A Simple Characterization of Stuttering Bisimulation

Kedar S. Namjoshi*

Department of Computer Sciences,
The University of Texas at Austin, U.S.A.

Abstract. Showing equivalence of two systems at different levels of abstraction often entails mapping a single step in one system to a sequence of steps in the other, where the relevant state information does not change until the last step. In [BCG 88,dNV 90], bisimulations that take into account such "stuttering" are formulated. These definitions are, however, difficult to use in proofs of bisimulation, as they often require one to exhibit a finite, but unbounded sequence of transitions to match a single transition; thus introducing a large number of proof obligations.
We present an alternative formulation of bisimulation under stuttering, in terms of a ranking function over a well-founded set. It has the desirable property, shared with strong bisimulation [Mil 90], that it requires matching *single* transitions only, which considerably reduces the number of proof obligations. This makes proofs of bisimulation short, and easier to demonstrate and understand. We show that the new formulation is equivalent to the original one, and illustrate its use with non-trivial examples that have infinite state spaces and exhibit unbounded stuttering.

1 Introduction

Showing equivalence between two systems at different levels of abstraction may entail mapping a single step in one system to a sequence of steps in the other, which is defined with a greater amount of detail. For instance, a compiler may transform the single assignment statement "$x := x * 10 + 2$" to several low-level instructions. When proving correctness of the compiler, the single assignment statement step is matched with a sequence of low-level steps, in which the value of x remains unchanged until the final step. If the program state is defined by the values of program variables, then the intermediate steps introduce a finite repetition of the same state, a phenomenon called "stuttering" by Lamport [La 80]. Stuttering arises in various contexts, especially as a result of operations that hide information, or refine actions to a finer grain of atomicity.

* This work was supported in part by SRC Contract 96-DP-388. The author can be reached at kedar@cs.utexas.edu. A preliminary version of this paper appeared at the 17th Conference on Foundations of Software Technology and Theoretical Computer Science, December 1997.

In [BCG 88,dNV 90], bisimulations that take into account such "stuttering" are defined. It is shown in [BCG 88] that states related by a stuttering bisimulation satisfy the same formulas of the powerful branching temporal logic CTL^* [EH 82] that do not use the next-time operator, X. Although these definitions are well suited to showing the relationship with CTL^*, they are difficult to use in proofs of bisimulation, as they often require one to exhibit a finite, but unbounded sequence of transitions to match a single transition; thus introducing a large number of proof obligations.

The main contribution of this paper is a simple alternative formulation, called *well-founded bisimulation*, because is based on the reduction of a rank function over a well-founded set. The new formulation has the pleasant property that, like strong bisimulation [Mil 90], it can be checked by considering *single* transitions only. This substantially reduces the number of proof obligations, which is highly desirable in applications to infinite state systems such as communication protocols with unbounded channels or parameterized protocols, where checks of candidate relations are often performed by hand or with the assistance of a theorem prover. We demonstrate the use of the new formulation with some non-trivial examples that have infinite state spaces and exhibit unbounded stuttering.

The use of rank functions and well-founded sets is inspired by their use in replacing operational arguments for termination of **do-od** loops with a proof rule that is checked for a single generic iteration (cf. [AO 91]). To the best of our knowledge, this is the first use of such concepts in a bisimulation definition. It seems possible that the ideas in this paper are applicable to other forms of bisimulation under stuttering, such as weak bisimulation [Mil 90], and branching bisimulation [GW 89]. We have chosen to focus on stuttering bisimulation because of its close connection to CTL^*.

The paper is structured as follows: Section 2 contains the definition of stuttering bisimulation from [BCG 88], and the definition of well-founded bisimulation. The equivalence of the two formulations is shown in Section 3. Applications of the well-founded bisimulation proof rule to the alternating bit protocol and token ring protocols are presented in Section 4, together with a new quotient construction for stuttering bisimulation equivalences. The paper concludes with a discussion of related work and future directions.

2 Preliminaries

Notation :

Function application is denoted by a ".", i.e., for a function $f : A \to B$, and an element $a \in A$, $f.a$ is the value of f at a. Quantified expressions are written in the format $(\mathbf{Q}x : r.x : p.x)$, where $\mathbf{Q}$ is the quantifier (one of $\forall, \exists, min, max$), x is the "dummy", $r.x$ is an expression indicating the range of x, and $p.x$ is the expression being quantified over. For example, in this notation, $\forall x\ r(x) \Rightarrow p(x)$ is written as $(\forall x : r.x : p.x)$, and $\exists x\ r(x) \land p(x)$ is written as $(\exists x : r.x : p.x)$.

