REFINEMENT-BASED REASONING OF OPTIMIZED REACTIVE SYSTEMS

Mitesh Jain

October, 2017

Submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

to the

Faculty of the College of Computer and Information Science

Northeastern University

Boston, Massachusetts

Abstract

Abstract:

Acknowledgments

Contents

1 Introduction **1**

2 Preliminaries **3**

3 Lattice of refinements **5**

3.1 Notions of refinement 6

3.2 SUMMARY 9

3.3 TODO 10

4 Skipping Simulation **11**

4.1 Running Example 11

4.2 Skipping Simulation 13

4.2.1 Algebraic Properties 15

4.3 Skipping Refinement 19

4.4 Mechanised Reasoning 25

4.4.1 Reduced Well-Founded Skipping Simulation 25

4.4.2 Well-founded Skipping Simulation 31

4.4.3 Reduced Local Well-founded Skipping Simulation 33

4.4.4 Local well-founded Skipping Simulation 36

4.5 Summary 39

5 Reconciling Simulation **41**

5.1 Running Example (Continued) 41

5.2 Reconciling Simulation 42

5.3 Algebraic Properties 44

5.4 Reconciling refinement 45

5.5 Mechanised reasoning 47

5.5.1 Reduced Well-founded Reconciling Simulation 47

5.5.2 Well-founded Reconciling Simulation 51

5.5.3 Well-founded Reconciling Simulation with Explicit Stuttering 54

5.6 Summary 61

6 Case Studies **65**

6.1 Event Processing Systems 66

6.2 Superword Level Parallelism with SIMD instructions 71

7 Related Work **75**

List of Figures

3.1.1 The lattice of refinements. 9

4.1.1 Event Processing System 13

4.2.1 An example TS to show that SKS is not closed under intersection: . 16

4.4.1 Reduced well-founded skipping simulation (a solid line with arrow indicates the transition relation, a dashed orange line indicates that states are related by B) 25

4.4.2 Well-founded skipping simulation (a solid line with arrow indicates the transition relation, a dashed orange line indicates that states are related by B) 31

4.4.3 Reduced local well-founded skipping simulation (a solid line with arrow indicates the transition relation, a dashed orange line indicates that states are related by B and a solid blue line indicate the states are related by $\mathcal{O}$) 34

4.4.4 Local well-founded skipping simulation (a solid line with arrow indicates the transition relation, a dashed orange line indicates that states are related by B and a solid blue line indicate the states are related by $\mathcal{O}$) 37

5.3.1 An example to show that reconciling simulation is not compositional 45

5.5.1 Reduced well-founded reconciling simulation 48

5.5.2 Well-founded reconciling simulation 51

5.5.3 Well-founded reconciling simulation with explicit stuttering 56

6.2.1 Superword Parallelism 71

6.2.2 Syntax and Semantics of Scalar and Vector Program 72

Nomenclature

pmatch A generic notion of matching fullpaths, page 7

Chapter 1

Introduction

Chapter 2

Preliminaries

A transition system model of a reactive system captures the concept of a state, atomic transitions that modify state during the course of a computation, and what is observable in a state. Any system with a well defined operational semantics can be mapped to a labeled transition system. Hence it is well-suited to describe and analyse a large class of reactive systems.

Definition 1 (Labeled Transition System)**.** A labeled transition system (TS) is a structure $\langle S, \rightarrow, L \rangle$, where S is a non-empty (possibly infinite) set of states, $\rightarrow \subseteq S \times S$, is a left-total transition relation (every state has a successor), and L is a labeling function whose domain is S.

Notation: We first describe the notational conventions used in the paper. Function application is sometimes denoted by an infix dot "." and is left-associative. For a binary relation R, we often write xRy instead of $(x, y) \in R$. The composition of relation R with itself i times (for $0 < i \leq \omega$) is denoted R^i ($\omega = \mathbb{N}$ and is the first infinite ordinal). Given a relation R and $1 < k \leq \omega$, $R^{<k}$ denotes $\bigcup_{1 \leq i < k} R^i$ and $R^{\geq k}$ denotes $\bigcup_{\omega > i \geq k} R^i$. Instead of $R^{<\omega}$ we often write the more common R^+. $\uplus$ denotes

the disjoint union operator. Quantified expressions are written as $\langle Qx\colon r\colon t\rangle$, where Q is the quantifier (*e.g.*, $\exists, \forall, min, \bigcup$), x is a bound variable, r is an expression that denotes the range of variable x (*true*, if omitted), and t is a term.

Let $\mathcal{M} = \langle S, \rightarrow, L\rangle$ be a transition system. An $\mathcal{M}$-path is a sequence of states such that for adjacent states, s and u, $s \rightarrow u$. The j^{th} state in an $\mathcal{M}$-path is denoted by $\sigma.j$. An $\mathcal{M}$-path σ starting at state s is a *fullpath*, denoted by $fp.\sigma.s$, if it is infinite. An $\mathcal{M}$-segment, $\langle v_1, \ldots, v_k\rangle$, where $k \geq 1$ is a finite $\mathcal{M}$-path and is also denoted by $\vec{v}$. The length of an $\mathcal{M}$-segment $\vec{v}$ is denoted by $|\vec{v}|$. Let *INC* be the set of strictly increasing sequences of natural numbers starting at 0. The i^{th} partition of a fullpath σ with respect to $\pi \in INC$, denoted by ${}^{\pi}\sigma^{i}$, is given by an $\mathcal{M}$-segment $\langle \sigma(\pi.i), \ldots, \sigma(\pi(i+1)-1)\rangle$.

Chapter 3

Lattice of refinements

In this chapter, we introduce a lattice of notions of refinement for reasoning about correctness of reactive systems. Reasoning based on refinement involves relating a specification system, say $\mathcal{A}$, and an implementation system, say $\mathcal{C}$. We say that $\mathcal{C}$ *refines* (implements) $\mathcal{A}$ iff all behaviors of $\mathcal{C}$ are the behaviors that are allowed by $\mathcal{A}$. But, $\mathcal{A}$ and $\mathcal{C}$ describe their behaviors at different levels of abstraction. A natural question to ask is what is an strongest notion of refinement to relate their behaviors? Since a specification is a part of the trusted computing base, it is highly desirable to keep it as simple as possible. One must not convolute a specification because a given notion of refinement is too strong to analyze correctness of an implementation. Thus it is useful to have a collection of notion of correctness and choose the strongest notion appropriate for the verification task at hand. Towards this we present a unified exposition of several notions of refinement. In total, we study sixteen notions of refinement To the best of our knowledge, twelve notions are new and have not been previously studied in the literature. We present a novel concise characterization for the notions and show that they form a lattice.

3.1 Notions of refinement

All notions of refinement that we study are formulated using different notions of *matching* fullpaths under a binary relation. Given a transition system $\mathcal{M} = \langle S, \rightarrow, L \rangle$ and a binary relation $B \subseteq S \times S$, a notion of match establishes when two fullpaths σ and δ in $\mathcal{M}$ are related by B. All notions of matching partition fullpaths σ and δ in to finite non-empty segments. What they differ in are the constraint they impose on the relationship between states in a segment in σ and states in the corresponding segment in δ. These differences can be put into three categories: (1) *number of states in a segment*: all partitions in a fullpath (σ or δ) are of length 1 or ≥ 1; (2) *label of states in a segment*: all states in a segment of a fullpath (σ or δ), are labeled identically or not, and (3) *states related by* B: $\{$*all*, *first*$\}$ states in a segment in σ are related by B to $\{$*all*, *first*$\}$ states in the corresponding segment in δ.

We first define a parameterized notion of matching fullpaths; next we define a parameterized notion of simulation in terms of a *single* transition system. Finally, we define notions of refinement that relates two transition systems: an *abstract* transition system and a *concrete* transition system. This approach has some technical advantages and it is easy to lift a notion defined on a single transition system to one that relates two transition system by taking the disjoint union of the transition systems.

Let $\mathcal{O} = \{SIM, ST, WST, SK\}$. Let $\mathcal{M}$ be a transition, θ be a fullpath in $\mathcal{M}$,

$\alpha \in INC$, $i \in \omega$, and $o \in \mathcal{O}$, we define the following

$$l(\theta, \alpha, o) = \begin{cases} \langle \forall i \in \omega :: \#({}^{\alpha}\theta^{i}) = 1 \rangle & o \in \{SIM\} \\ true & otherwise \end{cases}$$

$$L(\theta, \alpha, o) = \begin{cases} \langle \forall i \in \omega :: \langle \forall x, y \in {}^{\alpha}\theta^{i} :: L(x) = L(y) \rangle \rangle & o \in \{SIM, ST, WST\} \\ true & otherwise \end{cases}$$

$$P(\theta, \alpha, i, o) = \begin{cases} \{\theta.\alpha.i\} & o \in \{WST, SK\} \\ {}^{\alpha}\theta^{i} & otherwise \end{cases}$$

Next, using parameters $o^L, o^R \in \mathcal{O}$, we define a notion of *pmatch* that relates fullpaths σ and δ in $\mathcal{M}$ under a binray relation B.

Definition 2. [*pmatch*] Let $\mathcal{M} = \langle S, \rightarrow, L \rangle$ be a transition system, σ, δ be fullpaths in $\mathcal{M}$, and $\mathcal{O} = \{SIM, ST, WST, SK\}$. For $\pi, \xi \in INC$, $o^L, o^R \in \mathcal{O}$, and binary relation $B \subseteq S \times S$, we define

$$pcorr[o^L, o^R](B, \sigma, \pi, \delta, \xi) \equiv \langle \forall i \in \omega :: \langle \forall s, w : s \in P(\sigma, \pi, i, o^L) \wedge w \in P(\delta, \xi, i, o^R) : sBw \rangle \rangle$$

and

$$pmatch[o^L, o^R](B, \sigma, \delta) \equiv \langle \exists \pi, \xi \in INC :: pcorr[o^L, o^R](B, \sigma, \pi, \delta, \xi) \wedge l(\sigma, \pi, o^L) \wedge l(\delta, \xi, o^R) \wedge L(\sigma, \pi, o^L) \wedge L(\delta, \xi, o^R) \rangle$$

Example 1: Let o^L be ST and o^R be ST. Substituting the value of o^L, o^R in the above definition, we have that $pmatch[ST, ST](B, \sigma, \delta) \equiv \langle \forall i \in \omega :: \langle \forall s, w : s \in {}^{\pi}\sigma^{i} \wedge w \in {}^{\xi}\delta^{i} : sBw \rangle \wedge true \wedge true \wedge \langle \forall x, y \in {}^{\pi}\sigma^{i} :: L(x) = L(y) \rangle \wedge \langle \forall x, y \in {}^{\xi}\delta^{i} :: L(x) = L(y) \rangle \rangle$. This notion of match is used to define the notion of stuttering simulation and stuttering bisimulation [10].

Example 2: Let o^L be ST and o^R be SK. Substituting the value of o^L, o^R in the above definition, we have that $pmatch[SK, ST](B, \sigma, \delta) \equiv \langle \forall i \in \omega :: \langle \forall s, w : s \in {}^{\pi}\sigma^i \wedge w \in \{\delta.\xi.i\} : sBw \rangle \wedge true \wedge true \wedge \langle \forall x, y \in {}^{\pi}\sigma^i :: L(x) = L(y) \rangle \wedge true \rangle$. This notion of match is used to define the notion of skipping simulation Chapter 5.

Example 3: Let o^L be SK and o^R be SK. Substituting the value of o^L, o^R in the above definition, we have that $pmatch[SK, SK](B, \sigma, \delta) \equiv \langle \forall i \in \omega :: \langle \forall s, w : s \in \{\sigma.\pi.i\} \wedge w \in \{\delta.\xi.i\} : sBw \rangle \wedge true \wedge true \wedge true \wedge true \rangle$. This notion of match is used to define the notion of reconciling simulation in Chapter 5.

Remark 1. We require the condition L on the labeling of states in a segment to define notions based on weak stuttering. In absence of weak stuttering, requiring that all states that are related by B are also labeled identically would suffice.

Definition 3. Let $\mathcal{M} = \langle S, \rightarrow, L \rangle$ be a transition system, $\mathcal{O} = \{SIM, ST, WST, SK\}$, and $o^L, o^R \in \mathcal{O}$. We say $B \subseteq S \times S$ is an $L\_o^L\_R\_o^R$ simulation relation on $\mathcal{M}$ *iff* for all $s, w \in S$ and sBw both of the following holds:

- $L.s = L.w$
- $\langle \forall \sigma : fp.\sigma.s : \langle \exists \delta : fp.\delta.w : pmatch[o^L, o^R](B, \sigma, \delta) \rangle \rangle$.

Let $\mathcal{R} = \bigcup_{o^L, o^R \in \mathcal{O}} \{L\_o^L\_R\_o^R\}$ be the set of all simulation relations. Next, a notion of simulation in $\mathcal{R}$, which is defined in terms of a *single* transition system, is used to define the corresponding notion of refinement, a notion that relates *two* transition systems: an *abstract* transition system and a *concrete* transition system.

Definition 4. Let $\mathcal{M}_{\mathrm{A}} = \langle S_A, \xrightarrow{A}, L_A \rangle$ and $\mathcal{M}_{\mathrm{C}} = \langle S_C, \xrightarrow{C}, L_C \rangle$ be transition systems, X be an element in $\mathcal{R}$, and let $r : S_C \rightarrow S_A$ be a refinement map. We say $\mathcal{M}_C$ is an X-refinement of $\mathcal{M}_A$ with respect to refinement map r, written $\mathcal{M}_C \preceq_r^X \mathcal{M}_A$, if there exists a relation $B \in S_C \times S_A$ such that all of the following hold.

(1) $\langle \forall s \in S_C :: sBr.s \rangle$

(2) B is an X simulation relation on $\langle S_C \uplus S_A, \xrightarrow{C} \uplus \xrightarrow{A}, \mathcal{L} \rangle$ where $\mathcal{L}.s = L_A(s)$ for $s \in S_A$, and $\mathcal{L}.s = L_A(r.s)$ for $s \in S_C$.

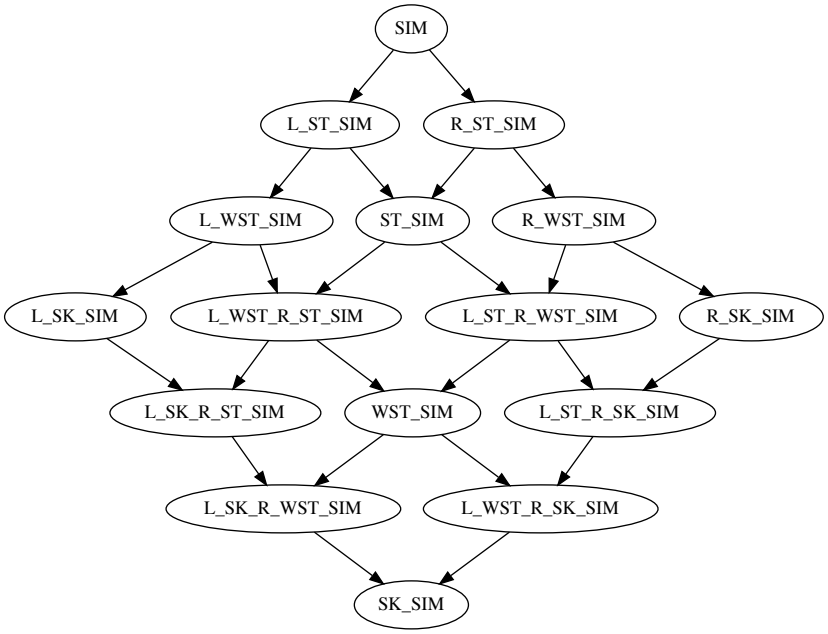

Figure 3.1.1: The lattice of refinements.

3.2 SUMMARY

In this chapter, we inroduced twenty new notions of refinement. We presented a parameterized notion of matching fullpaths that enabled us to succinctly define the notions of refinement.

3.3 TODO

- refine the beginning of the chapter;
- annotate the lattice with known notions and new notions. The known notions should also have a name in the literature/author/year of publication associated with it;
- match the naming convention in the lattice and formalization;
- proof of the claims;
- Summary

Chapter 4

Skipping Simulation

In this chapter, we first define the notion of skipping simulation. Then we study its algebraic properties and develop a compositional theory of skipping refinement. Finally, we develop sound and complete proof methods that are amenable for automated reasoning about skipping refinement.

4.1 Running Example

Consider an example of an event processing system system (EPS). An abstract high-level specification, AEPS, of an event processing system is defined as follows. Let E be a set of *events* and V be a set of *state variables*. A *state* of AEPS is a three-tuple $\langle t, Sch, St\rangle$, where t is a natural number denoting the current time; Sch is a set of pairs (e, t_e), where $e \in E$ is an event scheduled to be executed at time $t_e \geq t$; St is an assignment to variables in V. The transition relation for the AEPS system is defined as follows. If at time t there is no $(e, t) \in Sch$, *i.e.*, there is no event scheduled to be executed at time t, then t is incremented by 1. Otherwise, we (nondeterministically) choose and execute an event of the form $(e, t) \in Sch$. The execution of an event may result in modifying St and also removing and adding a finite number of new

pairs (e', t') to Sch. We require that $t' > t$. Finally, execution involves removing the executed event (e, t) from Sch. This is a simple but a generic model of an event processing system. We place no restriction on the type of state variables or the type of events. Moreover, the ability to remove events can be used to specify systems with preemption [13]: an event scheduled to execute at some future time may be cancelled (and possibly rescheduled to be executed at a different time in future) as a result of execution of an event that preempts it.

Now consider, tEPS, an optimized implementation of AEPS. As before, a state is a three-tuple $\langle t, Sch, St \rangle$. However, unlike the abstract system which just increments time by 1 when there are no events scheduled at the current time, the optimized system finds the earliest time in future an event is scheduled to execute. The transition relation of tEPS is defined as follows. An event (e, t_e) with the minimum time is selected, t is updated to t_e and the event e is executed, as in the AEPS.

