REFINEMENT-BASED REASONING OF OPTIMIZED REACTIVE SYSTEMS

Mitesh Jain

Contents

List of Figures

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

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.

3.1 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 [7]: 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 3.1.1. (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 notion 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.

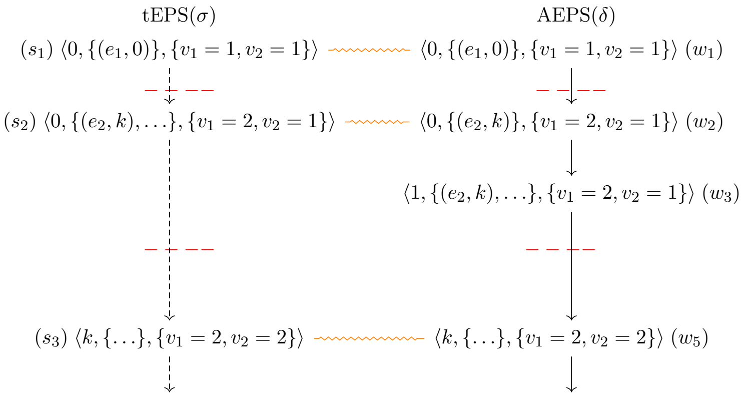
tEPS(σ)
AEPS(δ)
(s_1) $\langle 0, \{(e_1, 0)\}, \{v_1 = 1, v_2 = 1\}\rangle$
$\langle 0, \{(e_1, 0)\}, \{v_1 = 1, v_2 = 1\}\rangle$ (w_1)
(s_2) $\langle 0, \{(e_2, k), \ldots\}, \{v_1 = 2, v_2 = 1\}\rangle$
$\langle 0, \{(e_2, k)\}, \{v_1 = 2, v_2 = 1\}\rangle$ (w_2)
$\langle 1, \{(e_2, k), \ldots\}, \{v_1 = 2, v_2 = 1\}\rangle$ (w_3)
(s_3) $\langle k, \{\ldots\}, \{v_1 = 2, v_2 = 2\}\rangle$
$\langle k, \{\ldots\}, \{v_1 = 2, v_2 = 2\}\rangle$ (w_5)

Figure 3.1.1: Event Processing System

3.2 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 2 (smatch)**.** 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

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

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

In Figure 3.1.1, 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 $\mathit{smatch}(B, \sigma, \delta)$ holds, we have to find $\pi, \xi \in \mathit{INC}$ 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 ⁄\/\⁄. Since all elements of a σ partition are related to the first element of the corresponding δ partition, $\mathit{scorr}(B, \sigma, \pi, \delta, \xi)$ holds, therefore, $\mathit{smatch}(B, \sigma, \delta)$ holds.

Definition 3 (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) $\langle \forall \sigma \colon \mathit{fp}.\sigma.s \colon \langle \exists \delta \colon \mathit{fp}.\delta.w \colon \mathit{smatch}(B, \sigma, \delta) \rangle \rangle$

3.2.1 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 3.2.1.

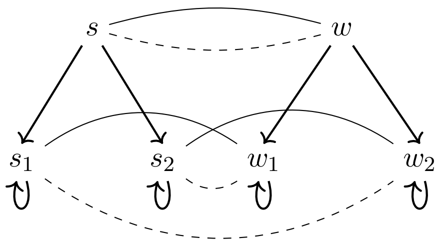

Figure 3.2.1: An example showing that SKS is not closed under intersection. Consider a TS with set of states $S = \{s, w, s_1, s_2, w_1, w_2\}$. The transition relation is denoted by solid arrows and all states are labeled identically. The first SKS relation B_1, denoted by solid lines, is $\{(s, w), (s_1, w_1), (s_2, w_2)\}$. 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 $\alpha, \beta, \theta, \gamma \in INC$ such that $scorr(P, \sigma, \alpha, \tau, \beta)$ and $scorr(Q, \tau, \theta, \delta, \gamma)$ hold. We use the fullpath δ as a witness and define $\pi, \xi \in INC$ such that $scorr(R, \sigma, \pi, \delta, \xi)$ 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 $\mathit{scorr}(P, \sigma, \alpha, \tau, \beta)$), 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 $\pi\alpha \in \mathit{INC}$, 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.

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

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

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.

□

3.3 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 3.4) provide a reasoning framework that can be used to analyse a large class of reactive systems.

Definition 4 (Skipping Refinement)**.** Let $\mathcal{M}_\text{A} = \langle S_A, \xrightarrow{A}, L_A\rangle$ and $\mathcal{M}_\text{C} = \langle S_C, \xrightarrow{C}, L_C\rangle$ 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* [1] 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 [6, 4]. 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 **??**.