Definition (Transition System)

A Transition System (TS) is a structure $(S, \rightarrow, L, I, AP)$, where S is a set of *states*, $\rightarrow \subseteq S \times S$ is the *transition relation*, AP is the set of *atomic propositions*, $L : S \rightarrow \mathcal{P}(AP)$ is the *labelling function*, that maps each state to the subset of atomic propositions that hold at the state, and I is the set of *initial states*. We write $s \rightarrow t$ instead of $(s,t) \in \rightarrow$. We only consider transition systems with denumerable branching, i.e., where for every state s, $|\{t \mid s \rightarrow t\}|$ is at most ω.
$\square$

Definition (Stuttering Bisimulation) (cf. [BCG 88][1])

Let $\mathcal{A} = (S, \rightarrow, L, I, AP)$ be a TS. A relation $B \subseteq S \times S$ is a stuttering bisimulation on $\mathcal{A}$ iff B is symmetric, and

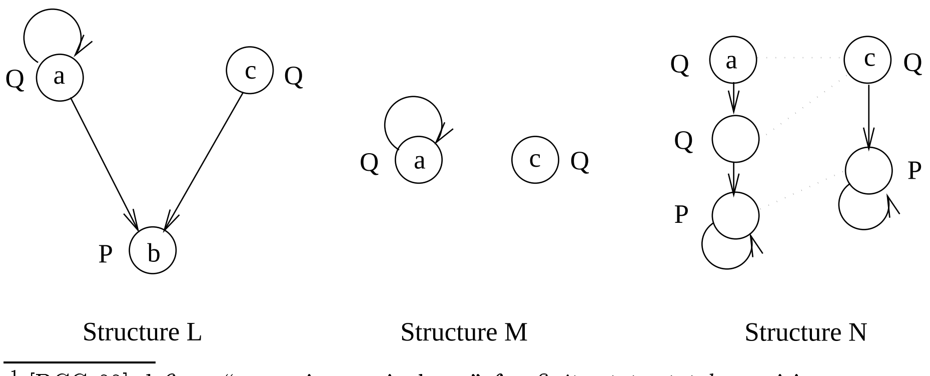
For every s, t such that $(s,t) \in B$,

1. $L.s = L.t$,
2. $(\forall \sigma : \mathit{fp}.(s,\sigma) : (\exists \delta : \mathit{fp}.(t,\delta) : \mathit{match}.B.(\sigma,\delta)))$.

where $\mathit{fp}.(s,\sigma)$ is true iff σ is a path starting at s, which is either infinite, or its last state has no successors w.r.t. $\rightarrow$. $\mathit{match}.B.(\sigma,\delta)$ is true iff σ and δ can be divided into an equal number of non-empty, finite, segments such that any pair of states from segments with the same index is in the relation B. The formal definition of *match* is given in the appendix.

States s and t are *stuttering bisimilar* iff there is a stuttering bisimulation relation B for which $(s,t) \in B$.
$\square$

Examples:



[1] [BCG 88] defines "stuttering equivalence" for *finite-state, total* transition systems, as the limit of a converging sequence of equivalences. For finite-state systems, these are just the Knaster-Tarski approximations to the greatest solution of the symmetric version of this definition.

States a and c are not stuttering bisimilar in structures L and M, but they are in structure N. Indeed, $L, c \models \mathsf{AF}.P$, but $L, a \not\models \mathsf{AF}.P$. Structure M shows that stuttering bisimulation distinguishes between deadlock (state c) and divergence (state a) : $M, c \not\models \mathsf{EX}.\mathit{true}$, but $M, a \models \mathsf{EX}.\mathit{true}$ [2]. The dotted lines show a stuttering bisimulation on structure N.
□

Our alternative formulation is based on a simple idea from program semantics: we define a mapping from states to a well-founded set, and require, roughly, that the mapping decrease with each stuttering step. Thus, each stuttering segment is forced to be of finite length, which makes it possible to construct matching fullpaths from related states.

Definition (Well-Founded Bisimulation)

Let $\mathcal{A} = (S, \rightarrow, L, I, AP)$ be a TS. Let $\mathit{rank} : S \times S \times S \rightarrow W$ be a total function, where $(W, \prec)$ is well-founded[3]. A relation $B \subseteq S \times S$ is a well-founded bisimulation on $\mathcal{A}$ w.r.t. *rank* iff B is symmetric, and

For every s, t such that $(s, t) \in B$,

1. $L.s = L.t$
2. $(\forall u : s \rightarrow u :$
   $(\exists v : t \rightarrow v : (u, v) \in B) \vee$ (a)
   $((u, t) \in B \wedge \mathit{rank}.(u, u, t) \prec \mathit{rank}.(s, s, t)) \vee$ (b)
   $((u, t) \notin B \wedge (\exists v : t \rightarrow v : (s, v) \in B \wedge \mathit{rank}.(u, s, v) \prec \mathit{rank}.(u, s, t))))$ (c)

Notice that if W is a singleton, then clauses (b) and (c) are not applicable, so B is a strong bisimulation.
□

The intuition behind this definition is that when $(s, t) \in B$ and $s \rightarrow u$, either there is a matching transition from t (clause (2a)), or $(u, t) \in B$ (clause (2b)) - in which case the rank decreases, allowing (2b) to be applied only a finite number of times - or $(u, t) \notin B$, in which case (by clause (2c)), there must be a successor v of t such that $(s, v) \in B$. As the rank decreases at each application of (2c), clause (2c) can be applied only a finite number of times. Hence, eventually, a state related to u by B is reached. Theorem 1 (soundness) is proved along these lines.