Consider an execution of AEPS and tEPS in Figure 4.1.1. (We only show the prefix of executions.) Suppose at $t = 0$, Sch be $\{(e_1, 0)\}$. The execution of event e_1 add a new pair (e_2, k) to Sch, where k is a positive integer. AEPS at $t = 0$, executes the event e_1, adds a new pair (e_2, k) to Sch, and updates t to 1. Since no events are scheduled to execute before $t = k$, the AEPS system repeatedly increments t by 1 until $t = k$. At $t = k$, it executes the event e_2. At time $t = 0$, tEPS executes e_1. The next event is scheduled to execute at time $t = k$; hence it updates in one step t to k. Next, in one step it executes the event e_2. Note that tEPS runs faster than AEPS by *skipping* over abstract states when no event is scheduled for execution at the current time. If $k > 1$, the step from s_2 to s_3 in tEPS neither corresponds to stuttering nor to a single step of the specification. Therefore, notions of refinement based on stuttering simulation and bisimulation are not directly applicable for reasoning about the correctness of tEPS. In this chapter, we define skipping refinement, a new no-

tion of refinement that directly supports reasoning about optimized reactive systems that can run faster than their high-level specification systems. We also develop a compositional theory of skipping refinement and provide sound and complete proof methods that require only local reasoning, thereby enabling mechanised verification of skipping refinement.

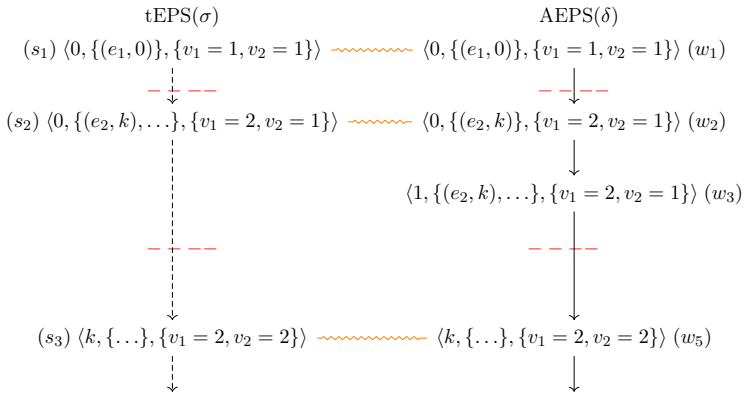

Figure 4.1.1: Event Processing System

4.2 Skipping Simulation

The definition of skipping simulation is based on the notion of *smatch*, a new notion of matching fullpaths. Informally, we say that a fullpath σ *smatches* a fullpath δ under the relation B if the fullpaths can be partitioned in to non-empty, finite segments such that all elements in a segment of σ are related to the first element in the corresponding segment of δ. Using the notion of *smatch*, skipping simulation is defined as follows: a relation B is a skipping simulation on a transition system $\mathcal{M} = \langle S, \rightarrow, L\rangle$, if for any $s, w \in S$ such that sBw, s and w are labeled identically and any fullpath starting at s can be smatched by some fullpath starting at w. Notice that we define skipping simulation using a single transition system, while our

ultimate goal is to define a notion of refinement that relates two transition system: an abstract transition system and a concrete transition system. We will see that this approach has some technical advantages. Moreover, it is easy to lift the notion defined on a single transition system to one that relates two transition systems by considering their disjoint union.

Definition 5 (smatch)**.** Let $\mathcal{M} = \langle S, \rightarrow, L\rangle$ be a transition system, σ, δ be fullpaths in $\mathcal{M}$. For $\pi, \xi \in INC$ and binary relation $B \subseteq S \times S$, we define

$scorr(B, \sigma, \pi, \delta, \xi) \equiv \langle \forall i \in \omega :: \langle \forall s \in {}^{\pi}\sigma^{i} :: sB\delta(\xi.i)\rangle\rangle$ *and*

$smatch(B, \sigma, \delta) \equiv \langle \exists \pi, \xi \in INC :: scorr(B, \sigma, \pi, \delta, \xi)\rangle$.

In Figure 4.1.1, we illustrate the notion of smatching using our running example of an event processing system. In the figure, σ is a fullpath of tEPS and δ is a fullpath of AEPS. (We only show the prefix of the fullpaths.) The other parameter for matching is the relation B, which is just the identity function. In order to show that $smatch(B, \sigma, \delta)$ holds, we have to find $\pi, \xi \in INC$ satisfying the definition. In the figure, we separate the partitions induced by our choice for π, ξ using $--$ and connect elements related by B with $\sim$. Since all elements of a σ partition are related to the first element of the corresponding δ partition, $scorr(B, \sigma, \pi, \delta, \xi)$ holds, therefore, $smatch(B, \sigma, \delta)$ holds.

Definition 6 (Skipping Simulation)**.** $B \subseteq S \times S$ is a skipping simulation on a transition system $\mathcal{M} = \langle S, \rightarrow, L\rangle$ *iff* for all s, w such that sBw, both of the following hold.

(SKS1) $L.s = L.w$

(SKS2) $\langle \forall \sigma : fp.\sigma.s : \langle \exists \delta : fp.\delta.w : smatch(B, \sigma, \delta)\rangle\rangle$

4.2.1 Algebraic Properties

SKS enjoys several useful algebraic properties. In particular, it is closed under union and relational composition. The later property enables us to develop a theory of refinement that enables (vertical) modular reasoning of complex reactive systems.

Lemma 1. Let $\mathcal{M} = \langle S, \rightarrow, L \rangle$ be a transition system and $\mathcal{C}$ be a set of SKS's on $\mathcal{M}$. Then $G = \langle \cup B : B \in \mathcal{C} : B \rangle$ is an SKS on $\mathcal{M}$.

Proof: Let $s, w \in S$ and sGw. We show that SKS1 and SKS2 hold for G. Since $G = \langle \cup B : B \in \mathcal{C} : B \rangle$, there is an SKS $B \in \mathcal{C}$ such that sBw. Since B is an SKS on $\mathcal{M}$, we have that $L.s = L.w$. Hence, SKS1 holds for G. Next SKS2 also holds for B, *i.e.*, for any fullpath σ starting at s, there is a fullpath δ starting at w such that $smatch(B, \sigma, \delta)$ holds. From the definition of $smatch$, there exists $\pi, \xi \in INC$ such that for all $i \in \omega$ and for all $s \in {}^{\pi}\sigma^{i}$, $sB\delta(\xi.i)$ holds. Since $B \subseteq G$, $sG\delta(\xi.i)$ holds. Hence $smatch(G, \sigma, \delta)$ holds, *i.e.*, SKS2 holds for G.

□

Corollary 2. For any transition system $\mathcal{M}$, there is a greatest SKS on $\mathcal{M}$.

Proof: Let $\mathcal{C}$ be the set of all SKS's on $\mathcal{M}$ and let $G = \langle \cup B : B \in \mathcal{C} : B \rangle$. By construction, G is the greatest and from Lemma 1, G is an SKS on $\mathcal{M}$.

□

The following lemma shows that SKS is not closed under intersection and negation.

Lemma 3. SKS are not closed under negation and intersection.

Proof: Consider a TS $\mathcal{M} = \langle S = \{a, b\}, \rightarrow = \{(a, a), (b, b)\}, L = \{(a, 1), (b, 2)\} \rangle$. The identity relation is an SKS, but its negation is not.

An example transition system to show that SKS is not closed under intersection appears in Figure 4.2.1.

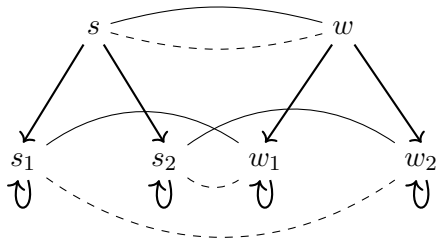

Figure 4.2.1: An example showing that SKS is not closed under intersection. Consider a TS with set of states $S = \{s, w, s_1, s_2, w_1, w_2\}$. The transition relation is denoted by solid arrows and all states are labeled identically. The first SKS relation B_1, denoted by solid lines, is $\{(s, w), (s_1, w_1), (s_2, w_2)\}$. The second SKS relation B_2, denoted by dashed lines is $\{(s, w), (s_1, w_2), (s_2, w_1)\}$. $B_1 \cap B_2$ is $\{(s, w)\}$ but does not include any related children of s and w.

The following lemma shows that skipping simulation is closed under relational composition.

Lemma 4. Let $\mathcal{M}$ be a transition system. If P and Q are SKS's on $\mathcal{M}$, then $R = P; Q$ is an SKS on $\mathcal{M}$.

Proof: To show that R is an SKS on $\mathcal{M} = \langle S, \rightarrow, L\rangle$, we show that for any $s, w \in S$ such that sRw, SKS1 and SKS2 hold. Let $s, w \in S$ and sRw. From the definition of R, there exists $x \in S$ such that sPx and xQw. Since P and Q are SKS's on $\mathcal{M}$, $L.s = L.x = L.w$, hence, SKS1 holds for R.

To prove that SKS2 holds for R, consider a fullpath σ starting at s. Since P and Q are SKSs on $\mathcal{M}$, there is a fullpath τ in $\mathcal{M}$ starting at x, a fullpath δ in $\mathcal{M}$ starting at w and $\alpha, \beta, \theta, \gamma \in \mathit{INC}$ such that $\mathit{scorr}(P, \sigma, \alpha, \tau, \beta)$ and $\mathit{scorr}(Q, \tau, \theta, \delta, \gamma)$ hold. We use the fullpath δ as a witness and define $\pi, \xi \in \mathit{INC}$ such that $\mathit{scorr}(R, \sigma, \pi, \delta, \xi)$ holds.

We define a function, r, that given i, corresponding to the index of a partition of τ under β, returns the index of the partition of τ under θ in which the first element

of τ's i^{th} partition under β resides.

$$r.i = j \text{ iff } \theta.j \leq \beta.i < \theta(j+1)$$

Note that r is indeed a function, as every element of τ resides in exactly one partition of θ. Also, since there is a correspondence between the partitions of α and β, (by $scorr(P, \sigma, \alpha, \tau, \beta)$), we can apply r to indices of partitions of σ under α to find where the first element of the corresponding β partition resides. Note that r is non-decreasing: $a < b \Rightarrow r.a \leq r.b$.

We define $\pi\alpha \in INC$, a strictly increasing sequence that will allow us to merge adjacent partitions in α as needed to define the strictly increasing sequence π on σ used to prove SKS2. Partitions in π will consist of one or more α partitions. Given i, corresponding to the index of a partition of σ under π, the function $\pi\alpha$ returns the index of the corresponding partition of σ under α.

$$\pi\alpha(0) = 0$$

$$\pi\alpha(i) = \text{ min } j \in \omega \text{ s.t. } |\{k : 0 < k \leq j \wedge r.k \neq r(k-1)\}| = i$$

Note that $\pi\alpha$ is an increasing function, *i.e.*, $a < b \Rightarrow \pi\alpha(a) < \pi\alpha(b)$. We now define π as follows.

$$\pi.i = \alpha(\pi\alpha.i)$$

There is an important relationship between r and $\pi\alpha$

$$r(\pi\alpha.i) = \cdots = r(\pi\alpha(i+1) - 1)$$

That is, for all α partitions that are in the same π partition, the initial states of the

corresponding β partitions are in the same θ partition.

We define ξ as follows.

$$\xi.i = \gamma(r(\pi\alpha.i))$$

We are now ready to prove SKS2. Let $s \in {}^{\pi}\sigma^{i}$. We show that $sR\delta(\xi.i)$. By the definition of π, we have

$$s \in {}^{\alpha}\sigma^{\pi\alpha.i} \vee \cdots \vee s \in {}^{\alpha}\sigma^{\pi\alpha(i+1)-1}$$

Hence,

$$sP\tau(\beta(\pi\alpha.i)) \vee \cdots \vee sP\tau(\beta(\pi\alpha(i+1)-1))$$

Note that by the definition of r (apply r to $\pi\alpha.i$):

$$\theta(r(\pi\alpha.i)) \leq \beta(\pi\alpha.i) < \theta(r(\pi\alpha.i)+1)$$

Hence,

$$\tau(\beta(\pi\alpha.i))Q\delta(\gamma(r(\pi\alpha.i))) \vee \cdots \vee \tau(\beta(\pi\alpha(i+1)-1))Q\delta(\gamma(r(\pi\alpha(i+1)-1)))$$

By the definition of ξ and the relationship between r and $\pi\alpha$ described above, we simplify the above formula as follows.

$$\tau(\beta(\pi\alpha.i))Q\delta(\xi.i) \vee \cdots \vee \tau(\beta(\pi\alpha(i+1)-1))Q\delta(\xi.i)$$

Therefore, by the definition of R, we have that $sR\delta(\xi.i)$ holds.

□

Theorem 5. The reflexive transitive closure of an SKS is an SKS.

Proof: Let $\mathcal{M} = \langle S, \rightarrow, L \rangle$ be a TS and B be an SKS on $\mathcal{M}$. The reflexive, transitive closure of B, written B^*, is $\langle \cup i \in \omega :: B^i \rangle$. First we show that for all $i \in \omega$, B^i is an SKS using induction on natural numbers. In the base case, B^0, the identity relation, is clearly an SKS. For $i \geq 0$, we have that $B^{i+1} = B; B^i$; from Lemma 4 and the induction hypothesis, we have that B^{i+1} is an SKS on $\mathcal{M}$. Finally, from Lemma 1, we have that $\langle \cup i \in \omega :: B^i \rangle$, *i.e.*, B^* is an SKS on $\mathcal{M}$.

□

Theorem 6. Given a TS $\mathcal{M}$, the greatest SKS on $\mathcal{M}$ is a preorder.

Proof: Let G be the greatest SKS on $\mathcal{M}$. From Theorem 5, G^* is an SKS. Hence $G^* \subseteq G$. Furthermore, since $G \subseteq G^*$, we have that $G = G^*$, *i.e.*, G is reflexive and transitive.

□

4.3 Skipping Refinement

In this section, we use the notion of skipping simulation, which is defined in terms of a *single* transition system to define the notion of skipping refinement, a notion that relates *two* transition systems: an *abstract* transition system and a *concrete* transition system. Informally, if a concrete system is a skipping refinement of an abstract system, then its observable behaviors are also behaviors of the abstract system, modulo skipping (which includes stuttering). The notion is parameterized by a *refinement map*, a function that maps concrete states to their corresponding abstract states. A refinement map along with a labeling function determines what is observable at a concrete state. Using the algebraic properties of skipping simulation proved in the previous section, we show that skipping refinement can be used for the modular reasoning of complex reactive systems using a stepwise refinement

methodology.

Note that we do not place any restriction on the state space size or the branching factor of the transition relations of the abstract and the concrete systems, and both can be of arbitrary infinite cardinalities. Thus, the theory of skipping refinement and sound and complete proof methods for reasoning about skipping refinement (developed in the Section 4.4) provide a reasoning framework that can be used to analyse a large class of reactive systems.

Definition 7 (Skipping Refinement)**.** Let $\mathcal{M}_{\mathrm{A}} = \langle S_A, \xrightarrow{A}, L_A \rangle$ and $\mathcal{M}_{\mathrm{C}} = \langle S_C, \xrightarrow{C}, L_C \rangle$ be transition systems and let $r : S_C \rightarrow S_A$ be a refinement map. We say $\mathcal{M}_C$ is a *skipping refinement* of $\mathcal{M}_A$ with respect to r, written $\mathcal{M}_C \lesssim_r \mathcal{M}_A$, if there exists a binary relation B such that all of the following hold.

1. $\langle \forall s \in S_C :: sBr.s \rangle$ *and*

2. B is an SKS on $\langle S_C \uplus S_A, \xrightarrow{C} \uplus \xrightarrow{A}, \mathcal{L} \rangle$ where $\mathcal{L}.s = L_A(s)$ for $s \in S_A$, and $\mathcal{L}.s = L_A(r.s)$ for $s \in S_C$.

Notice that the above definition does not place any restriction on the choice of refinement map and depending on the systems under analysis one has great flexibility in its choice. In particular, the refinement map is not restricted to a simple *projection function* [2] that projects the observable component of a concrete state. In conjunction with the sound and complete proof methods presented in the next section, this provides a theory of refinement and a reasoning framework that is applicable to a large class of optimized reactive systems. The flexibility in the choice of refinement map is also useful in developing computationally efficient methods for verification and testing [12, 8]. However, one should be prudent in the choice of refinement map; a complicated refinement map may bypass the verification problem. We discuss this aspect in more detail in Chapter 7.

Next, we use the property that skipping simulation is closed under relational composition to show that skipping refinement supports modular reasoning using stepwise refinement approach. In order to verify that a low-level complex implementation $\mathcal{M}_C$ refines a simple high-level abstract specification $\mathcal{M}_A$ one proceeds as follows: starting with $\mathcal{M}_A$ define a sequence of intermediate lower level systems leading to the final complex implementation $\mathcal{M}_C$. At each step in the sequence, show that system at the current step is a refinement of the previous one. This approach is often more scalable than a monolithic approach. This is primarily because at each step, the verification effort is largely focused only on the difference between two systems under consideration. Note that this methodology is orthogonal to (horizontal) modular reasoning that infers correctness of a system from the correctness of its sub-components.

The following lemma is useful for lifting the notion of skipping simulation, which is defined on single transition system, to the notion of skipping refinement, which relates two transition system.

Lemma 7. Let S, S_1, S_2 be a set of states such that $S_1 \cap S_2 = \varnothing$ and $S_1 \cup S_2 \subseteq S$. Let B be an SKS on $\mathcal{M} = \langle S, \rightarrow, L \rangle$ such that any state in S_1 can only reach states in S_1, and any state in S_2 can only reach states in S_2, then $B' = \{(s_1, s_2) \mid s_1 \in S_1 \wedge s_2 \in S_2 \wedge s_1 B s_2\}$ is an SKS on $\mathcal{M}$.

