathcal{R}v$ follows $\langle u,M\rangle \xrightarrow{\overline{\lambda''}}{}^{*} \langle v',M'\rangle$ such that $\widehat{\lambda''} = \widehat{\lambda'}$ and $u' \precsim \mathcal{R} \precsim v'$.
By $v \precsim t$ follows $\langle t,M\rangle \xrightarrow{\overline{\lambda'''}}{}^{*} \langle t',M'\rangle$ such that $\widehat{\lambda'''} = \widehat{\lambda''}$ and $u' \precsim t'$.
Hence $\widehat{\lambda'''} = \widehat{\lambda}$ and $s' \precsim\ \precsim \mathcal{R} \precsim\ \precsim t'$. By transitivity of $\precsim$: $s' \precsim \mathcal{R} \precsim t'$.

termination
If $s \equiv \mathsf{skip}$, then by $s \precsim u$, $\langle u,M\rangle \xrightarrow{\overline{\lambda}}{}^{*} \langle \mathsf{skip},M\rangle$ such that $\widehat{\lambda} = \langle\,\rangle$.
By $u\mathcal{R}v$ follows $\langle v,M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle \mathsf{skip},M\rangle$ such that $\widehat{\lambda'} = \langle\,\rangle$.
By $v \precsim t$ follows $\langle t,M\rangle \xrightarrow{\overline{\lambda''}}{}^{*} \langle \mathsf{skip},M\rangle$ such that $\widehat{\lambda''} = \langle\,\rangle$. □

Lemma 3.19 *If* $\mathcal{R}$ *is a weak stateless simulation up-to weak stateless refinement, then* $\mathcal{R} \subseteq\ \precsim$.

Proof
By Lemma 3.18, $\precsim \mathcal{R} \precsim$ is a weak stateless simulation, hence by Lemma 3.16, $\precsim \mathcal{R} \precsim\ \subseteq\ \precsim$.
From $Id_{\mathbb{S}} \subseteq\ \precsim$ follows $\mathcal{R} \subseteq\ \precsim$. □

Weak stateless refinement can be shown to be a precongruence for schedules. We show that the ordering $\precsim$ is preserved by all composition operators for schedules.

Lemma 3.20 *If s_1, s_2, t_1 and t_2 are schedules such that $s_1 \precsim t_1$ and $s_2 \precsim t_2$. Then $r \to s_1[t_1] \precsim r \to s_2[t_2]$.*

Proof
If $\langle r \to s_1[s_2], M\rangle \xrightarrow{\sigma} \langle s_1, M'\rangle$, then, by (N2) $\langle r \to t_1[t_2], M\rangle \xrightarrow{\sigma} \langle t_1, M'\rangle$.
Clearly $\langle r \to t_1[t_2], M\rangle \xrightarrow{\langle\,\sigma\,\rangle}{}^{*} \langle t_1, M'\rangle$ and $\widehat{\sigma} = \widehat{\sigma}$.

If $\langle r \to s_1[s_2], M\rangle \xrightarrow{\varepsilon} \langle s_2, M\rangle$, then, by (N1) $\langle r \to t_1[t_2], M\rangle \xrightarrow{\varepsilon} \langle t_2, M\rangle$.
Clearly $\langle r \to t_1[t_2], M\rangle \xrightarrow{\varepsilon}{}^{*} \langle t_2, M\rangle$ and $\widehat{\varepsilon} = \langle\ \rangle = \widehat{\varepsilon}$. $\square$

Lemma 3.21 *If s_1, s_2, t_1 and t_2 are schedules such that $s_1 \precsim t_1$ and $s_2 \precsim t_2$, then $s_1 \parallel s_2 \precsim t_1 \parallel t_2$.*

Proof Let $\mathcal{R} = \{(s_1 \parallel s_2, t_1 \parallel t_2) \mid s_1 \precsim t_1 \text{ and } s_2 \precsim t_2\}$.
We show that $\mathcal{R}$ is a weak stateless simulation by transition induction.
transition
There are five possible ways to derive a transition:

- By (N2) from $\langle s_1, M\rangle \xrightarrow{\lambda_1} \langle s_1', M'\rangle$.
  Then, by $s_1 \precsim t_1$, $\langle t_1, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_1', M'\rangle$ where $\widehat{\lambda} = \widehat{\lambda'}$ and $s_1' \precsim t_1'$.
  By (repeated inference using) (N2) we derive $\langle t_1 \parallel t_2, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_1' \parallel t_2, M'\rangle$.
  Clearly $(s_1' \parallel s_2, t_1' \parallel t_2) \in \mathcal{R}$.

- By (N2) from $\langle s_2, M\rangle \xrightarrow{\lambda_1} \langle s_2', M'\rangle$.
  The proof is analogous to the previous case.

- By (N3) from $\langle s_1, M\rangle \xrightarrow{\lambda} \langle s_1', M'\rangle$ and $\langle s_2, M\rangle \xrightarrow{\varepsilon} \langle s_2', M\rangle$.
  From $s_1 \precsim t_1$ follows $\langle t_1, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_1', M'\rangle$ where $\widehat{\lambda} = \widehat{\lambda'}$ and $s_1' \precsim t_1'$.
  From $s_2 \precsim t_2$ follows $\langle t_2, M\rangle \xrightarrow{\overline{\lambda''}}{}^{*} \langle t_2', M\rangle$ where $\widehat{\lambda''} = \langle\ \rangle$ and $s_2' \precsim t_2'$.
  By (repeated inference using) (N2) we derive $\langle t_1 \parallel t_2, M\rangle \xrightarrow{\overline{\lambda''}}{}^{*} \langle t_1 \parallel t_2', M'\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_1' \parallel t_2', M'\rangle$,
  Hence by transitivity of $\longrightarrow^{*}$ we get $\langle t_1 \parallel t_2, M\rangle \xrightarrow{\overline{\lambda''} \cdot \overline{\lambda'}}{}^{*} \langle t_1' \parallel t_2', M'\rangle$, where $\widehat{\lambda'' \cdot \lambda'} = \widehat{\lambda}$ and $(s_1' \parallel s_2, t_1' \parallel t_2) \in \mathcal{R}$.

- By (N3) from $\langle s_1, M\rangle \xrightarrow{\varepsilon} \langle s_1', M\rangle$ and $\langle s_2, M\rangle \xrightarrow{\lambda} \langle s_2', M'\rangle$.
  The proof is analogous to the previous case.

- By (N4) from $\langle s_1, M\rangle \xrightarrow{\sigma_1} \langle s_1', M_1'\rangle$ and $\langle s_2, M\rangle \xrightarrow{\sigma_2} \langle s_2', M_2\rangle$ where $\sigma = \sigma_1 \cdot \sigma_2$ and $M \models \sigma_1 \bowtie \sigma_2$.
  From $s_1 \precsim t_1$ follows $\langle t_1, M\rangle \xrightarrow{\overline{\lambda_1}}{}^{*} \langle t_1', M_1\rangle$ where $\widehat{\lambda_1} = \langle\,\sigma_1\,\rangle$ and $s_1' \precsim t_1'$.
  From $s_2 \precsim t_2$ follows $\langle t_2, M\rangle \xrightarrow{\overline{\lambda_2}}{}^{*} \langle t_2', M_2\rangle$ where $\widehat{\lambda_2} = \langle\,\sigma_2\,\rangle$ and $s_2' \precsim t_2'$.
  By definition of $\longrightarrow^{*}$, these transitions can also be written as

$$\langle t_1, M\rangle \xrightarrow{\varepsilon^{k1}}{}^{*} \langle t_{1,1}, M\rangle \xrightarrow{\sigma_1} \langle t_{1,2}, M_1\rangle \xrightarrow{\varepsilon^{k2}}{}^{*} \langle t_1', M_1\rangle$$

and

$$\langle t_2, M\rangle \overset{\varepsilon^{k_3}}{\longrightarrow^*} \langle t_{2,1}, M\rangle \xrightarrow{\sigma_2} \langle t_{2,2}, M_2\rangle \overset{\varepsilon^{k_4}}{\longrightarrow^*} \langle t_2', M_2\rangle$$