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 **??** 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 2.

□

Corollary 10. Let $\mathcal{M}_{\mathrm{C}} = \langle S_C, \xrightarrow{C}, L_C \rangle$ and $\mathcal{M}_{\mathrm{A}} = \langle S_A, \xrightarrow{A}, L_A \rangle$ 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 3.1, 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 $p' = p; I_1; I_2$.

3.4 Mechanised Reasoning

To prove that a transition system $\mathcal{M}_C$ is a skipping refinement of a transition system $\mathcal{M}_A$ using Definition 3, 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.

3.4.1 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 3.4.1): (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.

$$\begin{tikzcd}
s \arrow[r, dashed, no head, "B"] \arrow[d] & w \arrow[dl, dashed, no head, "B"'] \\
u &
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
s \arrow[r, dashed, no head, "B"] \arrow[d] & w \arrow[d, "\geq 1"] \\
u \arrow[r, dashed, no head, "B"] & v
\end{tikzcd}$$

(b)

Figure 3.4.1: 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 5 (Reduced Well-founded Skipping [3])**.** $B \subseteq S \times S$ 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

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

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

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

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 ([3]). 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 $\mathcal{M} = \langle S, \rightarrow, L \rangle$, 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.

$$\begin{gathered}\sigma(\pi.i + j) \not{B}\, w \rightarrow^{+} v : \neg(\sigma(\pi.i + j)Bv)\rangle \wedge \\ \sigma(\pi.i + j)Bw \;\wedge\; \mathit{rankt}(\sigma(\pi.i + j), w) \prec \mathit{rankt}(\sigma(\pi.i + j - 1), w)\end{gathered} \tag{3.1}$$

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 $\mathit{min}(\{l : l \notin J\})$. 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 3.1 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 *fp.δ.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 ([3]). 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 6. 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 7 (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

$\mathit{ranktCt}(\mathcal{M}, B, s, w)$ is the largest subtree of $\mathit{ctree}(\mathcal{M}, s)$ 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 $\pi, \xi \in \mathit{INC}$ 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 $\mathit{ranktCt}(\mathcal{M}, B, s, w)$ to be $\mathit{size}(\mathit{ranktCt}(\mathcal{M}, B, s, w), \langle s \rangle)$. We use $\preceq$ to compare ordinal numbers (and therefore cardinal numbers as well).

Lemma 14 ([5]). If $|S| \preceq \kappa$, where $\omega \preceq \kappa$ then for all $s, w \in S$, $\mathit{size}(\mathit{ranktCt}(\mathcal{M}, B, s, w))$ is an ordinal of cardinality $\preceq \kappa$.

Lemma 14 shows that we can use the cardinal $\mathit{max}(|S|^+, \omega)$ 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 $sBw, s \rightarrow u, \langle s, u \rangle \in \mathit{ranktCt}(\mathcal{M}, B, s, w)$ then $\mathit{size}(\mathit{ranktCt}(\mathcal{M}, B, u, w)) \prec \mathit{size}(\mathit{ranktCt}(\mathcal{M}, B, s, w))$.

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

□

We are now ready to prove that SKS implies RWFSK.

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

There are two cases. First, suppose that $\langle \exists v : w \rightarrow^+ v : uBv \rangle$ holds, then RWFSK2(b) holds. If not, then $\langle \forall v : w \rightarrow^+ v : \neg(uBv) \rangle$, 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 $\mathit{fp}.\delta.w$ and $\mathit{smatch}(B, \sigma, \delta)$. Hence, there exists $\pi, \xi \in \mathit{INC}$ such that $\mathit{scorr}(B, \sigma, \pi, \delta, \xi)$. 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, $\mathit{rankt}(u, w) = \mathit{size}(\mathit{ranktCt}(\mathcal{M}, B, u, w)) \prec \mathit{size}(\mathit{ranktCt}(\mathcal{M}, B, s, w)) = \mathit{rankt}(s, w)$.

□

3.4.2 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 $s \rightarrow u$, there are four cases to consider (Figure 3.4.2): (a) either we can match the move from s to u in a single step, *i.e.*, there is a v such that $w \rightarrow v$ 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.