Proof: Let $s_1 B' s_2$. We show that SKS1 and SKS2 holds for B'. From definition of B', we have that $s_1 \in S_1$, $s_2 \in S_2$, and $s_1 B s_2$. Since B is an SKS on $\mathcal{M}$, we have that $L.s_1 = L.s_2$; hence SKS1 holds for B'. Next let σ and δ be fullpaths in $\mathcal{M}$ starting at s_1 and s_2 respectively and $\pi, \xi \in \mathit{INC}$ such that $\langle \forall i \in \omega :: \langle \forall s \in {}^{\pi}\sigma^{i} :: sB\delta(\xi.i) \rangle \rangle$ holds. Next from the assumptions that any state in S_1 can only reach states in S_1, and σ is a fullpath in $\mathcal{M}$ starting at $s_1 \in S_1$, all states in ${}^{\pi}\sigma^{i}$ are in S_1. Also, since any state in S_2 can only reach states in S_2, state $\delta(\xi.i) \in S_2$. Hence we

have that $\langle \forall i \in \omega :: \langle \forall s \in {}^{\pi}\sigma^{i} :: sB'\delta(\xi.i) \rangle \rangle$, *i.e.*, SKS2 holds for B'.

□

Theorem 8. Let $\mathcal{M}_1 = \langle S_1, \xrightarrow{1}, L_1 \rangle$, $\mathcal{M}_2 = \langle S_2, \xrightarrow{2}, L_2 \rangle$, and $\mathcal{M}_3 = \langle S_3, \xrightarrow{3}, L_3 \rangle$ be transition systems, $p : S_1 \to S_2$ and $r : S_2 \to S_3$. If $\mathcal{M}_1 \lesssim_p \mathcal{M}_2$ and $\mathcal{M}_2 \lesssim_r \mathcal{M}_3$, then $\mathcal{M}_1 \lesssim_{p;r} \mathcal{M}_3$.

Proof: Since $\mathcal{M}_1 \lesssim_p \mathcal{M}_2$, we have an SKS, say A, such that $\langle \forall s \in S_1 :: sA(p.s) \rangle$. Furthermore, from Lemma 7, without loss of generality we can assume that $A \subseteq S_1 \times S_2$. Similarly, since $\mathcal{M}_2 \lesssim_r \mathcal{M}_3$, we have an SKS, say B, such that $\langle \forall s \in S_2 :: sB(r.s) \rangle$ and $B \subseteq S_2 \times S_3$. Define $C = A; B$. Then we have that $C \subseteq S_1 \times S_3$ and $\langle \forall s \in S_1 :: sCr(p.s) \rangle$. Also, from Theorem 5, C is an SKS on $\langle S_1 \uplus S_3, \xrightarrow{1} \uplus \xrightarrow{3}, \mathcal{L} \rangle$, where $\mathcal{L}.s = L_3(s)$ if $s \in S_3$ else $\mathcal{L}.s = L_3(r(p.s))$.

□

Formally, to establish that a complex low-level implementation $\mathcal{M}_C$ refines a simple high-level abstract specification $\mathcal{M}_A$, one defines intermediate systems $\mathcal{M}_1$, $\ldots \mathcal{M}_n$, where $n \geq 1$ and establishes the following: $\mathcal{M}_C = \mathcal{M}_0 \lesssim_{r_0} \mathcal{M}_1 \lesssim_{r_1} \ldots \lesssim_{r_{n-1}} \mathcal{M}_n = \mathcal{M}_A$. Then from Theorem 8, we have that $\mathcal{M}_C \lesssim_r \mathcal{M}_A$, where $r = r_0; r_1; \ldots; r_{n-1}$. We illustrate the utility of this approach in Chapter 6 by proving correctness of two optimized event processing systems.

Theorem 9. Let $\mathcal{M} = \langle S, \to, L \rangle$ be a transition system. Let $\mathcal{M}' = \langle S', \to', L \rangle$ where $S' \subseteq S$, $\to' \subseteq S' \times S'$, $\to'$ is a left-total subset of $\to^{+}$, and $L' = L|_{S'}$. Then $\mathcal{M}' \lesssim_I \mathcal{M}$, where I is the identity function on S'.

Proof: Let B be I. Let σ be a fullpath starting at an $\mathcal{M}'$ state. To show that B is an SKS relation, the key observation is that since $\to' \subseteq \to^{+}$, there is a fullpath starting from the correspodning $\mathcal{M}$ state, say δ, such that a step in σ corresponds to a finite, positive number of steps in δ. We choose such a fullpath δ as a witness

and show σ and δ smatch under B. We consider the partitioning of σ such that a partition has only one state. Next we define the partitioning of δ. The i^{th} partition of δ includes (1) the state, say s, in $\mathcal{M}$ corresponding to the state in the i^{th} partition of σ, and (2) intermediate states in $\mathcal{M}$ required to reach from s to the state in $\mathcal{M}$ corresponding to the state in the $(i+1)^{th}$ partition of σ. It is easy to see that σ, δ and their partitions defined above satisfy *scorr* in Definition 5.

□

Corollary 10. Let $\mathcal{M}_C = \langle S_C, \xrightarrow{C}, L_C \rangle$ and $\mathcal{M}_A = \langle S_A, \xrightarrow{A}, L_A \rangle$ be transition systems, $r : S_C \to S_A$ be a refinement map. Let $\mathcal{M}'_C = \langle S'_C, \xrightarrow{C}', L'_C \rangle$ where $S'_C \subseteq S_C$, $\xrightarrow{C}'$ is a left-total subset of $\xrightarrow{C}^+$, and $L'_C = L_C|_{S'_C}$. If $\mathcal{M}_C \lesssim_r \mathcal{M}_A$ then $\mathcal{M}'_C \lesssim_{r'} \mathcal{M}_A$, where r' is $r|_{S'_C}$.

Proof: From Lemma 9, $\mathcal{M}'_C \lesssim_I \mathcal{M}_C$, where I is the identity function on $S_{C'}$. Since $\mathcal{M}_C \lesssim_r \mathcal{M}_A$, from Theorem 8, we have that $\mathcal{M}'_C \lesssim_{I;r} \mathcal{M}_A = \mathcal{M}'_C \lesssim_{r'} \mathcal{M}_A$.

□

We now illustrate the usefulness of the theory of skipping refinement using our running example of event processing systems. Consider MPEPS, an optimized EPS that uses a priority queue to find a non-empty set of events to execute next. As in Section 4.1, a state of MPEPS is a three tuple $\langle t, Sch, St \rangle$. The transition relation of MPEPS is defined as follows. Let t be the current time, and E_t be the set of events in Sch that are scheduled to execute at time t. If E_t is empty, then MPEPS uses the priority queue to find the minimum time $t' > t$ at which an event is scheduled for execution and updates the time to t'. Otherwise, MPEPS uses the priority queue to choose a non-empty subset of events in E_t and executes them. Note that we allow the priority queue in MPEPS to be deterministic or nondeterministic. For example, the priority queue may deterministically select a single event in E_t to execute, or based on considerations such as resource utilization it may execute some subset of

events in E_t in a single step. When reasoning about the correctness of MPEPS, one thing to notice is that there is a difference in the data structures used in the two systems: MPEPS uses a priority queue to effectively find the next set of events to execute in the scheduler, while AEPS uses a simple abstract set representation for the scheduler. Another thing to notice is that MPEPS can run "faster" than AEPS in two ways: it can increment time by more than 1 and it can execute more than one event in a single step. The theory of skipping refinement developed in this chapter enables us to separate out these concerns and apply a stepwise refinement approach to effectively analyse MPEPS.

First, we account for the difference in the data structures between MPEPS and AEPS. Towards, this we define an intermediate system MEPS that is identical to MPEPS except that the scheduler in MEPS is now represented as a set of event-time pairs. Under a refinement map, say p, that extracts the set of event-time pairs in the priority queue of MPEPS, a step in MPEPS can be matched by a step in MEPS. Hence, MPEPS $\lesssim_p$ MEPS. Next we account for the difference between MEPS and AEPS in the number of events the two systems may execute in a single step. Towards this, observe that the state space of MEPS and tEPS are equal and the transition relation of MEPS is a left-total subset of the transitive closure of the transition relation of tEPS. Hence, from Theorem 9, we infer that MPEPS is a skipping refinement of tEPS using the identity function, say I_1, as the refinement map, *i.e.*, MEPS $\lesssim_{I_1}$ tEPS. Next observe that the state space of tEPS and AEPS are equal and the transition relation of tEPS is left-total subset of the transition relation of AEPS. Hence, from Theorem 9, we infer that tEPS is a skipping refinement of AEPS using the identity function, say I_2, as the refinement map, *i.e.*, tEPS $\lesssim_{I_2}$ AEPS. Finally, from the transitivity of skipping refinement (Theorem 8), we conclude that $MPEPS \lesssim_{p'} AEPS$, where $p' = p; I_1; I_2$.

4.4 Mechanised Reasoning

To prove that a transition system $\mathcal{M}_C$ is a skipping refinement of a transition system $\mathcal{M}_A$ using Definition 6, requires us to show that for any fullpath in $\mathcal{M}_C$, we can find a matching fullpath in $\mathcal{M}_A$. However, reasoning about nested quantifiers over infinite sequences is often problematic using automated tools. To redress the situation, we propose four alternative characterizations of skipping simulation that are amenable for mechanical reasoning.

4.4.1 Reduced Well-Founded Skipping Simulation

We first introduce reduced well-founded skipping simulation (RWFSK). The intuition is that for any pair of states s, w that are related and a state u such that $s \rightarrow u$, there are two cases to consider (Figure 4.4.1): (a) either the step from s to u is a stuttering step and u is related to w or (b) we can match the step from s to u with one or more steps from w.

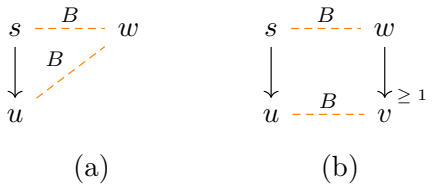

Figure 4.4.1: Reduced well-founded skipping simulation (a solid line with arrow indicates the transition relation, a dashed orange line indicates that states are related by B)

Definition 8 (Reduced Well-founded Skipping [7])**.** $B \subseteq S \times S$ is a reduced well-founded skipping relation on TS $\mathcal{M} = \langle S, \rightarrow, L \rangle$ iff :

(RWFSK1) $\langle \forall s, w \in S : sBw : L.s = L.w \rangle$

(RWFSK2) There exists a function, $rankt : S \times S \rightarrow W$, such that $\langle W, \prec \rangle$ is well-

founded and

$\langle \forall s,u,w \in S : s \to u \wedge sBw :$

(a) $(uBw \wedge rankt(u,w) \prec rankt(s,w)) \vee$

(b) $\langle \exists v : w \to^{+} v : uBv \rangle \rangle$

Observe that RWFSK2b accounts for both stuttering and skipping on the right. This is possible because skipping subsumes stuttering. Moreover, notice that the condition RWFSK2a is local, *i.e.*, it requires reasoning only about state and its successor. In contrast, the condition RWFSK2b is not local and in general may require unbounded reachability analysis, which can often be problematic for automated verification tools. Nevertheless, RWFSK is a useful proof method in case both stuttering and skipping on the right are bounded by a constant.

RWFSK characterizes skipping simulation, *i.e.*, it is a sound and complete proof method for reasoning about skipping simulation. We first prove the soundness, *i.e.*, RWFSK implies SKS.

Lemma 11 ([7]). Let $\mathcal{M}$ be a transition system. If B is an RWFSK on $\mathcal{M}$, then B is an SKS on $\mathcal{M}$.

Proof: To show that B is an SKS on $\mathcal{M} = \langle S, \to, L \rangle$, we show that for any $x, y \in S$ such that xBy, SKS1 and SKS2 hold. SKS1 follows directly from condition 1 of RWFSK.

Next we show that SKS2 holds. We start by recursively defining δ. In the process, we also define partitions π and ξ. For the base case, we let $\pi.0 = 0$, $\xi.0 = 0$ and $\delta.0 = y$. By assumption $\sigma(\pi.0)B\delta(\xi.0)$. For the recursive case, assume that we have defined $\pi.0, \ldots, \pi.i$ as well as $\xi.0, \ldots, \xi.i$ and $\delta.0, \ldots, \delta(\xi.i)$. We also assume that $\sigma(\pi.i)B\delta(\xi.i)$. Let s be $\sigma(\pi.i)$; let u be $\sigma(\pi.i+1)$; let w be $\delta(\xi.i)$. We consider two cases.

First, say that RWFSK2b holds. Then, there is a v such that $w \rightarrow^+ v$ and uBv. Let $\vec{v} = [v_0 = w, \ldots, v_m = v]$ be a finite path from w to v where $m \geq 1$. We define $\pi(i+1) = \pi.i + 1, \xi(i+1) = \xi.i + m$, ${}^{\xi}\delta^i = [v_0, \ldots, v_{m-1}]$ and $\delta(\xi(i+1)) = v$.

If the first case does not hold, *i.e.*, RWFSK2b does not hold, and RWFSK2a does hold. We define J to be the subset of the positive integers such that for every $j \in J$, the following holds.

$$\neg\langle\exists v : w \rightarrow^+ v : (\sigma(\pi.i+j)Bv)\rangle \ \wedge \tag{4.1}$$
$$\sigma(\pi.i+j)Bw \ \ \wedge \ \ \mathit{rankt}(\sigma(\pi.i+j), w) \prec \mathit{rankt}(\sigma(\pi.i+j-1), w)$$

The first thing to observe is that $1 \in J$ because $\sigma(\pi.i+1) = u$, RWFSK2b does not hold (so the first conjunct is true) and RWFSK2a does (so the second conjunct is true). The next thing to observe is that there exists a positive integer $n > 1$ such that $n \notin J$. Suppose not, then for all $n \geq 1, n \in J$. Now, consider the (infinite) suffix of σ starting at $\pi.i$. For every adjacent pair of states in this suffix, say $\sigma(\pi.i+k)$ and $\sigma(\pi.i+k+1)$ where $k \geq 0$, we have that $\sigma(\pi.i+k)Bw$ and that only RWFSK2a applies (*i.e.*, RWFSK2b does not apply). This gives us a contradiction because *rankt* is well-founded. We can now define n to be $\mathit{min}(\{l : l \in \omega \wedge l > 0 \wedge l \notin J\})$. Notice that only RWFSK2a holds between $\sigma(\pi.i+n-1)), \sigma(\pi.i+n)$ and w, hence $\sigma(\pi.i+n)Bw$ and $\mathit{rankt}(\sigma(\pi.i+n), w) \prec \mathit{rankt}(\sigma(\pi.i+n-1), w)$. Since Formula 4.1 does not hold for n, there is a v such that $w \rightarrow^+ v \wedge \sigma(\pi.i+n)Bv$. Let $\vec{v} = [v_0 = w, \ldots, v_m = v]$ be a finite path from w to v where $m \geq 1$. We are now ready to extend our recursive definition as follows: $\pi(i+1) = \pi.i + n$, $\xi(i+1) = \xi.i + m$, and ${}^{\xi}\delta^i = [v_0, \ldots, v_{m-1}]$.

Now that we defined δ we can show that SKS2 holds. We start by unwinding definitions. The first step is to show that $\mathit{fp}.\delta.y$ holds, which is true by construction. Next, we show that $\mathit{smatch}(B, \sigma, \delta)$ by unwinding the definition of *smatch*. That

involves showing that there exist π and ξ such that $scorr(B, \sigma, \pi, \delta, \xi)$ holds. The π and ξ we used to define δ can be used here. Finally, we unwind the definition of *corr*, which gives us a universally quantified formula over the natural numbers. This is handled by induction on the segment index; the proof is based on the recursive definitions given above.

□

Next we prove the completeness of RWFSK, *i.e.*, SKS implies RWFSK.

Lemma 12 ([7])**.** Let $\mathcal{M}$ be a transition system. If B is an SKS on $\mathcal{M}$, then B is an RWFSK on $\mathcal{M}$.

Let B be an SKS on $\mathcal{M}$. To show that B is an RWFSK on $\mathcal{M}$, we exhibit as witness the existence of a well-founded domain and a ranking function *rankt* that satisfy the conditions in RWFSK. Towards this, we first introduce a few definitions and lemmas.

Definition 9. Given a transition system $\mathcal{M} = \langle S, \rightarrow, L \rangle$, the *computation tree* rooted at a state $s \in S$, denoted $ctree(\mathcal{M}, s)$, is obtained by "unfolding" $\mathcal{M}$ from s. Nodes of $ctree(\mathcal{M}, s)$ are finite sequences over S and $ctree(\mathcal{M}, s)$ is the smallest tree satisfying the following.

1. The root is $\langle s \rangle$.
2. If $\langle s, \ldots, w \rangle$ is a node and $w \rightarrow v$, then $\langle s, \ldots, w, v \rangle$ is a node whose parent is $\langle s, \ldots, w \rangle$.

Our next definition is used to construct the ranking function appearing in the definition of RWFSK.

Definition 10 (ranktCt)**.** Given a transition system $\mathcal{M} = \langle S, \rightarrow, L \rangle$, $s, w \in S$ and an SKS B on $\mathcal{M}$, $ranktCt(\mathcal{M}, B, s, w)$ is the empty tree if $\neg(sBw)$, otherwise $ranktCt(\mathcal{M}, B, s, w)$ is the largest subtree of $ctree(\mathcal{M}, s)$ rooted at s such that

for any non-root node $\langle s, \ldots, x\rangle$ of the tree $\mathit{ranktCt}(\mathcal{M}, B, s, w)$, we have that xBw and $\langle \forall v : w \rightarrow^{+} v : \neg(xBv)\rangle$.

A basic property of our construction is the finiteness of paths.

Lemma 13. Let B be an SKS on $\mathcal{M}$. Every path of $\mathit{ranktCt}(\mathcal{M}, B, s, w)$ is finite.

Proof: The proof is by contradiction, so we start by assuming that there exists an infinite path, σ, in $\mathit{ranktCt}(\mathcal{M}, B, s, w)$. The path has to start at s. Let $\sigma(1) = u$ and let σ' be the suffix of σ starting at u. Since all states in σ' appear in $\mathit{ranktCt}(\mathcal{M}, B, s, w)$, by construction for any state x in σ', xBw and $\neg\langle \exists v : w \rightarrow^{+} v : xBv\rangle$. Since B is an SKS and uBw, there is a fullpath δ starting at w and $\pi, \xi \in INC$ such that $\mathit{scorr}(B, \sigma', \pi, \delta, \xi))$ holds. In particular, $\sigma(\pi.1)B\delta(\xi.1)$, and we have our contradiction because $\sigma(\pi.1)$ is in $\mathit{ranktCt}(\mathcal{M}, B, s, w)$ and can't be related by B to states reachable from w.