By ($k_1 + k_3$ times) (N2), then by (N4) (which is possible because $M \models \sigma_1 \bowtie \sigma_2$), and ($k_2 + k_4$ times) (N2), we derive

$$\langle t_1 \parallel t_2, M\rangle \overset{\varepsilon^{k_1+k_3}}{\longrightarrow^*} \langle t_{1,1} \parallel t_{2,1}, M\rangle \xrightarrow{\sigma_1 \cdot \sigma_2} \langle t_{1,2} \parallel t_{2,2}, M'\rangle \overset{\varepsilon^{k_2+k_4}}{\longrightarrow^*} \langle t_1' \parallel t_2', M'\rangle$$

Hence

$$\langle t_1 \parallel t_2, M\rangle \overset{\overline{\lambda}}{\longrightarrow^*} \langle t_1' \parallel t_2', M'\rangle$$

where $\widehat{\lambda} = \langle\, \sigma \,\rangle$. Clearly $(s_1' \parallel s_2', t_1' \parallel t_2') \in \mathcal{R}$.

termination
Then $s_1 \equiv \mathsf{skip}$ and $s_2 \equiv \mathsf{skip}$.
From $s_1 \precsim t_1$ follows $\langle t_1, M\rangle \xrightarrow{\overline{\lambda_1}} \langle \mathsf{skip}, M\rangle$ where $\widehat{\lambda_1} = \langle\,\rangle$.
From $s_2 \precsim t_2$ follows $\langle t_2, M\rangle \xrightarrow{\overline{\lambda_2}} \langle \mathsf{skip}, M\rangle$ where $\widehat{\lambda_2} = \langle\,\rangle$.
Then by (N2) we derive $\langle t_1 \parallel t_2, M\rangle \overset{\overline{\lambda_1}}{\longrightarrow^*} \langle t_2, M\rangle \overset{\overline{\lambda_2}}{\longrightarrow^*} \langle \mathsf{skip}, M\rangle$.
By transitivity of $\longrightarrow^*$ we get $\langle t_1 \parallel t_2, M\rangle \overset{\overline{\lambda_1}\cdot\overline{\lambda_1}}{\longrightarrow^*} \langle \mathsf{skip}, M\rangle$ where $\widehat{\lambda_1 \cdot \lambda_2} = \langle\,\rangle$ as required. □

Lemma 3.22 *If s_1, s_2, t_1 and t_2 are schedules such that $s_1 \precsim t_1$ and $s_2 \precsim t_2$, then $s_1; s_2 \precsim t_1; t_2$.*

Proof Let $\mathcal{R} = \{(s_1; s_2, t_1; t_2) \mid s_1 \precsim t_1 \text{ and } s_2 \precsim t_2\}$.
We show that $\mathcal{R}$ is a weak stateless simulation up-to $\sim$ by transition induction.
transition
There are two possible ways to derive a transition:

- By (N5) from $\langle s_1, M\rangle \xrightarrow{\lambda} \langle s_1', M'\rangle$.
  Then, by $s_1 \precsim t_1$, $\langle t_1, M\rangle \overset{\overline{\lambda'}}{\longrightarrow^*} \langle t_1', M'\rangle$ where $\widehat{\lambda} = \widehat{\lambda'}$ and $s_1' \precsim t_1'$.
  By (repeated inference using) (N5) we derive $\langle t_1; t_2, M\rangle \overset{\overline{\lambda'}}{\longrightarrow^*} \langle t_1'; t_2, M'\rangle$.
  By reflexivity of $\sim$ follows $(s_1'; s_2, t_1'; t_2) \in\ \sim\mathcal{R}\sim$.

- By (N9) from $s_1 \equiv \mathsf{skip}$ and $\langle s_2, M\rangle \xrightarrow{\lambda} \langle s_2', M'\rangle$.
  From $s_1 \precsim t_1$ we get $\langle t_1, M\rangle \overset{\overline{\lambda_1}}{\longrightarrow^*} \langle \mathsf{skip}, M\rangle$ where $\widehat{\lambda_1} = \langle\,\rangle$.
  From $s_2 \precsim t_2$ we get $\langle t_2, M\rangle \overset{\overline{\lambda_2}}{\longrightarrow^*} \langle t_2', M'\rangle$ where $\widehat{\lambda} = \widehat{\lambda_2}$ and $s_2' \precsim t_2'$.
  By (N5) we derive $\langle t_1; t_2, M\rangle \overset{\overline{\lambda_1}}{\longrightarrow^*} \langle t_2, M\rangle \overset{\overline{\lambda_2}}{\longrightarrow^*} \langle t_2', M'\rangle$.
  Hence by transitivity of $\longrightarrow^*$ we get $\langle t_1; t_2, M\rangle \overset{\overline{\lambda_1}\cdot\overline{\lambda_2}}{\longrightarrow^*} \langle t_2', M'\rangle$ where $\widehat{\lambda_1 \cdot \lambda_2} = \widehat{\lambda_2}$ as required.
  By $\mathsf{skip}; s_2' \sim s_2'$ and $\mathsf{skip}; t_2' \sim t_2'$ follows $(s_2', t_2') \in\ \sim\mathcal{R}\sim$.

termination
Then $s_1 \equiv \mathsf{skip}$ and $s_2 \equiv \mathsf{skip}$. The proof proceeds analogous to the *termination*-case for parallel

composition. $\square$

Lemma 3.23 *Let s_1, s_2, t_1 and t_2 be schedules such that $s_1 \precsim t_1$ and $s_2 \precsim t_2$. Then $c \triangleright s_1[s_2] \precsim c \triangleright t_1[t_2]$.*

Proof Follows straightforwardly using structural congruence by considering the cases $c = \mathit{true}$ and $c = \mathit{false}$. $\square$

Lemma 3.24 *If s and t are schedules such that $s \precsim t$ then $!s \precsim !t$.*

Proof Let $\mathcal{R} = \{(s_1 \parallel !s_2, t_1 \parallel !t_2) \mid s_1 \precsim t_1, s_2 \precsim t_2\}$.
We show that $\mathcal{R}$ is a weak stateless simulation up-to $\sim$ by induction on the depth of the inference. The result then follows from structural congruence by taking $s_1 \equiv \mathsf{skip}$ and $t_1 \equiv \mathsf{skip}$.
By Lemma 3.21 follows that $\mathcal{R}$ satisfies the following property

$$\text{If } (s,t) \in \mathcal{R} \text{ and } s' \precsim t' \text{ then } (s' \parallel s, t' \parallel t) \in \mathcal{R} \qquad (*)$$

transition
We consider the possible transitions for $(s_1 \parallel !s_2, t_1 \parallel !t_2)$.
A transition can be derived in the following ways:

1. By (N2) from $\langle s_1, M\rangle \xrightarrow{\lambda} \langle s_1', M'\rangle$.
   By $s_1 \precsim t_1$ follows $\langle t_1, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_1', M'\rangle$ where $\widehat{\lambda'} = \widehat{\lambda}$ and $s_1' \precsim t_1'$.
   By (N2) we infer $\langle t_1 \parallel !t_2, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_1' \parallel !t_2, M'\rangle$. By reflexivity of $\sim$ follows $(s_1' \parallel !s_2, t_1' \parallel !t_2) \in {\sim}\mathcal{R}{\sim}$.