3 Equivalence of the two formulations

The equivalence of the two formulations is laid out in the following theorems.

---

[2] The [dNV 90] formulation of stuttering bisimulation considers states a and c of N to be bisimilar. The difference between our formulations is only in the treatment of deadlock vs. divergence in non-total structures.

[3] $(W, \prec)$ is well-founded iff there is no infinite subset $\{a.i \mid i \in \mathbf{N}\}$ of W that is a strictly decreasing chain, i.e. where for all $i \in \mathbf{N}$, $a.(i+1) \prec a.i$.

Theorem 1 (Soundness). *Any well-founded bisimulation on a TS is a stuttering bisimulation.*

Proof. Let B be a well-founded bisimulation on a TS $\mathcal{A}$, w.r.t. a function *rank* and a well-founded structure $(W, \prec)$.

Let (s,t) be an arbitrary pair in B. Then, $L.s = L.t$, by clause (1) of the well-founded bisimulation definition. We show that if σ is a fullpath starting at s, then there is a fullpath δ starting at t such that $match.B.(\sigma, \delta)$ holds. In the following, we use the symbol ';' for concatenation of finite paths, and $\circ$ for concatenation with removal of duplicate state. For example, $aa; ab = aaab$, and $aa \circ ab = aab$.

We construct δ inductively. For the base case, $\delta.0 = t$. Inductively assume that after i steps, $i \geq 0$, δ has been constructed to the point where it matches a prefix γ of σ such that the end states of γ and δ mark the beginning of the ith segments. Let u be the last state of γ and v be the last state of δ. By the inductive hypothesis, $(u,v) \in B$.

If σ ends at u, then u has no successor states. Let ξ be any fullpath starting at v. Since u has no successors, a simple induction using (2b) shows that for every state x in ξ, (x,u) is in B. Each application of (2b) strictly decreases *rank* along ξ, hence ξ must be finite. The fullpath $\delta \circ \xi$ is a finite fullpath matching the finite fullpath σ.

If σ does not end at u, let w be the successor of u in σ. As $(u,v) \in B$,

(i) If (2a) holds, there is a successor x of v such that $(w,x) \in B$. Let w and x mark the beginning of a new segment. Extend δ to $\delta; x$, which matches $\gamma; w$. The induction step is proved. Otherwise,

(ii) If (2a) does not hold, but (2b) does, then $(w,v) \in B$. Let λ be the longest prefix of the suffix of σ starting at u such that for every state a in λ, $(a,v) \in B$, and only (2b) holds for (a,v) w.r.t. $a \rightarrow b$ for every successive pair of states $a; b$ in λ. λ has at least one pair, as $u; w$ is a prefix of λ.

λ cannot be infinite, as by (2b), for each successive pair $a; b$ in λ, $rank.(b,b,v) \prec rank.(a,a,v)$, so the rank decreases strictly in the well-founded set. Let y be the last state of λ. If σ terminates at y, the argument given earlier applies. Otherwise, y has a successor y' in σ, but as λ is maximal, either (2a) or (2c) must apply for $(y,v) \in B$ w.r.t. $y \rightarrow y'$. (2c) cannot apply, as then there is a successor x of v such that $(y,x) \in B$, which contradicts the properties of λ.

Hence (2a) must apply. Let x be the successor of v such that $(y',x) \in B$. Let y' and x mark the beginning of a new segment, and extend δ to $\delta; x$, which matches $(\gamma \circ \lambda); y'$.

(iii) If (2c) is the only clause that holds of (u,v) w.r.t. $u \rightarrow w$, let π be a finite path maximal w.r.t. prefix ordering such that π starts at v, and for every successive pair of states $a; b$ in π, $(u,a) \in B$, only (2c) is applicable w.r.t. $u \rightarrow w$, and b is the successor of a given by the application of (2c).

Such a maximal finite path exists as, otherwise, there is an infinite path ξ satisfying the conditions above. By (2c), for successive states $a; b$ in ξ, $rank.(w, u, b) \prec rank.(w, u, a)$; so there is an infinite strictly decreasing chain in $(W, \prec)$, which contradicts the well-foundedness of $(W, \prec)$. Let x be the last state in π. Then $(u, x) \in B$, and as π is maximal, either (2a) or (2b) holds of (u, x) w.r.t. $u \rightarrow w$. So $x \neq v$. (2b) cannot hold, as then (w, x) is in B; but then (2a) would hold for the predecessor of x in π.