□

A node in a computation tree is a finite sequence of states. This induces a natural partial order *"is an initial segment of"* on nodes. A computation tree rooted at $s \in S$ tree has a root node $\langle s\rangle$ and is the maximum node. In the case of finite-path trees we can also refer to *minimal* nodes. Furthermore, we can use ordinal numbers to classify finite-path trees. Given Lemma 13, we define a function, *size*, that given a non-empty finite-path tree, say T, maps a node in the tree to an ordinal number. The ordinal assigned to a node $x \in T$ is defined as follows: if x is a leaf node in T, then $\mathit{size}(T, x) = 0$, else $\mathit{size}(T, x) = (\cup_{c \in \mathit{children}(T,x)} \mathit{size}(T, c)) + 1$, where $\mathit{children}(T, x)$ returns a subset of nodes that are immediate successors of x in T. We let the size of a computation tree to be the size of its root.

Note that we are using the standard set-theoretic representation for ordinal numbers, where an ordinal number is defined to be the set of ordinal numbers below it (*e.g.*, $2 = \{0, 1\}$), which also explains the notation union of ordinal numbers.

We define the size of a $ranktCt(\mathcal{M}, B, s, w)$ to be $size(ranktCt(\mathcal{M}, B, s, w), \langle s \rangle)$. We use $\preceq$ to compare ordinal numbers (and therefore cardinal numbers as well).

Lemma 14 ([11]). If $|S| \preceq \kappa$, where $\omega \preceq \kappa$ then for all $s, w \in S$, $size(ranktCt(\mathcal{M}, B, s, w))$ is an ordinal of cardinality $\preceq \kappa$.

Lemma 14 shows that we can use the cardinal $max(|S|^+, \omega)$ as the domain of our well-founded function in RWFSK2: either ω if the state space is finite, or $|S|^+$, the cardinal successor of the size of the state space otherwise.

Lemma 15. If $sBw, s \to u, \langle s, u \rangle \in ranktCt(\mathcal{M}, B, s, w)$ then $size(ranktCt(\mathcal{M}, B, u, w)) \prec size(ranktCt(\mathcal{M}, B, s, w))$.

Proof: Since $\langle s, u \rangle \in ranktCt(\mathcal{M}, B, s, w)$, from Lemma 13 and the definition of *size*, it follows that $size(ranktCt(\mathcal{M}, B, u, w)) \prec size(ranktCt(\mathcal{M}, B, s, w))$.

□

We are now ready to prove that SKS implies RWFSK.

Proof: RWFSK1 follows directly from SKS1. To show that RWFSK2 holds, let W be $max(|S|^+, \omega)$ and let $rankt(a, b)$ be $size(ranktCt(\mathcal{M}, B, a, b))$. Given $s, u, w \in S$ such that $s \to u$ and sBw, we show that either RWFSK2(a) or RWFSK2(b) holds.

There are two cases. First, suppose that $\langle \exists v : w \to^+ v : uBv \rangle$ holds, then RWFSK2(b) holds. If not, then $\langle \forall v : w \to^+ v : \neg(uBv) \rangle$, but B is an SKS so let σ be a fullpath starting at s and $\sigma.1 = u$. Then there is a fullpath δ such that $fp.\delta.w$ and $smatch(B, \sigma, \delta)$. Hence, there exists $\pi, \xi \in INC$ such that $scorr(B, \sigma, \pi, \delta, \xi)$. By the definition of *corr*, we have that $uB\delta(\xi.i)$ for some i, but i cannot be greater than 0 because then uBx for some x reachable from w, violating the assumptions of the case we are considering. So, $i = 0$, *i.e.*, uBw. By Lemma 15, $rankt(u, w) = size(ranktCt(\mathcal{M}, B, u, w)) \prec size(ranktCt(\mathcal{M}, B, s, w)) = rankt(s, w)$.

□

4.4.2 Well-founded Skipping Simulation

We next introduce the notion of well-founded skipping simulation (WFSK). The intuition is that for any pair of states s, w that are related and a state u such that $s \to u$, there are four cases to consider (Figure 4.4.2): (a) either we can match the move from s to u in a single step, *i.e.*, there is a v such that $w \to v$ and u is related to v, or (b) a move from s to u is a stuttering step and u is related to w, or (c) w can make a move to v that is a stuttering step and v is related to s, or (d) a move from s to u skips one or more steps starting from w.

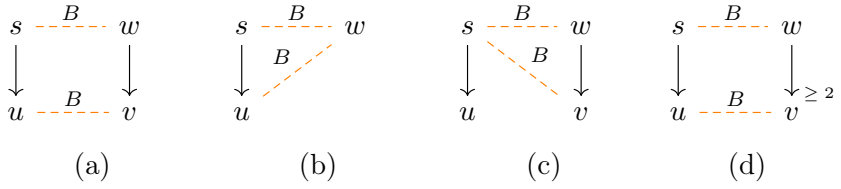

Figure 4.4.2: Well-founded skipping simulation (a solid line with arrow indicates the transition relation, a dashed orange line indicates that states are related by B)

Definition 11 (Well-founded Skipping (WFSK) [7])**.** $B \subseteq S \times S$ is a well-founded skipping relation on TS $\mathcal{M} = \langle S, \to, L \rangle$ iff :

(WFSK1) $\langle \forall s, w \in S : sBw : L.s = L.w \rangle$

(WFSK2) There exist functions, $rankt : S \times S \to W$, $rankl : S \times S \times S \to \omega$, such that $\langle W, \prec \rangle$ is well-founded and

$\langle \forall s,u, w \in S : s \to u \wedge sBw :$

(a) $\langle \exists v : w \to v : uBv \rangle \vee$

(b) $(uBw \wedge rankt(u, w) \prec rankt(s, w)) \vee$

(c) $\langle \exists v : w \to v : sBv \wedge rankl(v, s, u) < rankl(w, s, u) \rangle \vee$

(d) $\langle \exists v : w \to^{\geq 2} v : uBv \rangle \rangle$

Observe that, reasoning about stuttering both on left (WFSK2b) and right

(WFSK2c) is now local. Consider a scenario where we are verifying a system that has a bound—determined early early in the design—on the number of skipping steps possible but no such bound can be a priori determined for stuttering. If we use RWFSK, notice that RWSFK2b forces us to deal with stuttering and skipping steps in the same way, while WFSK enables us to distinguish the stuttering and skipping and locally deal with any amount of stuttering. However, the condition WFSK2d that accounts for skipping on the right still in general requires reachability analysis.

The following lemma asserts that WFSK and RWFSK are equivalent and therefore WFSK is also a sound and complete proof method to reason about skipping simulation.

Lemma 16 ([7]). Let $\mathcal{M}$ be a transition system. B is a WFSK on $\mathcal{M}$ *iff* B is an RWFSK on $\mathcal{M}$.

Proof: ($\Leftarrow$ direction): This direction is straightforward.
($\Rightarrow$ direction): Let $s, u, w \in S$, $s \rightarrow u$, and sBw. RWFSK1 follows directly from WFSK1.

Next we show that RWFSK2 holds. The key insight is that if we remove WFSK2c, then WFSK and RWFSK definitions are semantically equivalent. Therefore, it must be that WFSK2c is redundant. To see this, note that it allows finite stuttering on the right (because *rankl* is well-founded), but finite stuttering is just a special case of more permissive skipping allowed by WFSK2a,d.

Let $s, u, w \in S$, $s \rightarrow u$, and sBw. If WFSK2a or WFSK2d holds then RWFSK2b holds. If WFSK2b holds, then RWFSK2a holds. So, what remains is to assume that WFSK2c holds and neither of WFSK2a, WFSK2b, or WFSK2d hold. From this we will derive a contradiction.

Let δ be a path starting at w, such that only WFSK2c holds between $s, u, \delta.i$. There are non-empty paths that satisfy this condition, *e.g.*, let $\delta = \langle w \rangle$. In addition,

any such path must be finite. If not, then for any adjacent pair of states in δ, say $\delta.k$ and $\delta(k+1)$, $rankl(\delta(k+1), s, u) < rankl(\delta.k, s, u)$, which contradicts the well-foundedness of *rankl*. We also have that for every $k > 0$, $u \not\mathcal{B} \delta.k$; otherwise WFSK2a or WFSK2d holds. Now, let δ be a maximal path satisfying the above condition, *i.e.*, every extension of δ violates the condition. Let x be the last state in δ. We know that sBx and only WFSK2c holds between s, u, x, so let y be a witness for WFSK2c, which means that sBy and one of WFSK2a,b, or d holds between s, u, y. WFSK2b can't hold because then we would have uBy (which would mean WFSK2a holds between s, u, x). So, one of WFSK2a,d has to hold, but that gives us a path from x to some state v such that uBv. The contradiction is that v is also reachable from w, so WFSK2a or WFSK2d held between s, u, w.

□

4.4.3 Reduced Local Well-founded Skipping Simulation

Next we introduce the notion of reduced local well-founded skipping simulation (RLWFSK). Recall the event processing systems AEPS and tEPS described in Section(4.1). When no events are scheduled to execute at a give time, say t, tEPS increments time to the earliest time in future, say $k > t$, at which an event is scheduled to execute. Consider the scenario $k \geq t + 1$. Since AEPS increments time by at most 1, in this scenario tEPS skips multiple states of AEPS. Moreover, execution of an event may add a new event to be executed at an arbitrary time in future. Therefore, one cannot a priori determine an upper-bound on k. Using WFSK to analyse correctness of such systems would require unbounded reachability analysis, a task often difficult for automated verification tools. In contrast, RLWFSK requires reasoning about states and their successors and can be used to effectively analyse systems that exhibit finite unbounded skipping.

Figure 4.4.3: Reduced local well-founded skipping simulation (a solid line with arrow indicates the transition relation, a dashed orange line indicates that states are related by B and a solid blue line indicate the states are related by $\mathcal{O}$)

Definition 12 (Reduced Local Well-founded Skipping (RLWFSK))**.** $B \subseteq S \times S$ is a local well-founded skipping relation on TS $\mathcal{M} = \langle S, \rightarrow, L \rangle$ iff:

(RLWFSK1) $\langle \forall s, w \in S : sBw : L.s = L.w \rangle$

(RLWFSK2) There exist functions, $rankt : S \times S \longrightarrow W$, $rankls : S \times S \longrightarrow \omega$ such that $\langle W, \prec \rangle$ is well founded, and, a binary relation $\mathcal{O} \subseteq S \times S$ such that

$\langle \forall s, u, w \in S : sBw \ \wedge \ s \rightarrow u :$

(a) $(uBw \ \wedge \ rankt(u, w) \prec rankt(s, w)) \ \vee$

(b) $\langle \exists v : w \rightarrow v : u\mathcal{O}v \rangle \rangle$

and

$\langle \forall x, y \in S : x\mathcal{O}y :$

(c) $xBy \ \vee$

(d) $\langle \exists z : y \rightarrow z : x\mathcal{O}z \ \wedge \ rankls(z, x) < rankls(y, x) \rangle \rangle$

As is the case with RWFSK and WFSK, RLWFSK also characterizes skipping simulation, *i.e.*, it is a sound and complete proof method for reasoning about skipping simulation. To prove the completeness, *i.e.*, SKS implies RLWFSK, we first prove that RWFSK implies RLWFSK. Then from Lemma 12, we infer that SKS implies RLWFSK. The soundness of RLWFSK follows from Theorem 19 proved later in the section.

Lemma 17. Let $\mathcal{M}$ be a transition system. If B is an RWFSK on $\mathcal{M}$, then B is an

RLWFSK on $\mathcal{M}$.

Proof: Let B be an RWFSK on $\mathcal{M}$. RLWFSK1 follows directly from RWFSK1.

To show that RLWFSK2 holds, we use any *rankt* function that can be used to show that RWFSK2 holds. We define $\mathcal{O}$ as follows.

$$\mathcal{O} = \{(u, v) : \langle \exists z : v \rightarrow^{+} z : uBz \rangle\}$$

We define $rankls(u, v)$ to be the minimal length of a $\mathcal{M}$-segment that starts at v and ends at a state, say z, such that uBz, if such a segment exists and 0 otherwise.

Let $s, u, w \in S$, sBw and $s \rightarrow u$. If RWFSK2a holds between s, u, and w, then RLWFSK2a also holds. Next, suppose that RWFSK2a does not hold but RWFSK2b holds, *i.e.*, there is an $\mathcal{M}$-segment $\langle w, a, \ldots, v \rangle$ such that uBv; therefore, $u\mathcal{O}a$ and RLWFSK2b holds.

To finish the proof, we show that $\mathcal{O}$ and $rankls$ satisfy the constraints imposed by the second conjunct in RLWFSK2. Let $x, y \in S$, $x\mathcal{O}y$ and $x \not\mathcal{B}\ y$. From the definition of $\mathcal{O}$, we have that there is an $\mathcal{M}$-segment from y to a state related to x by B; let $\vec{y}$ be such a segment of minimal length. From the definition of $rankls$, we have $rankls(y, x) = |\vec{y}|$. Observe that y cannot be the last state of $\vec{y}$ and $|\vec{y}| \geq 2$. This is because the last state in $\vec{y}$ must be related to x by B, but from the assumption we know that $x \not\mathcal{B}\ y$. Let y' be a successor of y in $\vec{y}$. Clearly, $x\mathcal{O}y'$; therefore, $rankls(y', x) < |\vec{y}| - 1$, since the length of a minimal $\mathcal{M}$-segment from y' to a state related to x by B, must be less or equal to $|\vec{y}| - 1$.

□

Lemma 18. Let $\mathcal{M}$ be a transition system. If B is an SKS on $\mathcal{M}$, then B is an RLWFSK on $\mathcal{M}$.

Proof: Follows directly from Lemma 17 and Lemma 12.

□

4.4.4 Local well-founded Skipping Simulation

Reduced local well-founded skipping simulation introduced above requires only local reasoning and therefore is amenable for mechanical reasoning. However, note that RLWFSK (like RWFSK), does not differentiate between skipping and stuttering on the right. Such a differentiation can often be useful in practice. To redress this, we define local well-founded skipping simulation (LWFSK), a characterization of skipping simulation that separates reasoning about skipping from reasoning about stuttering on the right.

Definition 13 (Local Well-founded Skipping (LWFSK))**.** $B \subseteq S \times S$ is a local well-founded skipping relation on TS $\mathcal{M} = \langle S, \rightarrow, L \rangle$ iff:

(LWFSK1) $\langle \forall s, w \in S : sBw : L.s = L.w \rangle$

(LWFSK2) There exist functions, $rankt : S \times S \longrightarrow W$, $rankl : S \times S \times S \longrightarrow \omega$, and $rankls : S \times S \longrightarrow \omega$ such that $\langle W, \prec \rangle$ is well founded, and, a binary relation $\mathcal{O} \subseteq S \times S$ such that

$\langle \forall s, u, w \in S : sBw \wedge s \rightarrow u :$

(a) $\langle \exists v : w \rightarrow v : uBv \rangle \vee$

(b) $(uBw \wedge rankt(u, w) \prec rankt(s, w)) \vee$

(c) $\langle \exists v : w \rightarrow v : sBv \wedge rankl(v, s, u) < rankl(w, s, u) \rangle \vee$

(d) $\langle \exists v : w \rightarrow v : u\mathcal{O}v \rangle \rangle$

and

$\langle \forall x, y \in S : x\mathcal{O}y :$

(e) $xBy \vee$

(f) $\langle \exists z : y \rightarrow z : x\mathcal{O}z \wedge rankls(z, x) < rankls(y, x) \rangle \rangle$

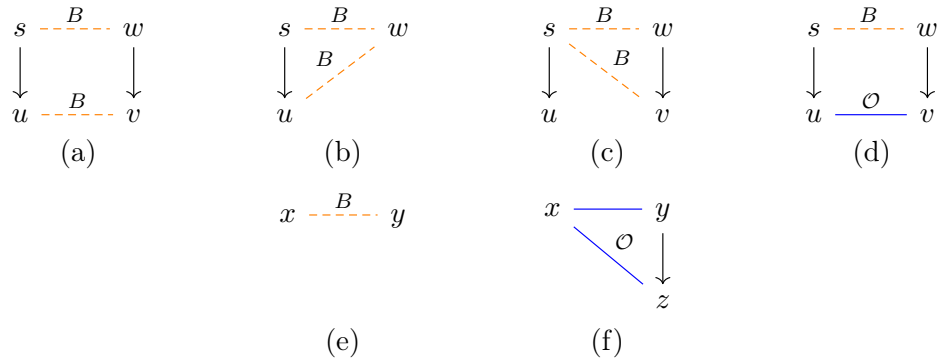

Figure 4.4.4: Local well-founded skipping simulation (a solid line with arrow indicates the transition relation, a dashed orange line indicates that states are related by B and a solid blue line indicate the states are related by $\mathcal{O}$)

As was the case with RLWFSK, to prove that a relation is an LWFSK, reasoning about single steps of the transition system suffices. However, LWFSK2c accounts for stuttering on the right, and LWFSK2d along with LWFSK2e and LWFSK2f account for skipping on the right. Also observe that states related by $\mathcal{O}$ are not required to be labeled identically.

We conclude by showing that the four notions, RWFSK, WFSK, RLWFSK, and LWFSK, introduced in this section are equivalent and completely characterize skipping simulation.