2. By (N2) from $\langle !s_2, M\rangle \xrightarrow{\lambda} \langle s_2', M'\rangle$.
   This transition can in turn be derived in the following ways:

   (a) By (N6) from $\langle s_2, M\rangle \xrightarrow{\lambda} \langle s_2', M'\rangle$.
   From $s_2 \precsim t_2$ follows $\langle t_2, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_2', M'\rangle$ where $\widehat{\lambda} = \widehat{\lambda'}$ and $s_2' \precsim t_2'$.
   By (N6) we get $\langle !t_2, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_2', M'\rangle$.
   By (N2) we get $\langle t_1 \parallel !t_2, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_1 \parallel t_2', M'\rangle$.
   From Lemma 3.21 follows that $s_1 \parallel s_2' \precsim t_1 \parallel t_2'$. By structural congruence $s \equiv s \parallel !\mathsf{skip}$, hence $(s_1 \parallel s_2', t_1 \parallel t_2') \in {\sim}\mathcal{R}{\sim}$.

   (b) By (N7) from $\langle s_2 \parallel !s_2, M\rangle \xrightarrow{\lambda} \langle s_2', M'\rangle$.
   By induction we have $\langle t_2 \parallel !t_2, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_2', M'\rangle$ where $\widehat{\lambda} = \widehat{\lambda'}$ and $s_2' \sim\mathcal{R}\sim t_2'$.
   By (N7) we get $\langle !t_2, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_2', M'\rangle$.
   By (N2) we get $\langle t_1 \parallel !t_2, M\rangle \xrightarrow{\overline{\lambda'}}{}^{*} \langle t_1 \parallel t_2', M'\rangle$.

Since $(s'_2, t'_2) \in \mathcal{R}$ we have by $(*)$ that $(s_1 \parallel s'_2, t_1 \parallel t'_2) \in \mathcal{R}$. By reflexivity of $\sim$ follows $(s_1 \parallel s'_2, t_1 \parallel t'_2) \in\ \sim\mathcal{R}\sim$.

3. By (N3) from $\langle s_1, M\rangle \xrightarrow{\lambda} \langle s'_1, M'\rangle$ and $\langle !s_2, M\rangle \xrightarrow{\varepsilon} \langle s'_2, M'\rangle$.
   The proof is a routine combination of cases 1. and 2. analogous to the corresponding case (derivation by (N3)) in the proof of precongruence of parallel composition.
4. By (N3) from $\langle s_1, M\rangle \xrightarrow{\varepsilon} \langle s'_1, M'\rangle$ and $\langle !s_2, M\rangle \xrightarrow{\lambda} \langle s'_2, M'\rangle$.
   The remainder of this proof is analogous to the previous case.
5. By (N4) from $\langle s_1, M\rangle \xrightarrow{\sigma_1} \langle s'_1, M_1\rangle$ and $\langle !s_2, M\rangle \xrightarrow{\sigma_2} \langle s'_2, M_2\rangle$ where $\sigma = \sigma_1 {\cdot} \sigma_2$ and $M \models \sigma_1 \bowtie \sigma_2$.
   The proof is a routine combination of cases 1. and 2. analogous to the corresponding case (derivation by (N5)) in the proof of precongruence of parallel composition.

termination
Then $s_1 \equiv \mathsf{skip}$ and $s_2 \equiv \mathsf{skip}$. The proof proceeds analogously to the *termination* proof for parallel composition. $\square$

Lemma 3.25 *Weak stateless refinement is preserved by recursive definitions.*

Proof Analogous to the recursion proof for strong stateless refinement in [4]. $\square$

Theorem 3.26 *Weak stateless refinement is a precongruence for schedules.*

Proof From Lemma's 3.20, 3.21, 3.22, 3.23 3.24 and 3.25. $\square$

The essential difference between weak and strong stateless refinement can be expressed as an algebraic law. In this law we use the rewrite rule *fail* that represents the class of rules that, when executed, can only make a failing transition (denoted by ε). We can think of it as being defined as: $\mathit{fail} \,\widehat{=}\, \overline{x} \mapsto m \Leftarrow \mathit{false}$. We then arrive at the following weak stateless law, which enables the elimination of *fail* from schedules.

$$15.\quad t \quad \sim \quad \mathit{fail} \to s[t]$$

Lemma 3.27 $t \sim \mathit{fail} \to s[t]$

Proof

- $t \precsim \mathit{fail} \to s[t]$: We show that $\mathcal{R} = \{(t, \mathit{fail} \to s[t])\} \cup \mathit{Id}_{\mathbb{S}}$ is a weak stateless simulation.
  transition
  If $\langle t, M\rangle \xrightarrow{\lambda} \langle t', M'\rangle$, then, by $(N0)$, $\langle \mathit{fail} \to s[t], M\rangle \xrightarrow{\varepsilon} \langle t, M\rangle$.
  By transitivity of $\longrightarrow$: $\langle \mathit{fail} \to s[t], M\rangle \overset{\varepsilon\cdot\lambda}{\longrightarrow}{}^{*} \langle t', M'\rangle$.
  Clearly $\widehat{\varepsilon \cdot \lambda} = \widehat{\lambda}$ and $(t', t') \in \mathit{Id}_{\mathbb{S}} \subseteq \mathcal{R}$.

termination
If $t \equiv \mathsf{skip}$, then by $(N0)$, $\langle \mathit{fail} \to s[t], M\rangle \xrightarrow{\varepsilon} \langle t, M\rangle$. Clearly $\widehat{\varepsilon} = \langle\ \rangle$.

- $\mathit{fail} \to s[t] \precsim t$:
  We show that $\mathcal{R} = \{(\mathit{fail} \to s[t], t)\} \cup \mathit{Id}_{\mathbb{S}}$ is a weak stateless simulation.
  transition
  By $(N0)$, $\langle \mathit{fail} \to s[t], M\rangle \xrightarrow{\varepsilon} \langle t, M\rangle$. By reflexivity of $\longrightarrow^*$: $\langle t, M\rangle \overset{\langle\ \rangle}{\longrightarrow}{}^* \langle t, M\rangle$.
  Clearly $(t, t) \in \mathit{Id}_{\mathbb{S}} \subseteq \mathcal{R}$.
  termination
  There are no s and t such that $\mathit{fail} \to s[t] \equiv \mathsf{skip}$, hence this case holds vacuously.

$\square$

Taking both $s \equiv \mathsf{skip}$ and $t \equiv \mathsf{skip}$, we get $\mathsf{skip} \sim \mathit{fail}$ as a special case of Law 15. These laws may seem futile because no sensible program or schedule uses *fail*. However, consider the following law that relates the schedule conditional $c \triangleright \_[\_]$ to the reaction condition b of a rewrite rule $\overline{x} \mapsto m \Leftarrow b$. Law 16 is a strong stateless equivalence that introduces *fail*.

$$16. \quad r \to s[t] \quad \simeq \quad c \triangleright (r \to s[t])[\mathit{fail}; t] \quad \text{if } b \Rightarrow c$$

Lemma 3.28 *If $b \Rightarrow c$ then $r \to s[t] \simeq c \triangleright (r \to s[t])[\mathit{fail}; t]$.*

Proof If $c = \mathit{true}$, then by structural congruence $c \triangleright (r \to s[t])[\mathit{fail}; t] \equiv r \to s[t]$, and the result follows by reflexivity of $\simeq$. It remains to consider the case $c = \mathit{false}$. From $b \Rightarrow c$ follows $\neg b$. From structural congruence follows $c \triangleright (r \to s[t])[\mathit{fail}; t] \equiv \mathit{fail}; t$.

- We show that $\mathcal{R} = \{(r \to s[t], \mathit{fail}; t) \mid \neg b\} \cup \mathit{Id}_{\mathbb{S}}$ is a strong stateless simulation.
  By reflexivity of $\leqslant$ we know that $\mathit{Id}_{\mathbb{S}}$ is a strong stateless simulation. We concentrate on the remaining terms. We consider the possible transitions.
  transition
  From $\neg b$, we get by $(N0)$, $\langle r \to s[t], M\rangle \xrightarrow{\varepsilon} \langle t, M\rangle$.
  By $(N0)$ and $(N5)$, $\langle \mathit{fail}; t, M\rangle \xrightarrow{\varepsilon} \langle t, M\rangle$, and $(t, t) \in \mathit{Id}_{\mathbb{S}} \subseteq \mathcal{R}$.
  termination
  There are no s and t such that $r \to s[t] \equiv \mathsf{skip}$, hence this case holds vacuously.