Hence (2a) holds; so x has a successor z for which $(w, z) \in B$. Let w and z mark the beginning of a new segment, and extend δ to $(\delta \circ \pi); z$, which matches $\gamma; w$.

The induction step is shown in either case.

The inductive argument shows that successively longer prefixes of σ have successively longer matching finite paths, which are totally ordered by prefix order. Hence, if σ is infinite, the limit of these matching paths is an infinite path from t which matches σ using the partitioning into finite non-empty segments constructed in the proof. $\square$

It is also desirable to have completeness : that for every stuttering bisimulation, there is a rank function over a well-founded set which gives rise to a well-founded bisimulation.

Theorem 2 (Completeness). *For any stuttering bisimulation B on a TS $\mathcal{A}$, there is a well-founded structure $(W, \prec)$ and corresponding function rank such that B is a well-founded bisimulation on $\mathcal{A}$ w.r.t. rank.* $\square$

Let $\mathcal{A} = (S, \rightarrow, L, I, AP)$. The well-founded set W is defined as the product $W_0 \times W_1$ of two well-founded sets, with the new ordering being lexicographic order. The definitions of the well-founded sets W_0 and W_1, and associated functions $rank_0$ and $rank_1$ are given below. Informally, $rank_0.(a, b)$ measures the height of a finite-depth computation tree rooted at a, whose states are related to b but not to any successor of b. $rank_1.(a, b, c)$ measures the shortest finite path from c that matches b and ends in a state related to the successor a of b.

Definition of $(W_0, \prec_0)$ and $rank_0$

For a pair (s, t) of states of $\mathcal{A}$, construct a tree, *tree*.(s, t), by the following (possibly non-effective) procedure, which is based on clause (2b) of the definition of well-founded bisimulation:

1. The tree is empty if the pair (s, t) is not in B. Otherwise,
2. s is the root of the tree. The following invariant holds of the construction: For any node y of the current tree, $(y, t) \in B$, and if y is not a leaf node, then for every child z of y in the tree, z is a successor of y in $\mathcal{A}$, and there is no successor v of t in $\mathcal{A}$ such that $(z, v) \in B$.
3. For a leaf node y, and any successor z of y in $\mathcal{A}$, if $(z, t) \in B$, but there is no successor v of t in $\mathcal{A}$ such that $(z, v) \in B$, then add z as a child of y in the tree. If no such successor exists for y, then terminate the branch at y. Repeat step 3 for every leaf node on an unterminated branch.

Lemma 3. *$tree.(s,t)$ is well-founded.*

Proof. Suppose to the contrary that there is an infinite branch σ, which is therefore a fullpath, starting at s. Let u be the successor of s on σ, and let σ' be the fullpath that is the suffix of σ starting at u.

By construction of the tree, for every state x on σ', $(x,t) \in B$, and for every successor v of t, $(x,v) \notin B$. However, as $(u,t) \in B$, there must be a fullpath δ starting at t for which $match.B.(\sigma',\delta)$ holds. Let w be the successor of t on δ. From the definition of *match*, for some x on σ', $(x,w) \in B$. This is a contradiction. Hence, every branch of the tree must be of finite length. □

Since $tree.(s,t)$ is well-founded, it can be assigned an ordinal height using a standard bottom-up assignment technique for well-founded trees : assign the empty tree height 0, and any non-empty tree T the ordinal $sup.\{height.S + 1 \mid S \lhd T\}$, where $S \lhd T$ holds iff S is a strict subtree of T. Let $rank_0.(s,t)$ equal the height of $tree.(s,t)$. As trees with countable branching need only countable ordinals as heights, let W_0 be the set of countable ordinals, ordered by the inclusion order $\in$.

Lemma 4. *If $tree.(s,t)$ is non-empty, and u is a child of s in the tree, then $rank_0.(u,t) \prec_0 rank_0.(s,t)$.*

Proof. From the construction, $tree.(u,t)$ is the subtree of $tree.(s,t)$ rooted at node u; hence its height is strictly smaller. □

Definition of $(W_1, \prec_1)$ and $rank_1$

Let $W_1 = \mathbf{N}$, the set of natural numbers, and let $\prec_1$ be the usual order $<$ on $\mathbf{N}$. The definition of $rank_1$ is as follows :

For a tuple (u,s,t) of states of $\mathcal{A}$,

1. If $(s,t) \in B$, $s \to u$, $(u,t) \notin B$, and for every successor v of t, $(u,v) \notin B$, then $rank_1.(u,s,t)$ is the length of the shortest initial segment that matches s among all matching fullpaths $s;\sigma$ and δ, where σ starts at u, and δ starts at t. Formally [4],

   $$\begin{aligned} rank_1.(u,s,t) = (min\ \delta,\xi,\sigma,\pi : fp.(t,\delta) \wedge fp.(u,\sigma) \wedge \pi,\xi \in INC \wedge \\ corr.((s;\sigma,\pi),(\delta,\xi)) : |seg.0.(\delta,\xi)|) \end{aligned}$$

   As $(s,t) \in B$, and $s \to u$, there exist matching fullpaths $s;\sigma$ and δ, with σ starting at u and δ starting at t. As $(u,t) \notin B$, and no successor of t matches u, under any partition ξ of any fullpath δ that matches a fullpath $s;\sigma$, the initial segment, $seg.0.(\delta,\xi)$, matches s, and must contain at least two states: t and some successor of t. Thus, $rank_1.(u,s,t)$ is defined, and is at least 2.