Theorem 19. Let $\mathcal{M} = \langle S, \to, L \rangle$ be a transition system and $B \subseteq S \times S$. The following statements are equivalent:

(i) B is an SKS on $\mathcal{M}$;

(ii) B is an RLWFSK on $\mathcal{M}$;

(iii) B is an LWFSK on $\mathcal{M}$;

(iv) B is a WFSK on $\mathcal{M}$;

(v) B is an RWFSK on $\mathcal{M}$;

Proof: That (ii) implies (iii) follows from the simple observation that RL-WFSK2 implies LWFSK2. That (iv) implies (v) follows from Lemma 16, that (v) implies (i) follows from Lemma 11, and that (i) implies (ii) follows from Lemma 18. To complete the proof, we prove that (iii) implies (iv) in Lemma 20,

Lemma 20. Let $\mathcal{M}$ be a transition system. If B is an LWFSK on $\mathcal{M}$, then B is a WFSK on $\mathcal{M}$.

Proof: Let B be an LWFSK on $\mathcal{M}$. WFSK1 follows directly from LWFSK1.

Let *rankt*, *rankl*, and *rankls* be functions, and $\mathcal{O}$ be a binary relation such that LWFSK2 holds. To show that WFSK2 holds, we use the same *rankt* and *rankl* functions and let $s, u, w \in S$ and $s \rightarrow u$ and sBw. LWFSK2a, LWFSK2b and LWFSK2c are equivalent to WFSK2a, WFSK2b and WFSK2c, respectively, so we show that if only LWFSK2d holds, then WFSK2d holds. Since LWFSK2d holds, there is a successor v of w such that $u\mathcal{O}v$. Since $u\mathcal{O}v$ holds, either LWFSK2e or LWFSK2f must hold between u and v. However, since LWFSK2a does not hold, LWFSK2e cannot hold and LWFSK2f must hold, *i.e.*, there exists a successor v' of v such that $u\mathcal{O}v' \wedge \mathit{rankls}(v', u) < \mathit{rankls}(v, u)$. So, we need a path of at least 2 steps from w to satisfy the universally quantified constraint on $\mathcal{O}$. Let us consider an arbitrary path, δ, such that $\delta.0 = w$, $\delta.1 = v$, $\delta.2 = v'$, $u\mathcal{O}\delta.i$, LWFSK2e does not hold between u and $\delta.i$ for $i \geq 1$, and $\mathit{rankls}(\delta.(i+1), u) < \mathit{rankls}(\delta.i, u)$. Notice that any such path must be finite because rankls is well founded. Hence, δ is a finite path and there exists a $k \geq 2$ such that LWFSK2e holds between u and $\delta.k$. Therefore, WFSK2d holds, *i.e.*, there is a state in δ reachable from w in two or more steps which is related to u by B.

□

Theorem 19 shows that under no restrictions on systems under consideration, RLWFSK (Definition 12) and LWFSK (Definition 13) are complete proof methods

and require only local reasoning. Thus, to prove that a concrete system is a skipping refinement of an abstract system, one can always prove it using local reasoning about states and their successors and do not require global reasoning about infinite paths. We discuss this in more detail in Chapter 7.

4.5 Summary

In this chapter, we introduced a new notion of skipping simulation. We showed that skipping simulation enjoys several useful algebraic properties and developed a compositional theory of skipping refinement that aligns with a stepwise refinement verification methodology [14, 3]. Finally, we developed four alternative characterizations of skipping simulation that are amenable for mechanised reasoning using formal-methods tools.

In Chapter 6, we present three case studies that highlight the applicability of the proof methods: a JVM-inspired stack machine, a simple memory controller and a scalar to vector compiler transformation. Our experimental results demonstrate that current model-checking and automated theorem proving tools have difficulty analysing these systems using existing notions of correctness, but they can effectively analyse the systems using skipping refinement.

Chapter 5

Reconciling Simulation

In this chapter, we first define the notion of reconciling simulation. Then we study its algebraic properties and develop a theory of reconciling refinement. Finally, we develop sound and complete proof methods that are amenable for automated reasoning about reconciling refinement.

5.1 Running Example (Continued)

Consider event processing systems AEPS and MPEPS described in Chapter 4. If no event is scheduled to execute at the current time, say t, AEPS increments time to $t + 1$ while MPEPS updates time to the earliest time in future at which an event is scheduled to execute. Otherwise, AEPS picks an event while MPEPS picks a non-empty subset of events from the set of events scheduled to execute at t. Earlier, we had argued that MPEPS is a skipping refinement of AEPS. Now suppose the set of events scheduled to execute at any given time is additionally constrained by a partial order. Furthermore, suppose that for any order of execution of events that respects the partial order, the EPS systems reach the same state. Observe that though AEPS may execute events in different order than MPEPS and AEPS may

execute at most one event in a single step, there is a sense in which AEPS refines MPEPS. If MPEPS and AEPS are in related states, then the two systems *reconcile* after all the events scheduled at a given time have been executed. However, a step of AEPS neither corresponds to stuttering nor corresponds to skipping with respect to a step in MPEPS. Therefore, skipping refinement (and stuttering refinement) is not an appropriate notion of refinement to directly analyse the correctness of AEPS. In this chapter, we define reconciling refinement, a new notion of refinement that can be used to directly analyse correctness of such reactive systems. Reconciling refinement is strictly weaker than skipping refinement, and extends the domain of applicability of the refinement-based approach to verification of a larger class of reactive systems. We develop the theory of reconciling refinement using an approach that is analogous to one used to develop the theory of skipping refinement (Chapter 4). We first define the notion of reconciling simulation using a single transition system and study its algebraic properties. Then we use the notion of reconciling simulation and a refinement map to define the notion of reconciling refinement, a notion that relates *two* transition systems: an abstract transition system and a concrete transition system. Finally, we develop sound and complete proof methods for reasoning about reconciling refinement that require only local reasoning, and are amenable for mechanised verification.

5.2 Reconciling Simulation

The notion of reconciling simulation is based on *rmatch*, a new notion of matching fullpaths. Informally, we say a fullpath σ rmatches a fullpath δ under the relation B if the fullpaths can be partitioned in to non-empty, finite segments such that the first element in a segment of σ is related to the first element in the corresponding segment of δ. Using the notion of rmatch, reconciling simulation is defined as follows:

a relation B is a reconciling simulation on a transition system $\mathcal{M} = \langle S, \rightarrow, L \rangle$, if for any $s, w \in S$ such that sBw, s and w are labeled identically and any fullpath starting at s can be rmatched by some fullpath starting at w.

Definition 14 (rmatch)**.** Let $\mathcal{M} = \langle S, \rightarrow, L \rangle$ be a transition system, σ, δ be fullpaths in $\mathcal{M}$. For $\pi, \xi \in \mathit{INC}$ and binary relation $B \subseteq S \times S$, we define

$\mathit{rcorr}(B, \sigma, \pi, \delta, \xi) \equiv \langle \forall i \in \omega :: \sigma(\pi.i) B \delta(\xi.i) \rangle$ *and*

$\mathit{rmatch}(B, \sigma, \delta) \equiv \langle \exists \pi, \xi \in \mathit{INC} :: \mathit{rcorr}(B, \sigma, \pi, \delta, \xi) \rangle$.

Definition 15 (Reconciling Simulation)**.** $B \subseteq S \times S$ is a reconciling simulation (RES) on a TS $\mathcal{M} = \langle S, \rightarrow, L \rangle$ *iff* for all s, w such that sBw, both of the following hold.

(RES1) $L.s = L.w$

(RES2) $\langle \forall \sigma : \mathit{fp}.\sigma.s : \langle \exists \delta : \mathit{fp}.\delta.w : \mathit{rmatch}(B, \sigma, \delta) \rangle \rangle$

Notice that the definition reconciling simulation differs from skipping simulation only in the notion of matching fullpaths used to define them. Both require that matching fullpaths σ and δ be partitioned into finite non-empty segments. But they differ in what states in a segment of σ relate to the first state in the corresponding segment of δ: while SKS requires that all states in a segment of σ are related to the first state in the corresponding segment of δ, RES only requires that first state in a segment of σ is related to the first state in the corresponding segment of δ. Clearly, RES is strictly weaker than SKS.

Theorem 21. If B is an SKS on $\mathcal{M}$, then B is an RES on $\mathcal{M}$.

Proof: Follows directly from the definitions of SKS and RES.

□

5.3 Algebraic Properties

We next study the algebraic properties of reconciling simulation. RES is closed under arbitrary union. However, it is not closed under relational composition.

Lemma 22. Let $\mathcal{M}$ be a transition system and $\mathcal{C}$ be a set of RES's on $\mathcal{M}$, then $G = \langle \cup B \colon B \in \mathcal{C} \colon B \rangle$ is an RES on $\mathcal{M}$.

Proof: Let $s, w \in S$ and sGw. We show that RES1 and RES2 holds for G. Since $G = \langle \cup B : B \in \mathcal{C} : B \rangle$, there is an RES $B \in \mathcal{C}$ on $\mathcal{M}$ such that sBw. Since B is an RES on $\mathcal{M}$, we have that $L.s = L.w$. Hence, RES1 holds for G. Next RES2 also holds for B, *i.e.*, for any fullpath σ starting at s, there is a fullpath δ starting at w such that $rmatch(B, \sigma, \delta)$ holds. From Definition 14 of $rmatch$, there exists $\pi, \xi \in INC$ such that $\langle \forall i \in \omega :: \sigma(\pi.i)\ B\ \delta(\xi.i) \rangle$. Since $B \subseteq G$, we have that $\langle \forall i \in \omega :: \sigma(\pi.i)\ G\ \delta(\xi.i) \rangle$. Hence, from Definition 14, $rmatch(G, \sigma, \delta)$ holds, *i.e.*, RES2 holds for G.

□

Corollary 23. For any transition system $\mathcal{M}$, there is a greatest RES on $\mathcal{M}$.

Proof: Let $\mathcal{C}$ be the set of all RES's on $\mathcal{M}$ and $G = \langle \cup B : B \in \mathcal{C} : B \rangle$. By construction, G is the greatest and from Lemma 22, G is an RES on $\mathcal{M}$.

□

The following lemma shows that RES is not closed under intersection and negation, as was the case with stuttering simulation [10] and skipping simulation 4.2.

Lemma 24. RESs are not closed under negation or intersection.

Proof: The example TSs used in the proof in Lemma 3 provide counterexamples.

□

However, unlike stuttering simulations and skipping simulations, RESs are not closed under relational composition.

Lemma 25. RESs are not closed under relational composition.

Proof: A counterexample appears in Figure 5.3.1.

□

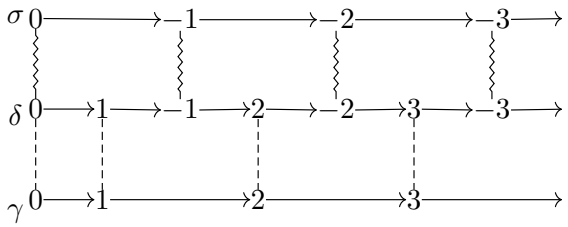

Figure 5.3.1: An example showing that RESs are not closed under relational composition. Consider a TS $\mathcal{M}$ that consists of three fullpaths σ, δ, and γ. The transition relation for the TS is represented by $\rightarrow$. The labeling function is defined as follows: $L(\sigma.0) = L(\delta.0) = L(\gamma.0) = 0$. And for all $i \geq 1$, $L(\sigma.i) = -i$, $L(\delta.i) = (i-1)/2 + 1$ if i is odd and $L(\delta.i) = -i/2$ if i is even, and $L(\gamma.i) = i$. The two reconciling simulations, B and B', are denoted by squiggly and dashed lines. We have that $(\sigma.0)B(\delta.0)$ and $(\delta.0)B'(\gamma.0)$. However, $(\sigma.0)B; B'(\gamma.0)$ does not hold because they have no rmatching successors. In fact there is no RES that relates $\sigma.0$ with $\gamma.0$.

5.4 Reconciling refinement

In this section, we use the notion of reconciling simulation to define the notion of reconciling refinement, a notion that relates *two* transition systems: an *abstract* transition system and a *concrete* transition system. The notion is parameterized by a refinement map, a function that maps a concrete state to an abstract state. Informally, if a concrete system, say $\mathcal{C}$, is a reconciling refinement of an abstract system,

say $\mathcal{A}$, under a refinement map r, then observable behaviors of $\mathcal{C}$ are observable behaviors of $\mathcal{A}$ up to reconciliation (reconciliation subsumes skipping and stuttering). Recall that a refinement map along with the labeling function determines what is observable at a concrete state.

Note that we do not place any restriction on the state space sizes and the branching factor of the transition relation of the abstract and the concrete systems, and both can be of arbitrary infinite cardinalities. Thus the theory of reconciling refinement, like the theory of skipping refinement developed in Chapter 4, and sound and complete proof methods for reasoning about reconciling refinement (developed in the Section 5.5) provide a reasoning framework that can be used to analyse a large class of reactive systems.

Definition 16 (Reconciling Refinement)**.** Let $\mathcal{M}_{\mathrm{A}} = \langle S_A, \xrightarrow{A}, L_A \rangle$ and $\mathcal{M}_{\mathrm{C}} = \langle S_C, \xrightarrow{C}, L_C \rangle$ be transition systems and let $r \colon S_C \to S_A$ be a refinement map. We say $\mathcal{M}_C$ is a *reconciling refinement* of $\mathcal{M}_A$ with respect to r, written $\mathcal{M}_C \trianglelefteq_r \mathcal{M}_A$, if there exists a binary relation B such that all of the following hold.

1. $\langle \forall s \in S_C :: sB(r.s) \rangle$ *and*

2. B is an RES on $\langle S_C \uplus S_A, \xrightarrow{C} \uplus \xrightarrow{A}, \mathcal{L} \rangle$ where $\mathcal{L}.s = L_A(s)$ for $s \in S_A$, and $\mathcal{L}.s = L_A(r.s)$ for $s \in S_C$.

Observe that our definition of reconciling refinement is quite general, *e.g.*, the state space and the branching factor of the transition relation of the systems can be of arbitrary infinite cardinality and there are no restrictions on the choice of refinement map. In the next section, we develop sound and complete proof methods that are amenable for mechanised reasoning. This provides a general theory of refinement and a reasoning framework that is applicable for analyzing a large class of optimized reactive systems. However, in general reconciling refinement is not compositional

and therefore, unlike skipping refinement, reconciling refinement does not align with the stepwise refinement verification methodology.

5.5 Mechanised reasoning

We turn our attention to the mechanical verification of correctness of systems using reconciling refinement. If we use Definition 16 to prove that transition system $\mathcal{M}_C$ is a reconciling refinement of transition system $\mathcal{M}_A$ with respect to a refinement map r, we must show that for any fullpath in $\mathcal{M}_C$ there is an rmatching fullpath in $\mathcal{M}_A$. However, reasoning about formulas with nested quantifiers over infinite sequences using formal-methods tools tends to be difficult, *e.g.*, SMT solvers and model checkers either do not allow or do not fully support quantifiers and the manual effort required to prove such theorems with interactive theorem provers can be quite high. To redress this situation, we introduce alternative characterizations of reconciling simulation that are amenable for mechanised reasoning.

5.5.1 Reduced Well-founded Reconciling Simulation

As a first step, we introduce the notion of *reduced well-founded reconciling simulation* (RWRS). It localizes reasoning about reconciliation on the left using an additional binary relation over states and a rank function. However, reasoning about reconciliation on the right still requires reachability analysis and in general is not local. Nevertheless, RWRS is a useful proof method in scenarios where number of steps required to reconcile with a state on the right can be bounded by a constant.

Definition 17 (Reduced Well-founded Reconciling Simulation (RWRS))**.** $B \subseteq S \times S$ is a reduced well-founded reconciling relation on a TS $\mathcal{M} = \langle S, \rightarrow, L \rangle$ iff:
(RWRS1) $\langle \forall s, w \in S : sBw : L.s = L.w \rangle$

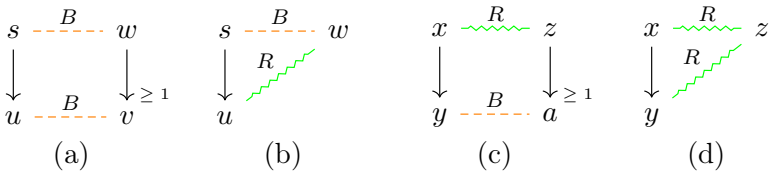

Figure 5.5.1: Reduced well-founded reconciling simulation (a solid line with an arrow indicates the transition relation, a dashed orange line indicates that states are related by B and a squiggly green line indicates that states are related by R).

(RWRS2) There exist a function $\mathit{rankts} \colon S \times S \longrightarrow W$ such that $\langle W, \prec \rangle$ is well founded and a binary relation $R \subseteq S \times S$ such that

$\langle \forall s, u, w \in S : sBw \wedge s \to u :$

(a) $\langle \exists v \colon w \to^{+} v \colon uBv \rangle \vee$

(b) $uRw \rangle$

and

$\langle \forall x, y, z \in S : xRz \wedge x \to y :$

(c) $\langle \exists a : z \to^{+} a : yBa \rangle \vee$

(d) $yRz \wedge \mathit{rankts}(y, z) \prec \mathit{rankts}(x, z) \rangle$

RWRS characterizes reconciling simulation, *i.e.*, it is a sound and complete proof method for reconciling simulation. The soundness of RWRS, *i.e.*, RWRS implies RES, is proved later in Theorem 33. Here we prove the completeness, *i.e.*, RES implies RWRS. Towards this, we first introduce some definitions and lemmas.

Definition 18 (ranktsCt)**.** Let $\mathcal{M} = \langle S, \to, L \rangle$ be a transition system and B be an RES on $\mathcal{M}$. Given $x, z \in S$, $\mathit{ranktsCt}(\mathcal{M}, B, x, z)$ is the empty tree if $\neg\langle \exists s : s \to^{+} x : sBz \rangle$, otherwise $\mathit{ranktsCt}(\mathcal{M}, B, x, z)$ is the largest subtree of $\mathit{ctree}(\mathcal{M}, x)$ such that for any node $\langle x, \ldots, y \rangle$ in $\mathit{ranktsCt}(\mathcal{M}, B, x, z)$, we have that $\langle \forall a : z \to^{+} a : \neg(yBa) \rangle$.

Remark 2. Observe that the computation tree $\mathit{ranktsCt}(\mathcal{M}, B, x, z)$ does not depend

on the choice of $s \in S$ that can reach x and is related by B to z.