- We show that $\mathcal{R} = \{(\mathit{fail}; t, r \to s[t]) \mid \neg b\} \cup \mathit{Id}_{\mathbb{S}}$ is a strong stateless simulation.
  By reflexivity of $\leqslant$ we know that $\mathit{Id}_{\mathbb{S}}$ is a strong stateless simulation. We concentrate on the remaining terms. We consider the possible transitions.
  transition
  By $(N0)$, $\langle \mathit{fail}; t, M\rangle \xrightarrow{\varepsilon} \langle t, M\rangle$.
  Because $\neg b$, we get by $(N0)$, $\langle r \to s[t], M\rangle \xrightarrow{\varepsilon} \langle t, M\rangle$, and $(t, t) \in \mathit{Id}_{\mathbb{S}} \subseteq \mathcal{R}$.
  termination
  There are no s and t such that $\mathit{fail}; t \equiv \mathsf{skip}$, hence this case holds vacuously.

$\square$

In the next section, we formally prove how the use of the notions of refinement that we have seen can be combined. In particular we may combine strong and weak stateless laws to derive weak stateless refinements. For example, using the weak stateless law 15 and the strong stateless law 16 we derive the following weak stateless equivalence

$$17. \quad r \to s[t] \quad \sim \quad c \rhd (r \to s[t])[t] \quad \text{if } b \Rightarrow c$$

3.6 Relating the notions of refinement

In the previous sections we presented some different notions of refinement. In this section we show how they are related and how their use can be combined into a hybrid proof method.

Our first result is that strong refinement is contained in weak refinement, for both statebased and stateless refinement.

Lemma 3.29

1. $\langle s, M\rangle \leqq \langle t, M\rangle \Rightarrow \langle s, M\rangle \precapprox \langle t, M\rangle$ *for all* M *(or* $\leqq \subseteq \precapprox$ *)*
2. $s \leqslant t \Rightarrow s \precsim t$ *(or* $\leqslant \subseteq \precsim$ *)*

Proof Immediately from the definitions of strong and weak refinement, and the fact that $\longrightarrow \subseteq \longrightarrow^*$. $\square$

Secondly, stateless refinement is contained in statebased refinement, for both their strong and weak versions.

Lemma 3.30

1. $s \leqslant t \Rightarrow \langle s, M\rangle \leqq \langle t, M\rangle$ *for all* M
2. $s \precsim t \Rightarrow \langle s, M\rangle \precapprox \langle t, M\rangle$ *for all* M

Proof

1. Let $\mathcal{R} = \{(\langle s, M\rangle, \langle t, M\rangle) \mid s \leqslant t, M \in \mathbb{M}\}$.
   The result follows by showing that $\mathcal{R}$ is a strong statebased simulation.
   transition
   Consider an arbitrary transition for s: $\langle s, M\rangle \xrightarrow{\lambda} \langle s', M'\rangle$.
   From $s \leqslant t$ follows $\langle t, M\rangle \xrightarrow{\lambda} \langle t', M'\rangle$ such that $s' \leqslant t'$, hence $(\langle s', M'\rangle, \langle t', M'\rangle) \in \mathcal{R}$.
   termination
   If $s \equiv \mathsf{skip}$, then from $s \leqslant t$ follows $t \equiv \mathsf{skip}$.

2. The proof is analogous to that of part 1.

$\square$

Law 15 implies that strong stateless refinement is strictly contained in weak stateless refinement. From Lemma 3.30, then follows that strong statebased refinement is strictly smaller than weak statebased refinement. Analogous to Example 3.2 we can prove $\langle \mathit{Sum}(n); \mathit{fail}, M\rangle \leqq \langle \Gamma_{add}, M\rangle$ for $\#M = n$, and by Lemma 3.29 $\langle \mathit{Sum}(n), M\rangle \precapprox \langle \Gamma_{add}, M\rangle$ for $\#M = n$. But $\mathit{Sum}(n); \mathit{fail}$ is not a strong stateless refinement of Γ_{add}, and $\mathit{Sum}(n)$ is not a weak stateless refinement of Γ_{add}. This proves that strong- and weak stateless are strictly contained in their statebased counterparts.

The inclusion relations between the various notions of refinement are schematically depicted in Figure 3. It shows that weak statebased refinement is the most liberal notion and strong stateless is the strictest. From Lemmas 3.29 and 3.30 follows that the refinements contained in the stricter notions are also contained in the more liberal ones. Deriving refinements will be an interplay between the methods presented. Typically, the first refinements involve the introduction of ordering on the selection of data. This relates the progress of a schedule to the contents of the multiset, which clearly is a statebased property, hence proving this requires the use of one of the statebased simulation methods. Subsequently, the ordering of actions may be refined using stateless laws. Through precongruence of the stateless notions, these refinements may be obtained in a modular way using the algebraic laws. The containment within statebased methods ensures that these refinement are also valid there.

$$\begin{array}{ccc}
\text{strong stateless} & \subset & \text{weak stateless} \\
\cap & & \cap \\
\text{strong statebased} & \subset & \text{weak statebased}
\end{array}$$

Figure 3: Relation between the notions of refinement.

These results make it possible to use hybrid *up-to* methods. For instance, we may use the strong stateless refinement laws of Section 3.4 to show that some relation $\mathcal{R}$ is a weak statebased simulation as follows:

Suppose that, given $\langle s', M\rangle \mathcal{R} \langle t', M\rangle$, we want to show that $\langle s, M\rangle \precapprox \mathcal{R} \precapprox \langle t, M\rangle$. Now if we can prove using the strong stateless laws that $s \leqslant s'$ and $t' \leqslant t$, then by Lemma 3.29(2) follows $s \precsim s'$ and $t' \precsim t$. And by Lemma 3.30(2) follows $\langle s, M\rangle \precsim \langle s', M\rangle$ and $\langle t', M\rangle \precsim \langle t, M\rangle$. Hence $\langle s, M\rangle \precapprox \mathcal{R} \precapprox \langle t, M\rangle$.

4 Applications

In this section we will illustrate the method of development with two examples: summation and the single source shortest paths problem.

4.1 Application 1: Summation

We return to the summation program *add* from Example 3.1. As in Example 3.1, we may exploit the size n of the given multiset to perform exactly $n-1$ additions. However, unlike there, we will not impose any (sequential) order on the computation. The schedule we consider here is simply add^{n-1}. Hence we are interested in the refinement $\langle add^{n-1}, M\rangle \precapprox \langle \Gamma_{add}, M\rangle$ with $\#M = n$.

Lemma 4.1 *$\langle add^{n-1}, M\rangle \precapprox \langle \Gamma_{add}, M\rangle$ with $\#M = n$.*

Proof Let $\mathcal{R} = \{(\langle add^{n-1}, M\rangle, \langle \Gamma^k_{add}, M\rangle) \mid \#M = n, n \geq 0, k \geq 1\}$. Showing that $\mathcal{R}$ is a weak statebased simulation proceeds analogously to the proof of Example 3.2. □

We can further refine the control structure by algebraic reasoning using the stateless refinement laws. For instance, a schedule that performs recursive doubling can be defined as follows.

$$RecDubSum(i) \mathrel{\widehat{=}} (i > 1) \triangleright (add^{\lfloor i/2 \rfloor}; RecDubSum(\lceil i/2 \rceil))$$

Lemma 4.2 *For all n, $RecDubSum(n) \leqslant add^{n-1}$*

Proof By induction on n.