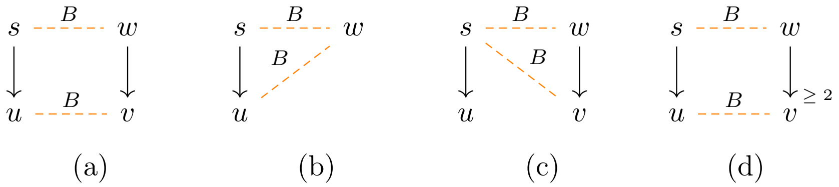

Figure 3.4.2: Well-founded skipping simulation (a solid line with arrow indicates the transition relation, a dashed orange line indicates that states are related by B)

Definition 8 (Well-founded Skipping (WFSK) [3])**.** $B \subseteq S \times S$ is a well-founded skipping relation on TS $\mathcal{M} = \langle S, \rightarrow, L \rangle$ iff :

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

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

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

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

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

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

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

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 **([3]).** 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 $\delta(k+1)$, $\mathit{rankl}(\delta(k+1), s, u) < \mathit{rankl}(\delta.k, s, u)$, which contradicts the well-foundedness of *rankl*. We also have that for every $k > 0$, $u \not\!\!B\ \delta.k$; 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.

□

3.4.3 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(3.1). 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.

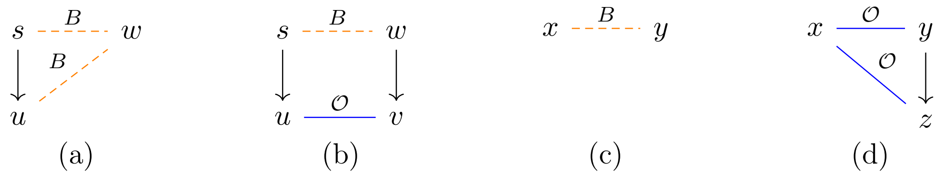

Figure 3.4.3: Reduced local well-founded skipping simulation (a solid line with arrow indicates the transition relation, a dashed orange line indicates that states are related by B and a solid blue line indicate the states are related by $\mathcal{O}$)

Definition 9 (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, $\mathit{rankt} : S \times S \longrightarrow W$, $\mathit{rankls} : S \times S \longrightarrow \omega$ 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) $(uBw \wedge \mathit{rankt}(u, w) \prec \mathit{rankt}(s, w)) \vee$

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

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 $\mathit{rankls}(u, v)$ 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 $x \not\!B\ y$. 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 $\mathit{rankls}(y, x) = |\vec{y}|$. 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 $x \not\!B\ y$. Let y' be a successor of y in $\vec{y}$. Clearly, $x\mathcal{O}y'$; therefore, $\mathit{rankls}(y', x) < |\vec{y}| - 1$, 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.

□

3.4.4 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 10 (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, $\mathit{rankt} : S \times S \longrightarrow W$, $\mathit{rankl} : S \times S \times S \longrightarrow \omega$, and $\mathit{rankls} : S \times S \longrightarrow \omega$ 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) $(uBw \wedge \mathit{rankt}(u, w) \prec \mathit{rankt}(s, w)) \vee$

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

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

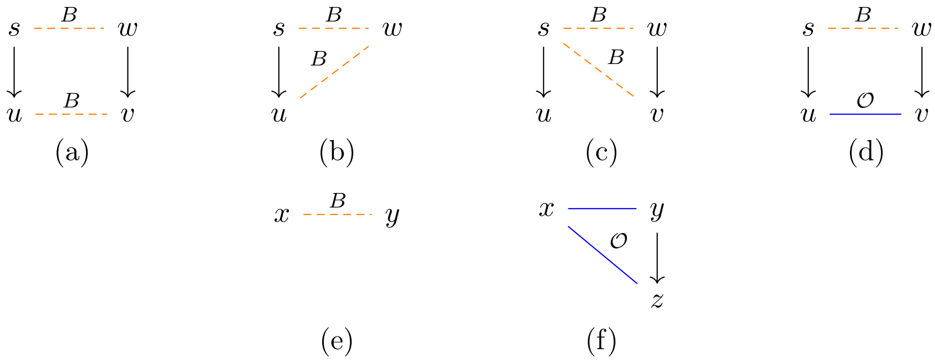

Figure 3.4.4: Local well-founded skipping simulation (a solid line with arrow indicates the transition relation, a dashed orange line indicates that states are related by B and a solid blue line indicate the states are related by $\mathcal{O}$)

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 RLWFSK2 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 9) and LWFSK (Definition 10) 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 **??**.

3.5 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 [8, 2]. Finally, we developed four alternative characterizations of skipping simulation that are amenable for mechanised reasoning using formal-methods tools.

In Chapter **??**, 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.

Bibliography

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

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

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

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

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

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

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

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