A basic property of our construction is the finiteness of paths in *ranktsCt*.

Lemma 26. Let $\mathcal{M} = \langle S, \rightarrow, L \rangle$ be a transition system, B be an RES on $\mathcal{M}$, and $x, z \in S$. Every path of $\mathit{ranktsCt}(\mathcal{M}, B, x, z)$ is finite.

Proof: We consider two cases. First, let $\neg\langle \exists s : s \rightarrow^+ x : sBz \rangle$. Then $\mathit{ranktsCt}(\mathcal{M}, B, x, z)$ is empty and the conclusion trivially holds. Second, let $s \in S$ such that $s \rightarrow^+ x$ and sBz. The proof for this case is by contradiction. We start by assuming that there exists an infinite path σ in $\mathit{ranktsCt}(\mathcal{M}, B, x, z)$. From any such path we can construct a fullpath, say σ, that starts at s and $\sigma.i = x$ for some positive integer i. Since sBz, and B is an RES on $\mathcal{M}$, from Definition 15, there is a fullpath δ starting at z such that $\mathit{rmatch}(B, \sigma, \delta)$ holds. From Definition 14, there is a $k > i$ and a state in δ that is related by B to $\sigma.k$. However, by assumption and from Definition 18 of *ranktsCt*, we have for all $j \geq i$, $\langle x = \sigma.i, \ldots, \sigma.j \rangle$ is a node in $\mathit{ranktsCt}(\mathcal{M}, B, x, z)$ and z cannot reach a state that is related by B to $\sigma.j$. In particular, $\langle x = \sigma.i, \ldots, \sigma.k \rangle$ is a node in $\mathit{ranktsCt}(\mathcal{M}, B, x, z)$ and z cannot reach a state that is related by B to $\sigma.k$. We have our contradiction.

□

Given Lemma 26, we define a function *size*, that given a $\mathit{ctree}(\mathcal{M}, s)$, say T, all of whose paths are finite, assigns an ordinal to T. The ordinal assigned to a node $x \in T$ is defined as follows: if x is a leaf node in T then $\mathit{size}(T, x) = 0$, else $\mathit{size}(T, x) = (\bigcup_{c \in \mathit{children}(T,x)} \mathit{size}(T, c)) + 1$, where $\mathit{children}(T, x)$ returns a subset of nodes that are immediate successor of x in T. The *size* of a computation tree is the *size* of its root.

Lemma 27 ([11]). If $|S| \preceq \kappa$, where $\omega \preceq \kappa$ then for all $s, w \in S$, $\mathit{size}(\mathit{ranktsCt}(\mathcal{M}, s, w))$ is an ordinal of cardinality $\preceq \kappa$.

We use the cardinal $max(|S|^+, \omega)$, where $|S|^+$ is the cardinal successor of the size of the state space, as the well-founded domain for *rankts* in Definition 17. Next we define the binary relation F_R such that the second universal quantifier in RWRS2 of Definition 17 holds.

Definition 19 (F_R)**.** Let $\mathcal{M} = \langle S, \rightarrow, L \rangle$ be a transition system and B be an RES on $\mathcal{M}$. We define $F_R(\mathcal{M}, B)$ as the set of pairs $(x, z) \in S \times S$ such that $ranktsCt(\mathcal{M}, B, x, z)$ is non-empty.

Lemma 28. Let $\mathcal{M} = \langle S, \rightarrow, L \rangle$ be a transition system, B be an RES on $\mathcal{M}$, $x, y, z \in S$, $x \rightarrow y$, and $(x, z) \in F_R(\mathcal{M}, B)$. If $\langle \forall a : z \rightarrow^+ a : \neg(yBa) \rangle$ then $(y, z) \in F_R(\mathcal{M}, B)$ and $size(ranktsCt(\mathcal{M}, y, z)) \prec size(ranktsCt(\mathcal{M}, x, z))$.

Proof: To show that $(y, z) \in F_R(\mathcal{M}, B)$, we show that $ranktsCt(\mathcal{M}, B, y, z)$ is non-empty. Towards this observe that there is a $s \in S$ such that $s \rightarrow^+ y \wedge sBz$. This is because by assumption $x \rightarrow y$ and from Definition 19 and Definition 18, $ranktsCt(\mathcal{M}, B, x, z)$ is non-empty and $\langle \exists s : s \rightarrow^+ x \wedge sBz \rangle$. Also, $\langle x, y \rangle \in ranktsCt(\mathcal{M}, B, x, z)$. This is because, by assumption, $x \rightarrow y$ and $\langle \forall a : z \rightarrow^+ a : \neg(yBa) \rangle$. Hence, $ranktsCt(\mathcal{M}, B, y, z)$ is non-empty. Also, since $y \in children(x)$ in the tree $ranktsCt(\mathcal{M}, B, x, z)$, from the definition of *size* above, we infer that $size(ranktsCt(\mathcal{M}, y, z)) \prec size(ranktsCt(\mathcal{M}, x, z))$.

□

We are now ready to prove the completeness of RWRS.

Theorem 29 (Completeness)**.** Let $\mathcal{M}$ be a transition system. If B is an RES on $\mathcal{M}$, then B is an RWRS on $\mathcal{M}$.

Proof: Let B be an RES on $\mathcal{M}$. RWRS1 follows directly from RES1. To show that RWRS2 holds, let W be $max(|S|^+, \omega)$. Let $a, b \in S$, $rankts(a, b)$ be $size(ranktsCt(\mathcal{M}, a, b))$, and $R = F_R(\mathcal{M}, B)$ be as defined in Definition 19. From

Lemma 28, we have that R and the rank function *rankts* satisfy the second universal quantifier in RWRS2 of Definition 17.

Next let $s, u, w \in S$, $s \to u$, and sBw. We consider two cases. First, suppose that $\langle \exists v : w \to^+ v : uBv \rangle$ holds, then RWRS2a holds. If not, then $\langle \forall v : w \to^+ v : \neg(uBv) \rangle$. Hence, $ranktsCt(\mathcal{M}, B, u, w)$ is non-empty. This is because B is an RES on $\mathcal{M}$, $s \to u$, and sBw. Finally, from the definition of F_R we have that $(u, w) \in F_R$ and uRw holds, *i.e.*, RWRS2b holds.

□

5.5.2 Well-founded Reconciling Simulation

Next, we introduce the notion of *well-founded reconciling simulation* that requires only local reasoning, *i.e.*, reasoning about states and their successors. Unlike RWRS, it can be used to effectively analyse systems that may take finite but unbounded number of steps to reconcile on the left.

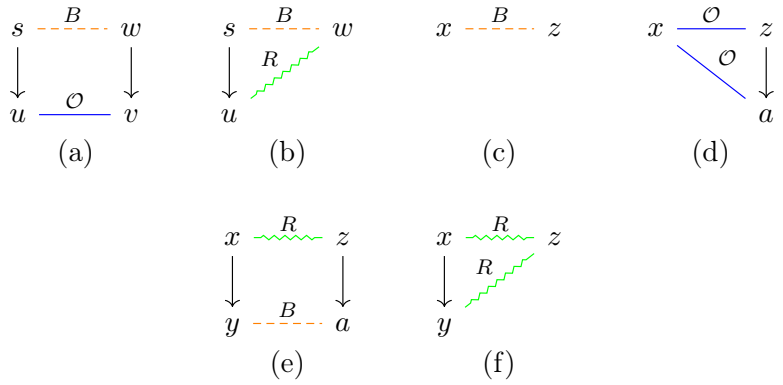

Figure 5.5.2: Well-founded reconciling simulation (a solid line with an arrow indicates the transition relation, a dashed orange line indicates that states are related by B, a solid blue line indicate the states are related by $\mathcal{O}$, and a squiggly green line indicates that states are related by R).

Definition 20 (Well-founded Reconciling Simulation (WRS))**.** $B \subseteq S \times S$ is a well-founded reconciling relation on a TS $\mathcal{M} = \langle S, \rightarrow, L \rangle$ iff:

(WRS1) $\langle \forall s, w \in S \colon sBw \colon L.s = L.w \rangle$

(WRS2) There exist functions, $rankls \colon S \times S \longrightarrow \omega$, and $rankts \colon S \times S \longrightarrow W$ such that $\langle W, \prec \rangle$ is well founded, and binary relations $R, \mathcal{O} \subseteq S \times S$ such that

$\langle \forall s, u, w \in S : sBw \ \wedge \ s \rightarrow u :$

(a) $\langle \exists v \colon w \rightarrow v \colon u \mathcal{O} v \rangle \ \vee$

(b) $uRw \rangle$

and

$\langle \forall x, z \in S : x \mathcal{O} z :$

(c) $xBz \ \vee$

(d) $\langle \exists a : z \rightarrow a : x \mathcal{O} a \ \wedge \ rankls(a, x) < rankls(z, x) \rangle$

and

$\langle \forall x, y, z \in S : xRz \wedge x \rightarrow y :$

(e) $\langle \exists a \colon z \rightarrow a \colon y \mathcal{O} a \rangle \ \vee$

(f) $yRz \ \wedge \ rankts(y, z) \prec rankts(x, z) \rangle$

Intuitively, the condition RWRS2a in Definition 17, that required reachability analysis, is replaced by conditions WRS2a, WRS2c and WRS2d in Definition 20. Similarly, the condition RWRS2c in Definition 17, that required reachability analysis, is replaced by conditions WRS2e, WRS2c and WRS2d in Definition 20. To show that a binary relation is a WRS, we have to provide as witness binary relations R and $\mathcal{O}$, and rank functions *rankts* and *rankls*. $\mathcal{O}$ and *rankls* enable local reasoning about reconciliation on the right and R and *rankts* as was the case with RWRS enable local reasoning about reconciliation on the left. In fact, RWRS and WRS

are equivalent. Here we show that RWRS implies WRS. The other direction, WRS implies RWRS, is inferred from Theorem 33 proved later in the section.

Theorem 30. Let $\mathcal{M}$ be a transition system. If B is an RWRS on $\mathcal{M}$, then B is a WRS on $\mathcal{M}$.

Proof: Let B be an RWRS on $\mathcal{M}$. WRS1 follows directly from RWRS1. Next we have to show that RWRS2 implies WRS2. Let $\langle W, \prec \rangle$ be a well-founded domain, $rankts : S \times S \rightarrow W$ and $R \subseteq S \times S$ such that RWRS2 holds. To show that WRS2 holds, we use the same rank function *rankts* and the binary relation R as in RWRS2. The binary relation $\mathcal{O} \subseteq S \times S$ is defined as follows: $\mathcal{O} = \{(u, v) : \langle \exists v' : v \rightarrow^* v' : uBv' \rangle\}$. The rank function $rankls : S \times S \rightarrow \omega$ is defined as follows: let $u, v \in S$ then $rankls(u, v)$ is the minimal length of an $\mathcal{M}$-segment that starts at v and ends in a state that is related to u by B, if such a segment exists and 0 otherwise.

Let $s, u, w \in S$, $s \rightarrow u$, and sBw. If RWRS2b holds then WRS2b holds. Next suppose that RWRS2b does not hold and RWRS2a holds, *i.e.*, $\langle \exists v : w \rightarrow^+ v : uBv \rangle$. Let $\langle w, v_1, \ldots, v_k = v \rangle$, where $k \geq 1$, be such an $\mathcal{M}$-segment. Hence, from construction of $\mathcal{O}$ we have that $u\mathcal{O}v_1$, and WRS2a holds.

Next let $x, y, z \in S$, xRz, and $x \rightarrow y$. Since R satisfies the second conjunct in RWRS2, one of RWRS2c or RWRS2d must hold. If RWRS2d holds then WRS2f holds. Next suppose RWRS2d does not hold and RWRS2c holds, *i.e.*, $\langle \exists a : z \rightarrow^+ a : yBa \rangle$. Let $\langle z, a_1, \ldots, a_k = a \rangle$, where $k \geq 1$, be such an $\mathcal{M}$-segment. Hence. from the definintion of $\mathcal{O}$, we have that $y\mathcal{O}a_1$, *i.e.*, WRS2e holds.

To finish the proof we show that $\mathcal{O}$ and rank function *rankls*, as defined above, satisfy the constraints imposed by the second conjunct in WRS2. Let $x, y \in S$, $x\mathcal{O}y$, and WRS2c does not hold, *i.e.*, $\neg(xBy)$. From the definition of $\mathcal{O}$, we have that there is an $\mathcal{M}$-segment from y to a state related to x by B; let $\vec{y}$ be such a segment with

the minimal length. From the definition of *rankls*, we have that $rankls(y, x) = |\vec{y}|$. Since the last state in $\vec{y}$ must be related to x by B, and from the assumption $\neg(xBy)$, we have that y cannot be the last state of $\vec{y}$ and $|\vec{y}| > 1$. Let y' be the successor of y in $\vec{y}$. Then from construction of $\mathcal{O}$, we have that $x\mathcal{O}y'$. Observe that the length of the minimal $\mathcal{M}$-segment from y' to a state that is related to x by B must be less or equal to $|\vec{y}| - 1$. Hence $rankls(y', x) = |\vec{y}| - 1 < rankls(y, x)$.

□

5.5.3 Well-founded Reconciling Simulation with Explicit Stuttering

Next we introduce the notion of *well-founded reconciling simulation with explicit stuttering* (WRSS). Like WRS, it only requires reasoning about a state and its successors. However, unlike WRS, it distinguishes between stuttering and reconciling.

Figure 5.5.3 illustrates the conditions in WRSS. The intuition is, for any pair of states s, w which are related by B, a state u such that $s \to u$, there are five cases to consider (a) either we can rmatch the move from s to u with a single step from w, or (b) there is stuttering on the left, or (c) there is stuttering on the right, or (d) there is reconciling on the right, or (e) there is reconciling on the left. Additionally conditions (f) and (g) together ensure that reconciling on the right is finite, and conditions (h) and (i) ensure that reconciling on the left is finite.

Definition 21 (Well-founded Reconciling With Explicit Stuttering (WRSS))**.** $B \subseteq S \times S$ is a well-founded reconciling relation with explicit stuttering on a transition system $\mathcal{M} = \langle S, \to, L\rangle$ iff:

(WRSS1) $\langle \forall s, w \in S : sBw : L.s = L.w\rangle$

(WRSS2) There exist functions, $rankt : S \times S \longrightarrow W$, $rankl : S \times S \times S \longrightarrow \omega$, and $rankls : S \times S \longrightarrow \omega$, and $rankts : S \times S \longrightarrow W$ such that $\langle W, \prec\rangle$ is well founded,

and binary relations $R, \mathcal{O} \subseteq S \times S$ such that

$\langle \forall s, u, w \in S : sBw \wedge s \to u :$

(a) $\langle \exists v \colon w \to v \colon uBv \rangle \vee$

(b) $(uBw \wedge \mathit{rankt}(u, w) \prec \mathit{rankt}(s, w)) \vee$

(c) $\langle \exists v : sBv \wedge \mathit{rankl}(v, s, u) < \mathit{rankl}(w, s, u) \rangle \vee$

(d) $\langle \exists v \colon w \to v \colon u\mathcal{O}v \rangle \vee$

(e) $uRw \rangle$

and

$\langle \forall x, z \in S : x\mathcal{O}z :$

(f) $xBz \vee$

(g) $\langle \exists a : z \to a : x\mathcal{O}a \wedge \mathit{rankls}(a, x) < \mathit{rankls}(z, x) \rangle$

and

$\langle \forall x, y, z \in S : xRz \wedge x \to y :$

(h) $\langle \exists a \colon z \to a \colon y\mathcal{O}a \rangle \rangle \vee$

(i) $yRz \wedge \mathit{rankts}(y, z) \prec \mathit{rankts}(x, z) \rangle$

We next prove that WRSS implies RES.

Theorem 31. Let $\mathcal{M}$ be a transition system. If B is a WRSS on $\mathcal{M}$ then B is an RES on $\mathcal{M}$.

Proof: Let B be a WRSS on $\mathcal{M} = \langle S, \to, L \rangle$, $x, y \in S$, and xBy. To show that B is an RES on $\mathcal{M}$, we show that RES1 and RES2 hold. RES1 follows directly from WRSS1.

Next we show that RES2 holds, *i.e.*, for any fullpath σ starting at x in $\mathcal{M}$, there is a fullpath δ starting at y in $\mathcal{M}$ such that $\mathit{rmatch}(B, \sigma, \delta)$ holds. We start by recursively defining $\pi, \xi \in \mathit{INC}$, and a fullpath δ in $\mathcal{M}$. For the base case,

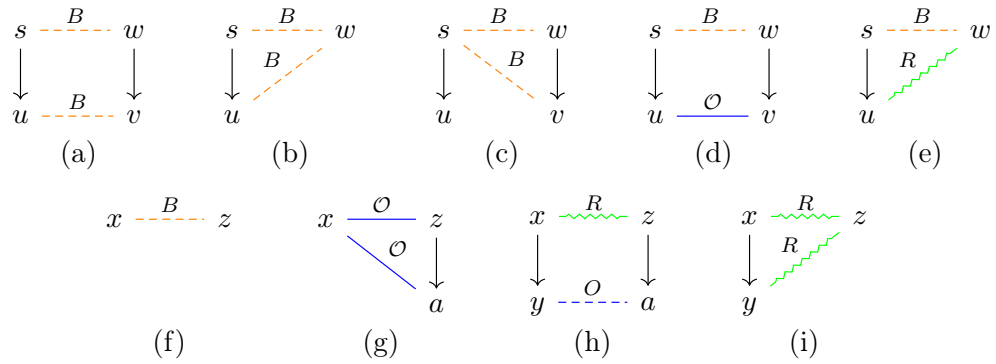

Figure 5.5.3: Well-founded reconciling simulation with explicit stuttering (a solid line with an arrow indicates the transition relation, a dashed orange line indicates that states are related by B, a solid blue line indicate the states are related by $\mathcal{O}$, and a squiggly green line indicates that states are related by R).

let $\pi.0 = 0, \xi.0 = 0$, and $\delta.0 = y$. From assumption, we have that $\sigma(\pi.0)B\delta(\xi.0)$. For the recursive case, we assume that $\pi.0, \ldots, \pi.i$, $\xi.0, \ldots, \xi.i$, and $\delta.0, \ldots, \delta(\xi.i)$ are defined, and that $\sigma(\pi.i)B\delta(\xi.i)$. Let $s = \sigma(\pi.i)$, $u = \sigma(\pi.i{+}1)$, and $w = \delta(\xi.i)$. Since WRSS2 holds there exists a well-founded domain $\langle W, \prec\rangle$, rank functions *rankt*, *rankl*, *rankls*, *rankts*, and binary relation $R, \mathcal{O}$ satisfying the three conjuncts in WRSS2. Before we proceed, we prove the following helpful lemma.