- $n \leq 1$: $RecDubSum(n) \simeq \mathsf{skip} \simeq add^{n-1}$ as required.
- $n > 1$:

$$\begin{array}{ll}
 & RecDubSum(n) \\
\simeq & \qquad\qquad\qquad\qquad \text{def.}RecDubSum \\
 & add^{\lfloor n/2 \rfloor}; RecDubSum(\lceil n/2 \rceil) \\
\leqslant & \qquad\qquad\qquad\qquad \text{Induction} \\
 & add^{\lfloor n/2 \rfloor}; add^{\lceil n/2 \rceil - 1} \\
\leqslant & \qquad\qquad\qquad\qquad \text{law 9} \\
 & add^{n-1}
\end{array}$$

□

In its turn, we may refine $RecDubSum(n)$ by $Sum(n)$ where $Sum(n) \mathrel{\widehat{=}} n > 1 \triangleright (add; Sum(n-1))$ is the sequential schedule for summation which we saw earlier in Examples 3.1 and 3.2.

Lemma 4.3 *For all n, $Sum(n) \leqslant RecDubSum(n)$*

Proof By induction on n.

- $n \leq 1$: $Sum(n) \simeq \mathsf{skip} \simeq RecDubSum(n)$

- $n > 1$: $Sum(n)$

$$\begin{array}{ll}
 & Sum(n) \\
\simeq & \hfill \text{def. } Sum, \lfloor n/2 \rfloor \geq 1 \\
 & \underbrace{add; \ldots; add}_{\lfloor n/2 \rfloor}; Sum(\lceil n/2 \rceil) \\
\leqslant & \hfill \text{law 9} \\
 & add^{\lfloor n/2 \rfloor}; Sum(\lceil n/2 \rceil) \\
\leqslant & \hfill \text{Induction} \\
 & add^{\lfloor n/2 \rfloor}; RecDubSum(\lceil n/2 \rceil) \\
\simeq & \hfill \text{def. } RecDubSum \\
 & RecDubSum(n)
\end{array}$$

□

Hence we arrive at the refinement ordering (for $\#M = n$):

$$\langle Sum(n), M \rangle \precapprox \langle RecDubSum(n), M \rangle \precapprox \langle add^{n-1}, M \rangle \precapprox \langle \Gamma_{add}, M \rangle$$

Figure 4 depicts the method by which we derived these refinement.

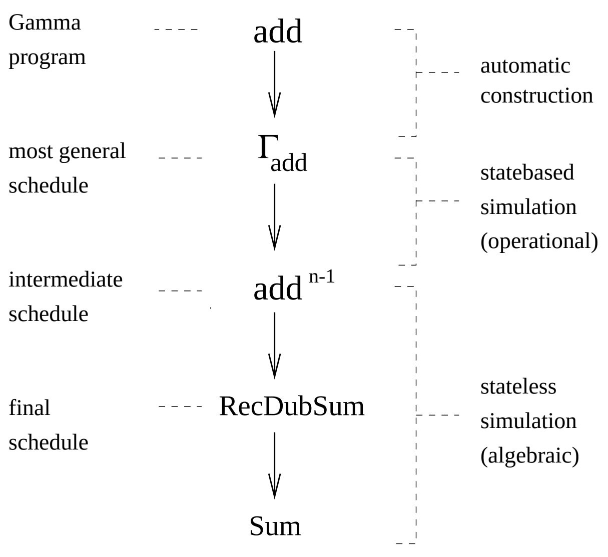

Figure 4: Lattice of Refinements for Summation

Even though the summation program is fairly simple, the results we have proven here provided useful in the more complex case of solving triangular systems of linear equations [5].

4.2 Application 2: Single Source Shortest Paths

We demonstrate application of the refinement techniques by considering an example problem that is commonly known as the single source shortest paths problem. Pursuing separation between computation and coordination we shall first specify the basic computation that is required to solve the problem in Gamma. After that we shall relate several coordination strategies.

The problem description is as follows. Assume we are given a directed graph with vertices $V = \{1, \ldots, n\}$. A function L associates with every edge (u, v) a non-negative length $L(u, v)$. If there is no edge between vertices u and v, then we take $L(u, v) = \infty$. Also $L(u, u) = 0$ for all vertices u. Given a source vertex s, the problem is to determine for every vertex v, the length of a shortest path from s to v.

A Gamma program for solving this problem is given by the rule:

$$\mathit{find}(u, v) \mathrel{\widehat{=}} (u, x), (v, y) \mapsto (u, x), (v, x + L(u, v)) \Leftarrow x + L(u, v) < y$$

The multiset consists of pairs (v, x), where v is a vertex number and x is the length of a path from the source s to v. The initial multiset is given by: $M_0 = \{(s, 0)\} \cup \{\ (v, \infty)\ \mid\ 1 \leq v \leq n, v \neq s\}$.

4.2.1 A First Refinement

Though the program performs the required computation, as can be proven formally using techniques from [1], it is hopelessly inefficient because of its unstructured search through the graph. We may coordinate the program's activities into a coherent (more deterministic) searching strategy by conducting a directed search on the graph starting from the source. From a given vertex u the search proceeds by an attempt to construct a shorter path to every adjacent vertex v (in no preferred order). If the attempt succeeds, the search is continued; otherwise the search at v is aborted. A schedule that expresses this strategy is given by $\mathit{Search}(s)$, where

$$\mathit{Search}(u) \quad \widehat{=} \quad (\Pi_{v=1}^{n} \mathit{find}(u, v) \rightarrow \mathit{Search}(v))$$

Note that the schedule still exhibits highly nondeterministic behaviour. The paths in the graph are traversed in any order (possibly in parallel). Using the refinement techniques, however, we can transform the schedule into more deterministic versions [4].

As an illustration of the method, we shall prove that *Search* is a correct refinement of Γ_{find}. We start by giving a general schedule, denoted GS, that allows us to describe any form that the schedule *Search* may evolve into during execution. Let F be a multiset over $V \times V$, define

$$GS(F) \mathrel{\widehat{=}} (\Pi_{(u_1, u_2) \in F} \mathit{find}(u_1, u_2) \rightarrow \mathit{Search}(u_2))$$

The next results prepare the ground for proving a simulation relation which establishes our ultimate goal: showing that there is some a weak statebased simulation relation $\mathcal{R}$ such that $(\langle \mathit{Search}(s), M\rangle, \langle \Gamma_{\mathit{find}}, M\rangle) \in \mathcal{R}$.

Let $\#i$ denote the number of tuples $(v, x) \in M$ such that [1] $v = i$. Define

$$\mathcal{R} \quad = \quad \{(\langle GS(F), M\rangle, \langle \Gamma^k_{\mathit{find}}, M\rangle) \mid \mathit{Inv1}(M) \wedge \mathit{Inv2}(M, F), k \geq 1\}$$

where

$$\mathit{Inv1}(M) \quad = \quad \forall i : 1 \leq i \leq n : \#i = 1$$

and

$$\mathit{Inv2}(M, F) \quad = \quad \forall (v_1, x_1), (v_2, x_2) \in M : (v_1, v_2) \notin F : x_2 \leq x_1 + L(v_1, v_2)$$

In the next two lemmas we consider the possible transitions that are derived from the execution of a single rewrite rule *find*(u, v) and show that these preserve invariants $Inv1$ and $Inv2$ as well as the syntactical form GS.

Lemma 4.4 *Let $\langle GS(F), M\rangle$ be a configuration such that $Inv1(M)$ and $Inv2(M, F)$. If $\langle GS(F), M\rangle \xrightarrow{\varepsilon} \langle s', M'\rangle$ is a single step transition, then*

1. *$s' \simeq GS(F')$ where $F' = F \ominus \{(u, v)\}$ for some $(u, v) \in F$*
2. *$Inv1(M')$*
3. *$Inv2(M', F')$*

Proof

1. By $(N0)$ and $(N2)$ we derive $s' \equiv GS(F')$ where $F' = F \ominus \{(u, v)\}$.
   By Lemma 4.7 from [4] follows $s' \simeq GS(F')$.
2. Because $M' = M$, clearly $Inv1(M')$.
3. We have $(u, x), (v, y) \in M$ and $x + L(u, v) \geq y$. Consequently, using $\mathit{Inv2}(M, F)$:

   $$\forall (v_1, x_1), (v_2, x_2) \in M : (v_1, v_2) \notin (F \ominus \{(u, v)\}) : x_2 \leq x_1 + L(v_1, v_2)$$

   Because $F' = F \ominus \{(u, v)\}$ this establishes $\mathit{Inv2}(M, F')$.