[4] The appendix has precise definitions of *INC* and *corr*.

2. Otherwise, $rank_1.(u,s,t) = 0$.

□

Theorem 2 (Completeness). *For any stuttering bisimulation B on TS $\mathcal{A}$, there is a well-founded set $(W, \prec)$ and corresponding function rank such that B is a well-founded bisimulation on $\mathcal{A}$ w.r.t. rank.*

Proof. Let $W = W_0 \times W_1$. The ordering $\prec$ on W is the lexicographic ordering on $W_0 \times W_1$, i.e., $(a,b) \prec (c,d) \equiv (a \prec_0 c) \vee (a = c \wedge b \prec_1 d)$. Define $rank.(u,s,t) = (rank_0.(u,t), rank_1.(u,s,t))$. W is well-founded, and *rank* is a total function. We have to show that B is a well-founded bisimulation w.r.t. *rank*. Let $(s,t) \in B$.

1. $L.s = L.t$, from the definition of stuttering bisimulation.
2. Let u be any successor of s. If there is no successor v of t such that $(u,v) \in B$, consider the following cases:
   - $(u,t) \in B$: As no successor of t is related to u by B, u is a child of s in $tree.(s,t)$, and by Lemma 4, $rank_0.(u,t) \prec_0 rank_0.(s,t)$. Hence, $rank.(u,u,t) \prec rank.(s,s,t)$.
   - $(u,t) \notin B$: As no successor of t is related to u by B, $rank_1.(u,s,t)$ is non-zero. Let fullpath δ starting at t and partition ξ "witness" the value of $rank_1.(u,s,t)$. Let v be the successor of t in the initial segment $seg.0.(\delta,\xi)$. This successor exists, as the length of the segment is at least 2. $rank_1.(u,s,v)$ is at most $rank_1.(u,s,t) - 1$, so $rank_1.(u,s,v) \prec_1 rank_1.(u,s,t)$.
     As no successor of t is related by B to u, $(u,v) \notin B$, so $rank_0.(u,v) = 0$. As $(u,t) \notin B$, $rank_0.(u,t) = 0$. Since *rank* is defined by lexicographic ordering, $rank.(u,s,v) \prec rank.(u,s,t)$.

   Hence, one of (2a),(2b) or (2c) holds for $(s,t) \in B$ w.r.t. $s \rightarrow u$.

□

For a transition system that is *finite-branching* (every state has finitely many successor states), $tree.(s,t)$ for any s,t is a finite, finitely-branching tree; so its height is a natural number. Hence, $W_0 = \mathbf{N}$.

Proposition 5. *For a finite-branching transition system, $W = \mathbf{N} \times \mathbf{N}$.* □

Theorem 6 (Main). *Let $\mathcal{A} = (S, \rightarrow, L, I, AP)$ be a transition system. A relation B on $\mathcal{A}$ is a stuttering bisimulation iff B is a well-founded bisimulation w.r.t. some rank function.*

Proof. The claim follows immediately from Theorems 1 and 2. □

For simplicity, the definitions are structured so that a bisimulation is a symmetric relation. The main theorem holds for bisimulations that are not symmetric, but the definition of rank has to be modified slightly, to take the direction of matching (by B or by B^{-1}) into account. Details will appear in the full paper.

4 Applications

The definition of a well-founded bisimulation is, by Theorem 6, in itself a simple proof rule for determining if a relation is indeed a bisimulation up to stuttering. In this section, we look at several applications of this proof rule. We outline the proofs of well-founded bisimulation for the alternating bit protocol from [Mil 90], and a class of token ring protocols studied in [EN 95]. We also present a new quotient construction for a well-founded bisimulation that is an equivalence. In all of these applications, the construction of the appropriate well-founded set and ranking function is quite straightforward. We believe that this is the case in other applications of stuttering bisimulation as well.

4.1 The Alternating Bit Protocol

A version of the alternating bit protocol is given in [Mil 90], which we follow closely. The protocol has four entities : *Sender* and *Replier* processes, and message (*Trans*) and acknowledgement (*Ack*) channels. Messages and acknowledgements are tagged with bits 0 and 1 alternately. For simplicity, message contents are ignored; both channels are sequences of bits. For a channel c, let $order.c$ represent the sequence resulting from removing duplicates from c, and let $count.c$ be a vector of the numbers of duplicate bits. Vectors are compared component-wise if they have the same length. For example, $order.(0^3;1^2) = 0;1$, $count.(0^3;1^2)$ $= (3,2)$, and $count.(1^5) = (5)$.