Lemma 32. Let $\mathcal{M} = \langle S, \rightarrow, L\rangle$ be a transition system, $x, z \in S$, $\mathcal{O}, B \subseteq S \times S$, and $\mathit{rankls} : S \times S \rightarrow \omega$. Let $\mathcal{O}, B$, and *rankls* satisfy the second universal quantifier in WRSS2. If $x\mathcal{O}z$ then there is a non-empty finite $\mathcal{M}$-segment starting from z to a state that is related to x by B.

Proof: Let $x\mathcal{O}z$. First suppose xBz holds. Then $\langle z\rangle$ is an $\mathcal{M}$-segment such that xBz. Next, suppose $\neg(xBz)$ and $\langle \exists a : z \rightarrow a : x\mathcal{O}a \wedge \mathit{rankls}(a, x) < \mathit{rankls}(z, x)\rangle$. Let δ be an $\mathcal{M}$-path starting at z such that only WRSS2g holds, *i.e.*, for all $i \geq 0$, $x\mathcal{O}\delta.i$ and $\mathit{rankls}(\delta(i+1), x) < \mathit{rankls}(\delta.i, x)$ and $\neg(xB\delta.i)$. Observe that δ must be a finite path in $\mathcal{M}$. Otherwise, it contradicts the well-foundedness of *rankls*. Let a be the last state of δ. Also since $x\mathcal{O}a$ and $\neg\langle\exists a' : a \rightarrow a' : x\mathcal{O}a' \wedge \mathit{rankls}(a', x) <$

$rankls(a, x)\rangle$, it must be the case that xBa holds.

□

We now return to the proof of Theorem 31. We consider five cases.

1. Suppose WRSS2a holds, *i.e.*, there is a successor v of w such that uBv holds. Let u and v mark the beginning of the next segment in σ and δ respectively. We define $\pi(i+1) = \pi.i + 1$, $\xi(i+1) = \xi.i + 1$, and ${}^{\xi}\delta^{i} = \langle w \rangle$.

2. Suppose WRSS2a does not hold and WRSS2d holds, *i.e.*, there is a successor v of w such that $u\mathcal{O}v$. From the second universal quantifier in WRSS2, and Lemma 32, we have that there is an $\mathcal{M}$-segment $\langle v_1 = v, \ldots, v_m \rangle$, where $m \geq 1$ such that uBv_m. Let u and v_m mark the beginning of the next segment in σ and δ respectively. We define $\pi(i+1) = \pi.i + 1$, $\xi(i+1) = \xi.i + m$, ${}^{\xi}\delta^{i} = \langle w, \ldots, v_{m-1} \rangle$, and $\delta(\xi(i+1)) = v_m$.

3. Suppose WRSS2a and WRSS2d do not hold and WRSS2e holds, *i.e.*, $\neg\langle \exists v : w \rightarrow v : uBv \rangle$, $\neg\langle \exists v : w \rightarrow v : u\mathcal{O}v \rangle$ and uRw. Let $\sigma(\pi.i{+}2) = u'$. We consider two cases. First, suppose that WRSS2h holds between u, u', and w, *i.e.*, there is a successor v of w such that $u'\mathcal{O}v$. From the second universal quantifier in WRSS2 and Lemma 32, we have that there is an $\mathcal{M}$-segment $\langle v_1 = v, \ldots, v_m \rangle$, where $m \geq 1$ such that $u'Bv_m$ holds. Let u' and v_m mark the beginning of the next segment in σ and δ respectively. We define $\pi(i+1) = \pi.i + 2$, $\xi(i+1) = \xi.i + m$, ${}^{\xi}\delta^{i} = \langle w, \ldots, v_{m-1} \rangle$ and, $\delta(\xi(i+1)) = v_m$. Second, suppose that WRSS2h does not hold and WRSS2i holds between u, u', and w. We define J to be the subset of the positive integers ≥ 2 such that for every $j \in J$, the following holds.

$$\langle \forall v : w \rightarrow v : \neg(\sigma(\pi.i+j)\mathcal{O}v)\rangle \ \wedge \tag{5.1}$$

$$\langle \sigma(\pi.i+j)Rw \ \ \wedge \ \ \mathit{rankts}(\sigma(\pi.i+j), w) \prec \mathit{rankts}(\sigma(\pi.i+j-1), w)\rangle$$

First observe that $2 \in J$ because $\sigma(\pi.i+2) = u'$, WRSS2h does not hold by assumption (so the first conjunct is true) and WRSS2i does (so the second conjunct is true). Next observe that there is an integer $n > 2$ such that $n \notin J$. Suppose not, then for all $n \geq 2, n \in J$. Now, consider the (infinite) suffix of σ starting at $\sigma(\pi.i+1)$. For every adjacent pair of states in this suffix, say $\sigma(\pi.i+k)$ and $\sigma(\pi.i+k+1)$ where $k \geq 1$, we have that only WRSS2i holds (*i.e.*, WRSS2h does not hold). This gives us a contradiction because *rankts* is well-founded. We now define $n = \mathit{min}(\{l : l \in \omega \wedge l > 2 \wedge l \notin J\})$. Notice that only WRSS2i holds between $\sigma(\pi.i+n-1)), \sigma(\pi.i+n)$ and w, hence $\sigma(\pi.i+n)Rw$ and $\mathit{rankts}(\sigma(\pi.i+n), w) \prec \mathit{rankts}(\sigma(\pi.i+n-1), w)$. Since Formula 5.1 does not hold for n, one of the conjuncts has to be false, and above we determined that it must be the first one, *i.e.*, there is a successor v of w such that $\sigma(\pi.i+n)\mathcal{O}v$. From the third universal quantifier in WRSS2 and Lemma 32, we have that there is an $\mathcal{M}$-segment $\langle v_1 = v, \ldots, v_m\rangle$, where $m \geq 1$, such that $\sigma(\pi.i+n)Bv_m$. We are now ready to extend our recursive definition as follows: $\pi(i+1) = \pi.i+n$, $\xi(i+1) = \xi.i+m$, and ${}^{\xi}\delta^{i} = \langle w, \ldots, v_{m-1}\rangle$.

4. Suppose only WRSS2b holds between s, u, and w. The argument in this case is similar to the above case except that we use well-foundedness of *rankt* instead of well-foundedness of *rankts*. We define K to be the subset of positive integers such that for every $k \in K$, following holds.

$$\langle \forall v : w \to v : \neg(\sigma(\pi.i+k)Bv)\rangle \wedge$$
$$\langle \forall v : w \to v : \neg(\sigma(\pi.i+k)\mathcal{O}v)\rangle \wedge \neg(\sigma(\pi.i+k)Rw) \wedge$$
$$(\sigma(\pi.i+k)Bw \wedge \mathit{rankt}(\sigma(\pi.i+k),w) \prec \mathit{rankt}(\sigma(\pi.i+k-1),w)) \quad (5.2)$$

First observe that $1 \in K$ because $\sigma(\pi.i) = s, \sigma(\pi.i{+}1) = u$, WRSS2a, WRSS2d and WRSS2e do not hold by assumption (so the first, the second, and the third conjunct are true) and WRSS2b does (so the fourth conjunct is true). Next observe that there exists an integer $n > 1$ such that $n \notin K$. Suppose not, then for all $n \geq 1$, $n \in K$. Consider the (infinite) suffix of σ starting at $\sigma(\pi.i+1)$. For every adjacent pair of states in this suffix, say $\sigma(\pi.i+l)$ and $\sigma(\pi.i+l+1)$ where $l \geq 1$, we have that only WRSS2b holds. This gives us a contradiction because *rankt* is well-founded. We now define $n = \mathit{min}(\{l : l \in \omega \wedge l > 1 \wedge l \notin K\})$. Notice that WRSS2b holds between $\sigma(\pi.i+n-1), \sigma(\pi.i+n)$ and w, hence $\sigma(\pi.i+n)Bw \wedge \mathit{rankt}(\sigma(\pi.i+n,w)) \prec \mathit{rankt}(\sigma(\pi.i+n-1,w))$. Since Formula 5.2 does not hold for n, one of the conjuncts has to be false, and we determined above that it is not the fourth conjunct. Suppose it is the first conjunct, *i.e.*, $\langle \exists v : w \to v : \sigma(\pi.i+n)Bv\rangle$. In this case, we extend our recursive definition as follows: $\pi(i+1) = \pi.i+n$, $\xi(i+1) = \xi.i+1$, ${}^{\xi}\delta^{i} = \langle w\rangle$. Suppose it is not the first conjuct and it is the second conjuct, *i.e.*, $\langle \exists v : w \to v : \sigma(\pi.i+n)\mathcal{O}v\rangle$. From the second universal quantifier in WRSS2 and Lemma 32, we have that there is an $\mathcal{M}$-segment $\langle v_1 = v, \ldots, v_m\rangle$, where $m \geq 1$, such that $\sigma(\pi.i+n)Bv_m$. We extend our recursive definition as follows: $\pi(i+1) = \pi.i+n$, $\xi(i+1) = \xi.i+m$, and ${}^{\xi}\delta^{i} = \langle w, \ldots, v_{m-1}\rangle$. Suppose it is the third conjunct, *i.e.*, $\sigma(\pi.i+n)Rw$. Then (as above) from the

well-foundedness of *rankts* and Lemma 32, we can infer that there are positive integers m, p, an $\mathcal{M}$-segment $\langle w = v_0, \ldots, v_m \rangle$ such that $\sigma(\pi.i+n+p)Bv_m$. In this case, we extend our recursive definition as follows: $\pi(i+1) = \pi.i + n + p$, $\xi(i+1) = \xi.i + m$, and ${}^{\xi}\delta^i = \langle w, \ldots, v_{m-1} \rangle$.

5. Finally, we consider the case when only WRSS2c holds between s, u and w, *i.e.*, there is a successor v of w such that sBv and $rankl(v, s, u) < rankl(w, s, u)$. Let γ be an $\mathcal{M}$-path starting at w such that only WRSS2c holds between s, u and a state in γ. There is a non-empty path that satisfy this condition, *e.g.*, let $\gamma = \langle w \rangle$. Notice that any such path must be finite. If not, then for any adjacent pair of states in γ, say $\gamma.k$ and $\gamma(k+1)$, where $k \in \omega$, $rankl(\gamma(k+1), s, u) < rankl(\gamma.k, s, u)$, which contradicts well-foundedness of *rankl*. Let $\vec{v} = \langle v_0 = w, \ldots, v_m \rangle$, where $m \geq 0$, be a maximal $\mathcal{M}$-segment satisfying the above condition. Since sBv_m holds, one of WRSS2a, WRSS2b, WRSS2d, WRSS2e must hold between s, u, and v_m and WRSS2c does not hold. Notice that we are in one the four cases above except that the segments starts at v_m. In each we can extend the segments appropriately , *i.e.*, we can infer that there are positive integers n, p, and an $\mathcal{M}$-segment obtained by extending $\vec{v}$, say $\langle v_0 = w, \ldots, v_m, \ldots, v_{m+n} \rangle$ such that $\sigma(\pi.i + p)Bv_{m+n}$. In this case, we extend our recursive definitions as follows: $\pi(i+1) = \pi.i + p$, $\xi(i+1) = \xi.i + m + n$, and ${}^{\xi}\delta^i = \langle w, \ldots, v_{m+n-1} \rangle$.

Now we show that RES2 holds. We start by unwinding definitions. The first step is to show that $fp.\delta.y$ holds, which is true by construction. Next, we show that $rmatch(B, \sigma, \delta)$ by unwinding the definition of *rmatch*. That involves showing that there exist π and ξ such that $rcorr(B, \sigma, \pi, \delta, \xi)$ holds. The π and ξ we used to define δ can be used here. Finally, we unwind the definition of *rcorr*, which gives us a universally quantified formula over the natural numbers. This is handled by

induction on the segment index; the proof is based on the recursive definitions given above.

□

The following theorem states that all the three characterizations of reconciling simulation introduced above are equivalent.

Theorem 33. Let $\mathcal{M} = \langle S, \rightarrow, L \rangle$ be a transition system and $B \subseteq S \times S$. The following are equivalent

(i) B is an RES on $\mathcal{M}$;

(ii) B is an RWRS on $\mathcal{M}$;

(iii) B is a WRS on $\mathcal{M}$;

(iv) B is WRSS on $\mathcal{M}$

Proof: (i) implies (ii) follows from Theorem 29, (ii) implies (iii) follows from Theorem 30, and (iv) implies (i) in Theorem 31. To conclude the proof, we show that (iii) implies (iv).

WRSS1 follows directly from WRS1. To prove that WRSS2 holds, we use the same rank functions *rankts* and *rankls* and binary relations R and $\mathcal{O}$. If WRS2a or WRS2b holds then WRSS2d or WRSS2e holds. Clearly, $\mathcal{O}$ and R satisfy the second and the third conjunct in WRSS2.

□

5.6 Summary

In this chapter, we introduced a new notion of reconciling simulation, studied its algebraic properties, and based on it developed a theory of reconciling refinement.

Unlike skipping refinement, reconciling refinement is not compositional. We developed three alternative characterizations of reconciling simulation that are amenable for mechanised reasoning using formal-methods tools.

Well-founded reconciling simulation (Definition 20) and well-founded reconciling simulation with explicit stuttering (Definition 21) are complete proof methods that require only local reasoning, *i.e.*, reasoning about states and their successors. Thus, if a concrete system is reconciling refinement of an abstract system under a refinement map, one can always prove it using local reasoing about states and their successors and do not require global reasoning about infinite paths. Furthermore, reconciling simulation is a strictly weaker notion of correctness than skipping simulation (and stuttering simulation). Together, these results significantly extend the domain of applicability of refinement-based methodology to verification of a large class of reactive systems.

The approach we followed towards developing a sound and complete proof methods for mechanised reasoning of reconciling simulation is similar to the approach for skipping simulation in Chapter 4. We first defined an intermediate characterization, RWRS (Definition 17), that localizes reasoning on the left by introducing an auxiliary binary relation on states (conditions RWRS2b-2d), while reasoning about reconciling on the left is accounted for by transitive closure (condition RWRS2a) and in general requires reachability analysis. Then, similar to RLWFSK (Definition 12), we defined WRS (Definition 20) that localizes reasoning about reconciling on the right by introducing another auxiliary binary relation on states (conditions WRS2a, WRS2c-2d). Finally, similar to LWFSK (Definition 13), we define WRSS that explicitly accounts for stuttering. In fact, one can look at LWFSK as WRSS modulo conditions WRSS2e, WRSS2h-2i, conditions that account for reconciling on the left, a scenario that does not occur when reasoning about skipping simulation.

Finally, observe that one can envision defining other equivalent characterizations. For example, we can define a notion that distinguishes between stuttering and reconciling on the left (as in condition WRSS2b and WRSS2e) and use transitive closure (as in RWRS2a) to combine stuttering and skipping the right. Such a characterization is justified as it can be useful when skipping on the right is bounded (and small). The three characterizations of reconciling simulation that we choose to study are interesting in the sense that they highlight the considerations when designing a sound and complete proof method that involves only local reasoning.

Chapter 6

Case Studies

In this chapter, we present case studies that enable us to explore the conceptual aspects of the theory of skipping refinement and applicability of associated proof methods. When verifying that an implementation system refines a specification system two primary choices are: (1) selecting an appropriate notion of refinement, and (2) selecting a proof-method that can be used to mechanically prove it. Our goal is to highlight the impact of these choices on the specification cost and effectiveness of proof methods when using existing automated verification tools. We evaluate the impact of these choices in analyzing correctness of four optimized reactive systems: (1) an asynchronous event-processing system, (2) a JVM-inspired stack machine, (3) an optimized memory controller, and (4) a vectorizing compiler transformation. The first case study is an example of a system where skipping is unbounded while (2)-(4) are examples of systems where skipping is bounded by a constant. To facilate the understanding and focus on evaluating the theory of refinement, we only model certain aspects of systems.

6.1 Event Processing Systems

Asynchronous event-based programming is a widely accepted methodology to design responsive and efficient user interfaces (UI). In this methodology a programmer designates a main UI thread that only performs short-running work in response to user events, while more computationally demanding part of the work is delegated to asynchronous tasks. Depending on the priority of the tasks and computational resources available on the execution platform, these asynchronous tasks are then scheduled to be executed at some time in the future. In this case study we study correctness of PEPS, a simple asynchronous event-based processing system. PEPS uses a priority queue to find the next event to execute and event-handlers interact with each other using a shared memory. We place no limits on the number of event-handlers in the scheduler. We show that PEPS is a skipping refinement of AEPS, a simple abstract event processing system introduced in Section 4.1. This case study illustrates reasoning about skipping refinement when one cannot a priori determine an upper-bound on skipping. In such cases unrolling the transition relation is not a viable option.

We begin by describing the transition system of PEPS. Let E be the set of event-handlers and V be the set of state variables. Let (e, t) be a timed-event pair, where $e \in E$ scheduled to be executed at time t. Let $<_e$ be a total order on event-handlers. A timed-event pair (e_1, t_1) must be executed before a timed-event pair (e_2, t_2) if $t_1 < t_2$ or $(t_1 = t_2) \wedge (e_1 <_e e_2)$. A state of PEPS is a three-tuple $\langle t, \mathit{otevs}, \mathit{mem}\rangle$ where t is a natural number denoting the current time, *otevs* is an ordered list of timed event-handler pairs of the form (e, t) denoting the scheduler, and *mem* is a list of assignments to state variable denoting the shared memory. We require that all event-handlers in *otevs* are scheduled to execute at a time greater or equal to the current time t. The transition function of PEPS is defined as follows. If there are

no event-handlers in *otevs*, then PEPS just increments the current time by 1. Else it picks the first element in *otevs* and executes it. Execution of an event-handler may remove zero or more event-handlers from the scheduler and also insert a finite number of new event-handlers in the scheduler. We require that any new event that is added to scheduler must be scheduled to execute at time $\geq t$. Execution of an event-handler may also update assignments to state variables in the memory *mem*. Finally, execution involves removing the executed event-handler form *otevs*. To keep the modeling simple, we assume that execution of an event-handler is an atomic operation.