□

Lemma 4.5 *Let $\langle GS(F), M\rangle$ be a configuration such that $Inv1(M)$ and $Inv2(M, F)$. If $\langle GS(F), M\rangle \xrightarrow{\sigma} \langle s', M'\rangle$ is a single step transition, then*

1. *$s' \simeq GS(F')$ where $F' = F \ominus \{(u, v)\} \cup \{(v, w) \mid w \in V\}$*
2. *$Inv1(M')$*

[1]Because M is formally a function, we can define $\#i = \Sigma_{x \in \mathbb{N} \cup \{\infty\}} M(i, x)$.

3. $Inv2(M', F')$

Proof Because σ is the label of a single step transition, $\sigma = \{(v, y')\}/\{(v, y)\}$ for some $(v, y), (u, x) \in M$ such that $y' = x + L(u, v)$, $y' < y$ and $(u, v) \in F$.

1. By $(N1)$ and $(N2)$ we derive $s' \equiv GS(F \ominus \{(u, v)\}) \,\|\, \mathit{Search}(v)$. From $\mathit{Search}(v) \simeq GS(\{(v, w) \mid w \in V\})$ follows $s' \simeq GS(F')$ where $F' = F \ominus \{(u, v)\} \cup \{(v, w) \mid w \in V\}$.
2. From *Inv1*(M) and $M' = M \ominus \{(v, y)\}) \cup \{(v, x + L(u, v))\}$ it is clear that *Inv1*(M') holds.
3. We start by deriving some auxiliary results.
   - As a special case of *Inv2*(M, F) we have

     $$\forall (v_1, x_1), (v_2, x_2) \in M : \quad (v_1, v_2) \notin F \wedge (v_2, x_2) = (v, y) : \quad y \leq x_1 + L(v_1, v) \quad (1)$$

     By definition of M' and $y' < y$ we have, by *Inv1*(M'),

     $$\forall (v_1, x_1), (v_2, x_2) \in M' : \quad (v_1, v_2) \notin F \wedge (v_2, x_2) = (v, y') : \quad y' \leq x_1 + L(v_1, v) \quad (2)$$

   - By definition of M' and *Inv1*(M') follows

     $$\forall (v_1, x_1), (v_2, x_2) \in M' : \quad v_1 = u \wedge v_2 = v : \quad x_2 = x_1 + L(v_1, v_2) \quad (3)$$

   By definition of M' and *Inv2*(M, F) follows

$$\begin{array}{ll}
 & \forall (v_1, x_1), (v_2, x_2) \in M' : (v_1, v_2) \notin F \wedge v_1 \neq v \wedge v_2 \neq v : x_2 \leq x_1 + L(v_1, v_2) \\
\Rightarrow & \hfill \{ \text{ by } (2) \} \\
 & \forall (v_1, x_1), (v_2, x_2) \in M' : (v_1, v_2) \notin F \wedge v_1 \neq v : x_2 \leq x_1 + L(v_1, v_2) \\
\Rightarrow & \hfill \{ \text{ set calculus } \} \\
 & \forall (v_1, x_1), (v_2, x_2) \in M' : (v_1, v_2) \notin (F \cup \{(v, w) \mid w \in V\}) : x_2 \leq x_1 + L(v_1, v_2) \\
\Rightarrow & \hfill \{ F' \cup \{(u, v)\} = F \cup \{(v, w) \mid w \in V\} \} \\
 & \forall (v_1, x_1), (v_2, x_2) \in M' : (v_1, v_2) \notin (F' \cup \{(u, v)\}) : x_2 \leq x_1 + L(v_1, v_2) \\
\Rightarrow & \hfill \{ \text{ by } (3) \text{ and } \mathit{Inv1}(M') \} \\
 & \forall (v_1, x_1), (v_2, x_2) \in M' : (v_1, v_2) \notin F' : x_2 \leq x_1 + L(v_1, v_2)
\end{array}$$

   Which proves *Inv2*(M', F').

□

Lemma 4.6 *If* $\langle GS(F), M \rangle \xrightarrow{\lambda} \langle s', M' \rangle$*, then* $\langle \Gamma_{\mathit{find}}, M \rangle \xrightarrow{\lambda} \langle \Gamma^k_{\mathit{find}}, M' \rangle$ *for some* $k \geq 1$

Proof By induction on $\#F$. Note that the proposition holds vacuously if $F = \emptyset$.
Consider the case $\#F = 1$; i.e. $F = \{(u, v)\}$. Then $GS(F) \mathrel{\widehat{=}} \mathit{find}(u, v) \to \Gamma_{\mathit{find}}$.
A transition $\langle GF(F), M \rangle \xrightarrow{\lambda} \langle s', M' \rangle$ can be derived by $(N0)$ or by $(N1)$:

- $(N0)$: $\lambda = \varepsilon$: hence $\langle \mathit{find}(u,v) \to \Gamma_{\mathit{find}}, M\rangle \xrightarrow{\varepsilon} \langle \mathsf{skip}, M\rangle$.
  By (N2) $\langle \mathit{find}(u,v) \to \Gamma_{\mathit{find}} \,\|\, \Gamma_{\mathit{find}}, M\rangle \xrightarrow{\varepsilon} \langle \Gamma_{\mathit{find}}, M\rangle$.
  By (N7) $\langle \Gamma_{\mathit{find}}, M\rangle \xrightarrow{\varepsilon} \langle \Gamma_{\mathit{find}}, M\rangle$.

- $(N1)$: then $\lambda = \sigma$ and $s' \equiv \mathit{Search}(v)$. $\langle \mathit{find}(u,v) \to \Gamma_{\mathit{find}}, M\rangle \xrightarrow{\sigma} \langle \Gamma_{\mathit{find}}, M'\rangle$.
  By (N6) and (N8) $\langle \Gamma_{\mathit{find}}, M\rangle \xrightarrow{\sigma} \langle \Gamma_{\mathit{find}}, M'\rangle$.

Next, we consider $\#F \geq 2$. By commutativity and associativity of $\|$ follows that $GS(F)$ can be written as $GS(F) \equiv GS(F_1) \,\|\, GS(F_2)$ where $F_1 = \{(u,v)\}$ and $F_2 = F \ominus \{(u,v)\}$. Now, consider the possible derivations of a transition $\langle GS(F_1) \,\|\, GS(F_2), M\rangle \xrightarrow{\lambda} \langle s', M'\rangle$:

- By $(N2)$ or $(N3)$: If the last derivation is by $(N2)$ or $(N3)$, then the result follows immediately from the induction hypothesis.

- By $(N4)$ from $\langle GS(F_1), M\rangle \xrightarrow{\sigma_1} \langle s_1, M_1\rangle$ and $\langle GS(F_2), M\rangle \xrightarrow{\sigma_2} \langle s_2, M_2\rangle$ where $\sigma = \sigma_1 \cdot \sigma_2$ and $M \models \sigma_1 \bowtie \sigma_2$. By the induction hypothesis follows $\langle \Gamma_{\mathit{find}}, M\rangle \xrightarrow{\sigma_2} \langle \Gamma^k_{\mathit{find}}, M_2\rangle$ for some $k \geq 1$.
  From the former transition we get by $(N8)$ and $(N1)$ that $\langle \mathit{find} \to \Gamma_{\mathit{find}}, M\rangle \xrightarrow{\sigma_1} \langle \Gamma_{\mathit{find}}, M_1\rangle$.
  Then by $(N4)$ follows $\langle (\mathit{find} \to \Gamma_{\mathit{find}}) \,\|\, \Gamma_{\mathit{find}}, M\rangle \xrightarrow{\sigma} \langle \Gamma^{k+1}_{\mathit{find}}, M'\rangle$.
  And by $(N7)$ we conclude $\langle \Gamma_{\mathit{find}}, M\rangle \xrightarrow{\sigma} \langle \Gamma^{k+1}_{\mathit{find}}, M'\rangle$ such that $k+1 \geq 1$.