Let $(s,t) \in B$ iff in s and t, the local states of the *sender* and *replier* processes are identical, and the *order* of messages in both channels is the same. Note that the number of duplicate messages is abstracted away.

Let $\alpha.s = (count.(Trans.s), count.(Ack.s))$. Let $rank.(u,s,t)$ be $(\alpha.s, \alpha.t)$. The operations of the protocol are sending a bit or receiving a bit on either channel, and duplicating or deleting a bit on either channel. It is straightforward to verify that B is a well-founded bisimulation. The rank function is used, for instance, at a receive action in s from a channel with contents $a^l;b$, while the same channel in the corresponding state t has contents $a^m;b^n$ $(n > 1)$. The receive action at s results in a state u with channel content a^l, while the same action at t results in a state v with channel content $a^m;b^{n-1}$. u and v are not related, but v is related to s, and $rank.(u,s,v) < rank.(u,s,t)$ (cf. clause (2c)).

The example exhibits unbounded stuttering. With the original formulations of stuttering bisimulation, one would have to construct a computation of length n from state t to match the receive action from state s. This introduces n proof obligations, and complicates the proof. In contrast, with the new formulation, one need consider only a single transition from t.

Although the bisimulation B is an equivalence, it has an infinite number of equivalence classes. For the protocol, however, the initial state is that with empty channels, and it is possible to show that, although the reachable state space is still infinite, the reachable states of the protocol have channels with *order* values of length at most 2. Thus, B induces a *finite* partition of the reachable state space. This fact can be exploited to model check the properties of the protocol, as described subsequently.

4.2 Simple Token Ring Protocols

In [EN 95] (cf. [BCG 89]), stuttering bisimulation is used to show that for token rings of similar processes, a small *cutoff* size ring is equivalent to one of any larger size. [EN 95] shows that the computation trees of process 0 in rings of size 2 and of size n, $n \geq 2$, are stuttering bisimilar. It follows that a property over process 0 is true of *all* sizes of rings iff it is true of the ring of size 2. From symmetry arguments (cf. [ES 93,CFJ 93]), a property holds of all processes iff it holds for process 0.

The proof given in the paper uses the [BCG 88] definition and is quite lengthy; we indicate here how to use well-founded bisimulation. Each process alternates between blocking receive and send token transfer actions, with a finite number of local steps in between. For an n-process system with state space S_n, define $\alpha_n : S_n \rightarrow \mathbf{N}^2$ as the function given by $\alpha_n.s = (i, j)$ where, in state s, if process m has the token, then $i = (n - m)$ *mod* n is the distance of of the token from process 0, and j is the sum over processes of the maximum number of steps of each process from its local state to the first token transfer action. The tuples are ordered lexicographically. Let the rank function be $rank.(u, s, t) = (\alpha_m.s, \alpha_n.t)$, where s and t are states in instances with m and n processes respectively. Let the relation B be defined by $(s, t) \in B$ iff the local state of process 0 is identical in s and t.

It is straightforward to verify that B is a well-founded bisimulation w.r.t. *rank*. The rank function is used in the situation where the token is received by process 0 by a move from state s to state u; however, the reception action is not enabled for process 0 in a state t related to s by B. In this case, some move of a process other than 0 is enabled at t, and results in a state v that reduces α_n, and hence the rank, either by a transfer of the token to the next process, or by reducing the number of steps to the first token transfer action. The next state v is related to s by B (cf. clause (2c) of the definition).

4.3 Quotient Structures

For a bisimulation B on TS $\mathcal{A}$ that is an equivalence relation, a *quotient structure* $\mathcal{A}/B$ (read as $\mathcal{A}$ "mod" B) can be defined, where the states are equivalence classes (w.r.t. B) of states of $\mathcal{A}$, and the new transition relation is derived from the transition relation of $\mathcal{A}$. Quotient structures are usually much smaller than the original; a bisimulation with finitely many classes induces a finite quotient, as is the case in the examples given in the previous sections.

Let $\mathcal{A} = (S, \rightarrow, L, I, AP)$ be a TS, and B be a well-founded bisimulation on $\mathcal{A}$ w.r.t. a rank function α, that is an equivalence relation on S. The equivalence class of a state s is denoted by $[s]$. Define $\mathcal{A}/B$ as the TS $(\mathcal{S}, \rightsquigarrow, \mathcal{L}, \mathcal{I}, AP)$ given by :

- $\mathcal{S} = \{[s] \mid s \in S\}$
- The transition relation is given by : For $C, D \in \mathcal{S}$, $C \rightsquigarrow D$ iff either
  1. $C \neq D$, and $(\exists s, t : s \in C \wedge t \in D : s \rightarrow t)$, or
  2. $C = D$, and $(\forall s : s \in C : (\exists t : t \in C : s \rightarrow t))$.