Next we recall the operational semantics of the AEPS, a simple abstract event-processing system. Like PEPS, the state of AEPS is a three-tuple $\langle t, tevs, mem\rangle$, where t is a natural number denoting the current time, *tevs* is a list of timed-event pairs (e, t) denoting the scheduler, and *mem* is a list of assignments to state variables denoting the shared memory. Note that unlike PEPS, where the scheduler is an ordered list of timed-event pairs, the scheduler in AEPS is simply a *set* of timed-event pairs. Let w, v be AEPS states, and $w = \langle t_w, tevs_w, mem_w\rangle$. We say w transitions to v iff there exists a timed-event pair, say (e, t_e), in $tevs_w$, $t_w = t_e$ and the result of executing the e results in a state equivalent to v. Thus, unlike PEPS which is a deterministic system, AEPS can nondeterministically choose an event-handler to execute from the set of event-handlers scheduled to execute at the current time.

Let $\mathcal{M}_{\mathrm{A}} = \langle S_A, \xrightarrow{A}, L_A\rangle$ and $\mathcal{M}_{\mathrm{C}} = \langle S_C, \xrightarrow{C}, L_C\rangle$ be the transition systems of AEPS and PEPS respectively. The set of states S_A, S_C, and the transition relations $\xrightarrow{A}, \xrightarrow{C}$ are as described above, and labeling functions L_A and L_C are just the identity functions. We show that $\mathcal{M}_C$ refines $\mathcal{M}_A$ using a stepwise refinement approach: first we define an intermediate system $\mathcal{M}_H$ obtained by augmenting $\mathcal{M}_C$ with history

information and show that $\mathcal{M}_C$ is a skipping refinement of $\mathcal{M}_H$. Second we show that $\mathcal{M}_H$ is a skipping refinement of $\mathcal{M}_A$. Finally, we appeal to Theorem 8 to infer that $\mathcal{M}_C$ refines $\mathcal{M}_A$.

We begin by defining $\mathcal{M}_{\mathrm{H}} = \langle S_H, \xrightarrow{H}, L_H \rangle$. Let $H = \langle valid, pre \rangle$, where *valid* is a boolean variable and $pre \in S_C$. The S_H, the state space of $\mathcal{M}_H$ is defined as: $S_H = S_C \times H$. The transition relation $\xrightarrow{H}$ is defined as follows: $\langle s, h \rangle \xrightarrow{H} \langle s', \langle valid, pre \rangle \rangle$ iff $s \xrightarrow{C} s'$, $valid = true$, and $pre = s$ Finally, L_H is defined as the projection of S_H to S_C. We show that $\mathcal{M}_C$ is a skipping refinement of $\mathcal{M}_H$ under a refinement map P, where $P\langle t, otevs, mem \rangle = \langle t, otevs, mem, h \rangle$ and $h \in H$ is such that its *valid* field is false. Intuitively, this implies that $\mathcal{M}_C$ and $\mathcal{M}_H$ have same observable behaviors, *i.e.*, augmenting PEPS with history information does not modify its observable behavior. From Definition 7, we must exhibit as witness a binary relation A that satisfies two conditions: first it agrees with the the refinement map P and second that A is an SKS on the disjoint union of PEPS and HPEPS. We define A to be the relation induced by P, *i.e.*, a PEPS state s is related to a HPEPS state w *iff* $P.s = w$. It is straightforward to show that the A satisfies the first condition. Next, we use RWFSK to show that A is an SKS relation. We choose RWFSK over the other three proof-methods (Section 4.4) based on the observation that PEPS does not skip with respect to HPEPS. Hence, we avoid defining additional constructs that account for skipping in other three proof methods. We can further simplify RWFSK2 based on following two observations: (1) PEPS does not stutter; therefore we can ignore RWFSK2a, and (2) HPEPS is a deterministic systems; hence we can drop the existential quantifier in RWFSK2b. Let $f_C : S_C \to S_C$ and $f_H : S_H \to S_H$ be the transition functions of $\mathcal{M}_C$ and $\mathcal{M}_H$ respectively. Then to show that $\mathcal{M}_C \lesssim_P \mathcal{M}_H$, our proof obligation is:

For all $s \in S_C$, $w \in S_H$, $sAw \Rightarrow (f_C.s)A(f_H.w)$

Note that we can prove that $\mathcal{M}_C$ is a simulation refinement of $\mathcal{M}_H$ and establish a stronger relationship between the two systems. However, for our purpose it suffices to show that $\mathcal{M}_C$ is a skipping refinement of $\mathcal{M}_H$.

Next we show that $\mathcal{M}_H$ is a skipping refinement of $\mathcal{M}_A$ under the refinement map $R : S_H \to S_A$ defined as let $s = \langle t, otevs, mem, h\rangle$ be an $\mathcal{M}_H$-state, where $h \in H$. Then $R.s = \langle t, otevs, mem\rangle$. To show that $\mathcal{M}_H \lesssim_R \mathcal{M}_A$, from Definition 7, we must show as witness a binary relation B that satisfies two conditions: first it agrees with the the refinement map R and second that B is an SKS on the disjoint union of HPEPS and AEPS. Let $B = \{(s, R.s) : s \in S_H\}$. It is straightforward to show that B satisfies the first condition. To show that B is an SKS on the disjoint union of HPEPS and AEPS, we have a choice of four proof-methods (Section 4.4). Which is an appropriate one to analyse correctness of HPEPS? Towards this observe that when no event-handlers are scheduled to execute at the current time `t`, HPEPS skips over states of AEPS: while AEPS increments the `t` by 1, HPEPS updates `t` to the earliest time in future when an event-handler is scheduled for execution. Moreover, we cannot a priori determine an upper bound on skipping. For example, execution of an event-handler may add a new event-handler that is scheduled to execute at an arbitrary time in the future. Without an upper bound on skipping, conditions RWFSK2b (Definition 8) and WFSK2d (Definition 11) requires reachability analysis that is not amenable for mechanical reasoning. In contrast, RLWFSK and LWFSK require only local reasoning. Finally, AEPS does not stutter; hence the distinction between stuttering and skipping provided by LWFSK is not helpful. Therefore, we choose RLWFSK to show that B is an SKS relation.

RLWFSK1 holds trivially. To prove that RLWFSK2 holds we define a binary relation $\mathcal{O}$ and a rank function *rankls* and show that they satisfy the two universally quantified formulas in RLWFSK2. Moreover, since HPEPS does not stutter we

ignore RLWFSK2a, and that is why we do not define *rankt*.

For all $s \in S_H, w \in S_A$, and sBw:

$$\textit{(RLWFSK2b)}\ \langle \exists v : w \xrightarrow{A} v : (f_H.s)\mathcal{O}v \rangle$$

For all $x \in S_H$, $y \in S_A$, $x\mathcal{O}y$:

$$\textit{(RLWFSK2c)}\ xBy \vee$$
$$\textit{(RLWFSK2d)}\ \langle \exists z : y \xrightarrow{A} z : x\mathcal{O}z \wedge \textit{rankls}(z,x) < \textit{rankls}(y,x) \rangle$$

Note that the proof obligations involves only states and their successors, *i.e.*, the reasoning is local. This is in contrast to reasoning about SKS using WFSK and RWFSK, which in presence of unbounded skipping exhibited by PEPS, would require reachability analysis. Informally, we define $\mathcal{O}$ as follows: states x, y are related by $\mathcal{O}$ *iff* one of the following conditions hold: (1) x and y are related by B, or (2) a predecessor of x, obtained from the history `h` in x, has the current time component less than or equal to the current time component in y, has a non-empty scheduler that is set-equal to the scheduler in y, and has a memory that is set-equal to the memory in y. The function *rankls* is defined as follows: given a HPEPS state x, and an AEPS state y, *rankls* is equal to the (absolute) difference between the `t` in x and `t` in y plus the number of events-handlers in the scheduler in y scheduled to execute at `t` in x. The witness states in the above proof obligations were easy to determine based on whether or not there is an event-handler scheduled for execution at the current time. We model the operational semantics of PEPS and AEPS and discharge the proof obligations in ACL2s. The proof also involves discharging appropriate lemmas about adding and removing events from a list (representing sets) and an ordered-list (representing priority queue). The models and the proof script are publicly

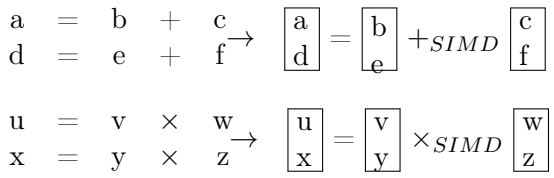

Figure 6.2.1: Superword Parallelism

available [1].

Finally, we remark that PEPS can be seen as an instance of MPEPS system introduced in 4.3: consider an implementation of the priority queue in MPEPS that deterministically selects *one* event-handler from the set of event-handlers scheduled to execute at the current time. Therefore, PEPS is a skipping refinement of AEPS can also be inferred from the proof that MPEPS is a skipping refinement of AEPS.

6.2 Superword Level Parallelism with SIMD instructions

An effective way to improve the performance of multimedia programs running on modern commodity architectures is to exploit Single-Instruction Multiple-Data (SIMD) instructions (*e.g.*, the SSE/AVX instructions in x86 microprocessors). Compilers analyze programs for *superword level parallelism* and when possible replace multiple scalar instructions with a compact SIMD instruction that operates on multiple data in parallel [9]. In this case study, we illustrate the applicability of skipping refinement to verify the correctness of such a compiler transformation using ACL2s. The source language consists of scalar instructions and the target language consists of both scalar and vector instructions. We model the transformation as a function that takes as input a program in the source language and outputs a program in the target language. Instead of proving that the compiler is correct, we use the translation validation approach [4] and prove the equivalence of the source program and the generated target program.

We make some simplifying assumptions for modelling purpose: the state of the

source and target programs (modeled as transition systems) is a three-tuple consisting of a sequence of instructions, a program counter and a store. We also assume that a SIMD instruction operates on *two* sets of data operands simultaneously and that the transformation identifies parallelism at the basic block level. Therefore, we do not model any control flow instruction. Note that we do *not* reorder instructions in the source program. Figure 6.2.1 shows how two add and two multiply scalar instructions are transformed into corresponding SIMD instructions.

$$loc := \{x, y, z, a, b, c, \ldots\} \quad \text{(Variables)}$$
$$sop := add \mid sub \mid mul \mid and \mid or \mid nop \quad \text{(Scalar Ops)}$$
$$vop := vadd \mid vsub \mid vmul \mid vand \mid vor \mid vnop \quad \text{(Vector Ops)}$$
$$sinst := sop\langle z\ x\ y\rangle \quad \text{(Scalar Inst)}$$
$$vinst := vop\langle c\ a\ b\rangle\langle f\ d\ e\rangle \quad \text{(Vector Inst)}$$
$$sprg := [] \mid sinst :: sprg \quad \text{(Scalar Program)}$$
$$vprg := [] \mid (sinst \mid vinst) :: vprg \quad \text{(Vector Program)}$$
$$store := [] \mid \langle x, v_x\rangle :: store \quad \text{(Registers)}$$

Scalar Machine ($\xrightarrow{A}$)

$$\frac{\langle sprg, pc, store\rangle,\ \{\langle x, v_x\rangle, \langle y, v_y\rangle\} \subseteq store,\ sprg[pc] = sop\langle z\ x\ y\rangle,\ v_z = [\![(sop\ v_x\ v_y)]\!]}{\langle sprg, pc + 1, store|_{z:=v_z}\rangle}$$

Vector Machine ($\xrightarrow{C}$)

$$\frac{\langle vprg, pc, store\rangle,\ \{\langle x, v_x\rangle, \langle y, v_y\rangle\} \subseteq store,\ vprg[pc] = sop\langle z\ x\ y\rangle,\ v_z = [\![(sop\ v_x\ v_y)]\!]}{\langle vprg, pc + 1, store|_{z:=v_z}\rangle}$$

$$\frac{\langle vprg, pc, store\rangle,\ vprg[pc] = vop\langle c\ a\ b\rangle\langle f\ d\ e\rangle,\ \{\langle a, v_a\rangle, \langle b, v_b\rangle, \langle d, v_d\rangle, \langle e, v_e\rangle\} \subseteq store,\ \langle v_c, v_f\rangle = [\![(vop\ \langle v_a\ v_b\rangle\langle v_d\ v_e\rangle)]\!]}{\langle vprg, pc + 1, store|_{c:=v_c, f:=v_f}\rangle}$$

Figure 6.2.2: Syntax and Semantics of Scalar and Vector Program

The syntax and operational semantics of source and target programs are given

in Figure 6.2.2, using the same conventions. We denote that $x, \ldots, y$ are variables with values $v_x, \ldots, v_y$ in *store* by $\{\langle x, v_x\rangle, \ldots, \langle y, v_y\rangle\} \subseteq store$.$[\![(sop\ v_x\ v_y)]\!]$ denotes the result of the scalar operation *sop* and $[\![(vop\ \langle v_a\ v_b\rangle\langle v_d\ v_e\rangle)]\!]$ denotes the result of the vector operation *vop*. Finally, we use $store|_{x:=v_x,\ldots,y:=v_y}$ to denote that variables $x, \ldots, y$ are updated (or added) to *store* with values $v_x, \ldots, v_y$.

Since target program can execute multiple scalar instructions in a single step, the notion of stuttering simulation is too strong to relate the source and the target program produced by the compiler, no matter what refinement map we use. To see this, note that the target program might run exactly twice as fast as the source program and during each step the scalar program might be modifying the memory. Since both the programs do not stutter, in order to use stuttering refinement the length of run of the source and target program must be equal, which is not in our case.

Let $\mathcal{M}_A$ and $\mathcal{M}_C$ be transition systems of the source and target programs, respectively. The target program is correct iff $\mathcal{M}_C$ is a skipping refinement of $\mathcal{M}_A$. We show $\mathcal{M}_C \lesssim_r \mathcal{M}_A$, using Definition 8. Determining j, an upper-bound on skipping that reduces condition RWFSK2b in Definition 8 to bounded reachability is simple because a step of the target program corresponds to at most 2 steps of the source program; therefore $j = 3$ suffices.

We next define the refinement map. Let *sprg* be the source program and *vprg* be the compiled target program. We use the compiler to generate a lookup table pcT that maps values of the target program counter to the corresponding values of the source program counter. The refinement map $R : S_C \to S_A$ is defined as follows.

$$R(\langle vprg, pc, store\rangle) = \langle sprg, pcT(pc), store\rangle$$

Note that $pcT(pc)$ can also be determined using a history variable and this is prefer-

able from a verification efficiency perspective. Given R, we define $B = \{\langle s, R.s\rangle \mid s \in S_C\}$.

Since the machines do not stutter, RWFSK2 (Definition 8) can be simplified as follows. For all $s, u \in S_C$ such that $s \xrightarrow{C} u$:

$$R.s \xrightarrow{A}{}^{<3} R.u \tag{6.1}$$

Since the semantics of the source program is deterministic, u is a function of s, so we can remove u from the above formula, if we wish. Also, we can expand out $\xrightarrow{A}{}^{<3}$ to obtain a formula using only $\xrightarrow{A}$ instead.

For this case study we used deductive verification methodology to prove correctness. The source and the target program are specified using the data-definition framework in ACL2s [6, 5]. We formalized the operational semantics of a scalar and a target program using standard methods. The sizes of the program and store are unbounded and thus the state space of the machines is infinite. Once the definitions were in place, proving skipping refinement with ACL2s was straightforward. In contrast proving input-output equivalence would have required theorem proving expertise and insight to come up with the right invariants, something we avoided. The proof scripts are publicly available [1].

Chapter 7

Related Work

Clarify the terminology “skipping on the left” and “skipping on the right”. Which system is running faster.

Bibliography

[1] Skipping simulation model, 2015.

[2] M. Abadi and L. Lamport. The existence of refinement mappings. In *Theoretical Computer Science*, 1991.

[3] R.-J. Back. Refinement calculus, part II: Parallel and reactive programs. In *Stepwise Refinement of Distributed Systems Models, Formalisms, Correctness*, 1990.

[4] C. W. Barrett, Y. Fang, B. Goldberg, Y. Hu, A. Pnueli, and L. D. Zuck. TVOC: A translation validator for optimizing compilers. In *CAV*, 2005.

[5] H. R. Chamarthi, P. C. Dillinger, and P. Manolios. Data definitions in the ACL2 sedan. In *ACL2*, 2014.

[6] H. R. Chamarthi, P. C. Dillinger, P. Manolios, and D. Vroon. The ACL2 sedan theorem proving system. In *TACAS*, 2011.

[7] M. Jain and P. Manolios. Skipping refinement. In *CAV*, 2015.

[8] M. Jain and P. Manolios. An efficient runtime validation framework based on the theory of refinement. *CoRR*, abs/1703.05317, 2017.

[9] S. Larsen and S. P. Amarasinghe. Exploiting superword level parallelism with multimedia instruction sets. In *PLDI, 2000*, 2000.

[10] P. Manolios. Correctness of pipelined machines. In *FMCAD*, 2000.

[11] P. Manolios. *Mechanical verification of reactive systems*. PhD thesis, University of Texas, 2001.

[12] P. Manolios and S. K. Srinivasan. A computationally effecient method based on commitment refinement maps for verifying pipelined machines. In *MEMOCODE*, 2005.

[13] J. Misra. Distributed discrete-event simulation. *ACM Computing Survey*, 1986.

[14] N. Wirth. Program development by stepwise refinement. *Communications of the ACM*, 1971.