□

Lemma 4.7 $\mathcal{R} = \{(\langle GS(F), M\rangle, \langle \Gamma^k_{\mathit{find}}, M\rangle) \mid \mathit{Inv1}(M) \wedge \mathit{Inv2}(M,F), k \geq 1\}$ *is a weak statebased simulation up-to weak statebased refinement.*

Proof
transition
If $\langle GS(F), M\rangle \xrightarrow{\lambda} \langle s', M'\rangle$, then by Lemma 2.1 there exist $\lambda_1, \ldots, \lambda_n, n \geq 1$ such that

$$\langle s_0, M_0\rangle \xrightarrow{\lambda_1} \langle s_1, M_1\rangle \ldots \xrightarrow{\lambda_i} \ldots \langle s_{n-1}, M_{n-1}\rangle \xrightarrow{\lambda_n} \langle s_n, M_n\rangle$$

where $\langle GS(F), M\rangle = \langle s_0, M_0\rangle$, $\langle s_n, M_n\rangle = \langle s', M'\rangle$ and each transition is single-step. By induction on the length of the transition sequence it follows by Lemma 4.4 and Lemma 4.5 that $s' \simeq GS(F')$ and $Inv1(M')$ and $Inv2(M', F')$.
By Lemma 4.6 follows $\langle \Gamma_{\mathit{find}}, M\rangle \xrightarrow{\lambda} \langle \Gamma^{k'}_{\mathit{find}}, M'\rangle$ for some $k' \geq 1$.
By $(N2)$, $\langle \Gamma^k_{\mathit{find}}, M\rangle \xrightarrow{\lambda} \langle \Gamma^{k-1+k'}_{\mathit{find}}, M'\rangle$ with $k - 1 + k' \geq 1$.
Hence $\langle s', M'\rangle \precsim \mathcal{R} \precsim \langle \Gamma^{k-1+k'}_{\mathit{find}}, M'\rangle$.
termination
$GS(F, W) \equiv \mathsf{skip}$ only if $F = \emptyset$. By $\mathit{Inv2}(M, \emptyset)$ then follows that for all $(v_1, x_1), (v_2, x_2) \in M$: $x_2 \leq x_1 + L(v_1, v_2)$. Hence by $(N0)$ $\langle \mathit{find} \to \Gamma_{\mathit{find}}, M\rangle \xrightarrow{\varepsilon} \langle \mathsf{skip}, M\rangle$.
By $(N6)$, $(N9)$ and $(N2)$ and the definition of $\longrightarrow^*$, $\langle \Gamma^k_{\mathit{find}}, M\rangle \overset{\varepsilon}{\longrightarrow^*} \langle \mathsf{skip}, M\rangle$.
□

To prove $\langle \mathit{Search}(s), M_0\rangle \precapprox \langle \Gamma_{\mathit{find}}, M_0\rangle$ it remains to show that $(\langle \mathit{Search}(s), M_0\rangle, \langle \Gamma_{\mathit{find}}, M_0\rangle) \in \mathcal{R}$. Clearly $\mathit{Inv1}(M_0)$ holds. To verify $\mathit{Inv2}(M_0, \{(s,v) \mid v \in V\})$ we need to check that $\forall (v_1, x_1), (v_2, x_2) \in M_0 : v_2 \notin \{s\} : x_2 \leq x_1 + L(v_1, v_2)$. This follows from the initialization of all vertices $v_i \in V - \{s\}$ by (v_i, ∞) because it entails $\infty + L(v_i, v_j) \leq \infty$.

The most laborious part of proving statebased refinements consists of showing that the invariants are preserved by all possible transitions of the schedule. If a schedule consists of the parallel composition of k components, then the multistep semantics may give rise to an exponential number, $2^k - 1$, of possible transitions. However, Lemma 2.1 allows us to consider only the single-step transitions (as illustrated by Lemma 4.4 and Lemma 4.5). This effectively reduces reasoning about the parallel behaviour of schedules to reasoning about their interleaved behaviour.

A further reduction of the effort of proving statebased refinements can be obtained by generalizing Lemma 4.6. We conjecture that the most general schedule has the following more general property (which entails Lemma 4.6): the most general schedule can mimic any transition by a schedule that is built from strengthenings of rules that the most general schedule is built from. This reduces the proof obligation of Lemmas analogous to 4.6 to checking that a schedule is built from strengthenings of the rules of the most general schedule. The formal proof of this property will be the subject of future research.

In the next section we will illustrate how a further refinement of the shortest paths schedule *Search* can be proven by algebraic reasoning using results from Section 3.5.

4.2.2 Some Further Refinements

Further refinements can be derived using the algebraic laws from Sections 3.4 and 3.5. Because a shorter path to v via u can be found only if there is an edge between u and v, we only need to look for shorter paths from u to neighbours of u; i.e. the vertices v such that $L(u,v) < \infty$. Formally, this depends on the following property:

$$x + L(u,v) < y \Rightarrow L(u,v) < \infty$$

By law 17 from Section 3.5 then follows

$$(\Pi_{v=1}^{n} L(u,v) < \infty \triangleright \mathit{find}(u,v) \rightarrow \ \mathit{Search}(v)) \sim (\Pi_{v=1}^{n} \mathit{find}(u,v) \rightarrow \ \mathit{Search}(v))$$

hence $\mathit{Search}'(s) \sim \mathit{Search}(s)$ where

$$\mathit{Search}'(u) \mathrel{\widehat{=}} (\Pi_{v=1}^{n} L(u,v) < \infty \triangleright \mathit{find}(u,v) \rightarrow \ \mathit{Search}'(v))$$

In [4] we derive, mainly using algebraic reasoning, a number of schedules that refine the schedule *Search'*. These schedules impose depth-first and (parallel) breadth-first orderings on the execution of the shortest paths program. Figure 5 illustrates the method of refinement. It shows how the schedules are related by the notions of refinement.

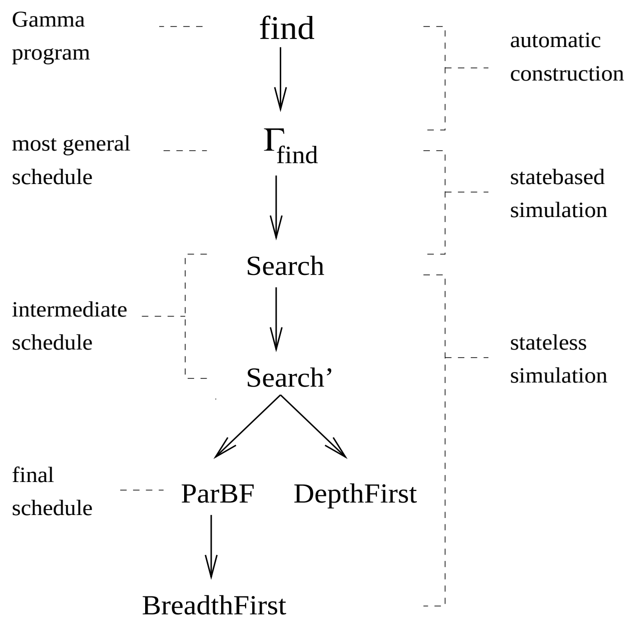

Figure 5: Lattice of Refinements for Shortest Path Schedules

5 Conclusions

Our aim is to devise a formal method for the design of (parallel) programs where computation and coordination are addressed separately. In our approach we use Gamma to express the basic computations of a program with only a minimum of control. In a second stage of the design process we exploit the highly nondeterministic behaviour of Gamma to introduce explicit control by using a separate coordination language.