  The distinction between the two cases is made in order to prevent spurious self-loops in the quotient, arising from stuttering steps in the original.
- The labelling function is given by $\mathcal{L}.C = L.s$, for some s in C. (states in an equivalence class have the same label)
- The set of initial states, $\mathcal{I}$, equals $\{[s] \mid s \in I\}$.

Theorem 7. *$\mathcal{A}$ is stuttering bisimilar to $\mathcal{A}/B$.*

Proof. Form the disjoint union of the TS's $\mathcal{A}$ and $\mathcal{A}/B$. The bisimulation on this structure relates states of $\mathcal{A}$ and $\mathcal{A}/B$ as follows : $(a, b) \in R$ iff $[a] = b \vee [b] = a$.

Let $sw : \mathcal{S} \rightarrow S$ (read "state witness") be a partial function, defined at C only when $C \rightsquigarrow C$ does not hold. When defined, $v = sw.C$ is such that $v \in C$, but no successor of v w.r.t. $\rightarrow$ is in C. Such a v exists by the definition of $\rightsquigarrow$. Let $ew : \mathcal{S}^2 \rightarrow S^2$ (read "edge witness") be a partial function, defined at (D, C) iff $C \rightsquigarrow D$. When defined, $(v, u) = ew.(D, C)$ is such that $u \in C, v \in D$, and $u \rightarrow v$.

Let *rank* be a function defined on $W \cup \{\bot\}$ ($\bot$ is a new element unrelated to any elements of W) by : If $u, s \in S$, and $sw.C$ is defined, then $rank.(u, s, C) = \alpha.(u, s, sw.C)$. If $D, C \in \mathcal{S}$ and $s \in S$, then $rank.(D, C, s) = \alpha.(ew.(D, C), s)$, if $ew.(D, C)$ is defined. Otherwise, $rank.(a, b, c) = \bot$.

Let $(a, b) \in R$. From the definition of R, a and b have the same label.

- $a \in S$: For clarity, we rename (a, b) to (s, C). By the definition of R, $C = [s]$. Let $s \rightarrow u$. If $[s] \rightsquigarrow [u]$, then there is a successor $D = [u]$ of C such that $(u, D) \in R$, and clause (2a) holds.
  If the edge from $[s]$ to $[u]$ is absent, then $[s]$ must equal $[u]$, and $sw.C$ is defined. Let $x = sw.C$. As $(s, x) \in B$, and $(u, x) \in B$, but x has no successors to match u, clause (2b) holds for B, i.e., $\alpha.(u, u, x) \prec \alpha.(s, s, x)$. By definition of *rank*, $rank.(u, u, C) \prec rank.(s, s, C)$, so (2b) holds for R.
- $a \in \mathcal{S}$: For clarity, we rename (a, b) to (C, s). Let $C \rightsquigarrow D$. Let $(y, x) = ew.(D, C)$. As $x \rightarrow y$, and $(x, s) \in B$, there are three cases to consider :
  1. There is a successor u of s such that $(y, u) \in B$. Then $[y] = [u]$, so $(D, u) \in R$, and (2a) holds.
  2. $(y, s) \in B$. Then $[y] = [x]$, so $C = D$. As $C \rightsquigarrow D$, and $s \in C$, s has a successor u such that $u \in C$; hence (D, u) is in R and (2a) holds.

3. $(y, s) \notin B$ and there exists u such that $s \ \rightarrow \ u$, $(x, u) \in B$, and $\alpha.(y, x, u) \prec \alpha.(y, x, s)$. Hence, $(C, u) \in R$, and $rank.(D, C, u) \prec rank.(D, C, s)$. So clause (2c) holds.

$\square$

5 Related Work and Conclusions

Other formulations of bisimulation under stuttering have been proposed; however, they too involve reasoning about finite, but unbounded sequences of transitions. Examples include branching bisimulation [GW 89], divergence sensitive stuttering [dNV 90], and weak bisimulation [Mil 90]. We believe that it is possible to characterize branching bisimulation in a manner similar to our characterization of stuttering bisimulation, given the close connection between the two that is pointed out in [dNV 90]. An interesting question is whether a similar characterization can be shown for weak bisimulation [Mil 90].

Many proof rules for temporal properties are based on well-foundedness arguments, especially those for termination of programs under fairness constraints (cf. [GFMdR 83,Fr 86,AO 91]). Vardi [Va 87], and Klarlund and Kozen [KK 91] develop such proof rules for very general types of linear temporal properties. Our use of well-foundedness arguments for defining a bisimulation appears to be new, and, we believe, of intrinsic mathematical interest. The motivation in each of these instances is the same : to replace reasoning about unbounded or infinite paths with reasoning about single transitions.