It is desirable that by adding operational detail, the established correctness of a Gamma program remains unaffected. We therefore aim at a correctness preserving derivation process. The starting point is the most general schedule, which can be regarded as a model for Gamma's operational behaviour.

In this paper we discussed different notions of refinement, based on the concept of simulation. Each of these notions has an associated proof method. Together, these notions can be combined into a hybrid method that can be used effectively for proving refinements between schedules.

The concept of statebased simulation provides the most powerful proof method in the sense that it allows to resolve nondeterminism by using properties of the multiset. However, the entailed refinement relation is not a precongruence. In practice this results in complex proofs for large programs.

Stateless simulation yields a smaller refinement relation that is precongruent. It induces a number of refinement laws that can be used in an algebraic and hence compositional style of reasoning about schedules. It is less powerful than statebased refinement because of its neglect of the multiset.

By combining both notions of refinement we obtain a hybrid proof method that can be used to reason about different properties of schedules. It would be interesting to learn whether a precongruence relation exists that is larger than stateless refinement but still contained in statebased refinement. Work on this topic is currently in progress.

Acknowledgments

This report has benefited from detailed comments by Edwin de Jong, Chris Hankin and Juarez Milhaert Filho.

References

[1] J.-P. Banâtre and D. Le Métayer. The GAMMA model and its discipline of programming. *Science of Computer Programming*, 15:55–77, November 1990.

[2] J.-P. Banâtre and D. Le Métayer. Programming by multiset transformation. *Communications of the ACM*, 36(1):98–111, January 1993.

[3] M. Chaudron. Schedules for multiset transformer programs. Technical Report 94-36, Rijksuniversiteit Leiden, Departement of Mathematics and Computing Science, P.O. Box 9512, 2300 RA Leiden, The Netherlands, December 1994.

[4] M. Chaudron. Towards compositional design of schedules for multiset transformer programs. Technical Report 95-32, Rijksuniversiteit Leiden, Departement of Mathematics and Computing Science, P.O. Box 9512, 2300 RA Leiden, The Netherlands, November 1995.

[5] M. Chaudron and A. C. N. van Duin. Design of strategies for solving triangular systems using Gamma. Technical Report 96-08, Rijksuniversiteit Leiden, Departement of Mathematics and Computing Science, P.O. Box 9512, 2300 RA Leiden, The Netherlands, 1996.

[6] M. Chaudron and E. De Jong. Notions of refinement for a coordination language for Gamma. In *Proceedings of the Theory and Formal Methods Workshop 1996*. Imperial College Press, 1996. (to be published).

[7] M. Chaudron and E. De Jong. Schedules for multiset transformer programs. In *Coordination Programming: Mechanisms, Models and Semantics*. Imperial College Press, 1996.

[8] M. Chaudron and E. De Jong. Towards a compositional method for coordinating Gamma programs. In *LNCS 1061, Proceedings Coordination '96*, pages 107–123. Springer Verlag, 1996.

[9] B. A. Davey and H. A. Priesty. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[10] D. Gelernter and N. Carriero. Coordination languages and their significance. *Communications of the ACM*, 35(2):97–107, February 1992.

[11] D. Gries. *The Science of Computer Programming.* Springer-Verlag, 1981.

[12] C. Hankin, D. Le Métayer, and D. Sands. A calculus of GAMMA programs. In *5th International Workshop on Languages and Compilers for Parallel Computing, LNCS 757*, pages 342–355. Springer-Verlag, 1992.

[13] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[14] D. M. R. Park. Concurrency and automata on infinite sequences. In *LNCS 104, Proceedings of the 5th G. I. Conference on Theoretical Computer Science*, pages 167–183. Springer-Verlag, 1980.

6 Appendix On Multisets

Definition 19 *Let A be a set.*

1. *A multiset over A is a function $M : A \to \mathbb{N}$.*
2. *Let $\mathbb{M}$ be the set of multisets; i.e. $\mathbb{M} = \{M \mid M \text{ is a multiset}\}$.*

Definition 20 *Let A be a set and let M and N be multisets over A.*

1. *a is a member of M, denoted $a \in M$, if $M(a) > 0$.*
2. *M is equal to N, denoted $M = N$, if $M(a) = N(a)$ for all $a \in A$.*
3. *M is a sub-multiset of N, denoted $M \subseteq N$, if $M(a) \leq N(a)$ for all $a \in A$.*

Definition 21 *Let A be a set and let M and N be multisets over A.*

1. *$M \cup N = \{(a, M(a) + N(a)) \mid a \in A\}$ is the* union *of M and N.*
2. *$M \cap N = \{(a, min(M(a), N(a))) \mid a \in A\}$ is the* intersection *of M and N.*
3. *$M \ominus N = \{(a, M(a) \dot{-} N(a))) \mid a \in A\}$ where $x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$ is the* difference *between M and N.*

Definition 22 *Let N and N' be multisets. A* multiset substitution *that replaces N by N' is a function $\sigma : \mathbb{M} \to \mathbb{M}$ that is written as N'/N. Formally*

$$\sigma(M) = \begin{cases} (M \ominus N) \cup N' & \text{if } N \subseteq M \\ M & \text{otherwise} \end{cases}$$

To conform with conventional notation for substition, we also write $M[\sigma]$ to denote the application of σ to M.

Definition 23 *Let M be a multiset and let $\sigma_1 = N_1/M_1$ and $\sigma_2 = N_2/M_2$ be multisets substitutions.*

1. *σ_1 is* independent from *σ_2 in M, denoted $M \models \sigma_1 \triangleleft \sigma_2$, if $N_1 \subseteq (M \ominus N_2) \cup N_2'$.*
2. *If σ_1 and σ_2 are mutually independent from each other, more succinctly called* independent, *then we write $M \models \sigma_1 \bowtie \sigma_2$.*

Lemma 6.1 *Let M be a multiset and let $\sigma_1 = N_1'/N_1$ and $\sigma_2 = N_2'/N_2$ be multiset substitutions. If $N_1 \subseteq M$, $N_2 \subseteq M$ and $M \models \sigma_1 \bowtie \sigma_2$, then $\sigma_2 \cdot \sigma_1(M) = \sigma_1 \cdot \sigma_2(M)$.*

Proof Recall that $M(x)$ denotes the number of elements x in multiset M. We reason as follows

$$
\begin{array}{ll}
 & x \in \sigma_2 \cdot \sigma_1(M) \\
\Leftrightarrow & \hfill \text{subst. } \sigma_1, N_1 \subseteq M \\
 & x \in \sigma_2((M \ominus N_1) \cup N_1') \\
\Leftrightarrow & \hfill \text{subst. } \sigma_2, N_2 \subseteq (M \ominus N_1) \cup N_1' \\
 & x \in (((M \ominus N_1) \cup N_1') \ominus N_2) \cup N_2' \\
\Leftrightarrow & \hfill \text{def. } \ominus, \cup, N_1 \subseteq M, N_2 \subseteq M \ominus N_1 \cup N_1' \\
 & M(x) - N_1(x) + N_1'(x) - N_2(x) + N_2'(x) \geq 1 \\
\Leftrightarrow & \hfill \text{arithmetic} \\
 & M(x) - N_2(x) + N_2'(x) - N_1(x) + N_1'(x) \geq 1 \\
\Leftrightarrow & \hfill \text{def. } \ominus, \cup, N_2 \subseteq M, N_1 \subseteq M \ominus N_2 \cup N_2' \\
 & x \in (((M \ominus N_2) \cup N_2') \ominus N_1) \cup N_1' \\
\Leftrightarrow & \hfill \text{subst. } \sigma_2, N_1 \subseteq (M \ominus N_2) \cup N_2' \\
 & x \in \sigma_1((M \ominus N_2) \cup N_2') \\
\Leftrightarrow & \hfill \text{subst. } \sigma_1, N_2 \subseteq M \\
 & x \in \sigma_1 \cdot \sigma_2(M)
\end{array}
$$

□