Earlier definitions of stuttering bisimulation are difficult to apply to large problems essentially because of the difficulty of reasoning about unbounded stuttering paths. Our new characterization, which replaces such reasoning with reasoning about single steps, makes proofs of equivalence under stuttering easier to demonstrate and understand. In the example applications, it was quite straightforward to determine an appropriate well-founded set and rank function. Indeed, rank functions are implicit in proofs that use the earlier formulations. As the examples demonstrate, using rank functions explicitly leads to proofs that are shorter, and which can be carried out with assistance from a theorem prover.

Acknowledgements. Thanks to Prof. E. Allen Emerson, Peter Manolios, Jun Sawada, Robert Sumners, and Richard Trefler for carefully reading an earlier draft of this paper. Peter Manolios helped to strengthen some of the theorems and simplify the proofs. The comments from the referees helped to improve the presentation.

References

[AO 91] Apt, K. R., Olderog, E-R. *Verification of Sequential and Concurrent Programs*, Springer-Verlag, 1991.

[BCG 88] Browne, M. C., Clarke, E. M., Grumberg, O. Characterizing Finite Kripke Structures in Propositional Temporal Logic, *Theor. Comp. Sci.*, vol. 59, pp. 115–131, 1988.

[BCG 89] Browne, M. C., Clarke, E. M., Grumberg, O. Reasoning about Networks with Many Identical Finite State Processes, *Information and Computation*, vol. 81, no. 1, pp. 13–31, April 1989.

[CFJ 93] Clarke, E.M., Filkorn, T., Jha, S. Exploiting Symmetry in Temporal Logic Model Checking, 5th CAV, Springer-Verlag LNCS 697.

[EH 82] Emerson, E. A., Halpern, J. Y. "Sometimes" and "Not Never" Revisited: On Branching versus Linear Time Temporal Logic. in *POPL*, 1982.

[EN 95] Emerson, E.A., Namjoshi, K.S. Reasoning about Rings. in *POPL*, 1995.

[ES 93] Emerson, E.A., Sistla, A.P. Symmetry and Model Checking, 5th CAV, Springer-Verlag LNCS 697.

[Fr 86] Francez, N. *Fairness*, Springer-Verlag, 1986.

[GW 89] van Glabbeek, R. J., Weijland, W. P. Branching time and abstraction in bisimulation semantics. in *Information Processing 89*, Elsevier Science Publishers, North-Holland, 1989.

[GFMdR 83] Grumberg, O., Francez, N., Makowski, J., de Roever, W-P. A proof rule for fair termination, in *Information and Control*, 1983.

[KK 91] Klarlund, N., Kozen, D. Rabin measures and their applications to fairness and automata theory. in *LICS*, 1991.

[La 80] Lamport, L. "Sometimes" is Sometimes "Not Never". in *POPL*, 1980.

[Mil 90] Milner, R. *Communication and Concurrency*, Prentice-Hall International Series in Computer Science. Edited by C.A.R. Hoare.

[dNV 90] de Nicola, R., Vaandrager, F. Three logics for branching bisimulation. in *LICS*, 1990. Full version in *Journal of the ACM*, 42(2):458-487, 1995.

[Va 87] Vardi, M. Verification of Concurrent Programs - The Automata Theoretic Framework. in *LICS*, 1987. Full version in *Annals of Pure and Applied Logic*, 51:79-98, 1991.

6 Appendix

Definition of *match*

Let INC be the set of strictly increasing sequences of natural numbers starting at 0. Precisely, $INC = \{\pi \mid \pi : \mathbf{N} \to \mathbf{N} \wedge \pi.0 = 0 \wedge (\forall i : i \in \mathbf{N} : \pi.i < \pi.(i+1))\}$. Let σ be a path, and π a member of INC. For $i \in \mathbf{N}$, let $intv.i.(\sigma,\pi) = [\pi.i, min.(\pi.(i+1), length.\sigma))$. The ith segment of σ w.r.t. π, $seg.i.(\sigma,\pi)$, is defined by the sequence of states of σ with indices in $intv.i.(\sigma,\pi)$.

Let σ and δ, under partitions π and ξ respectively, correspond w.r.t. B iff they are subdivided into the same number of segments, and any pair of states in segments with the same index are related by B. Precisely, $corr.B.((\sigma,\pi),(\delta,\xi)) \equiv (\forall i : i \in \mathbf{N} : intv.i.(\sigma,\pi) \neq \emptyset \equiv intv.i.(\delta,\xi) \neq \emptyset \wedge (\forall m,n : m \in intv.i.(\sigma,\pi) \wedge n \in intv.i.(\delta,\xi) : (\sigma.m, \delta.n) \in B)))$.

Paths σ and δ match iff there exist partitions that make them correspond. Precisely, $match.B.(\sigma,\delta) \equiv (\exists \pi,\xi : \pi,\xi \in INC : corr.B.((\sigma,\pi),(\delta,\xi)))$.
$\square$