Weighted bisimulation in linear algebraic form*

Michele Boreale
Università di Firenze

Abstract. We study bisimulation and minimization for weighted automata, relying on a geometrical representation of the model, linear weighted automata (LWA). In a LWA, the state-space of the automaton is represented by a vector space, and the transitions and weighting maps by linear morphisms over this vector space. Weighted bisimulations are represented by sub-spaces that are invariant under the transition morphisms. We show that the largest bisimulation coincides with weighted language equivalence, can be computed by a geometrical version of partition refinement and that the corresponding quotient gives rise to the minimal weighted-language equivalent automaton. Relationships with Larsen and Skou's probabilistic bisimulation and with classical results in Automata Theory are also discussed.

1 Introduction

We study bisimulation [17,18], quotients and minimization in finite-state weighted automata over the field of real numbers. The motivation of this study is twofold. In the past two decades, bisimulation has been adapted to several flavours of probabilistic and stochastic systems. There is a large body of literature on the subject, see e.g. [11,15,4,1,5] and references therein. These definitions are presented in different formats and notations and give rise, in general, to different equivalences, whose mutual relationships are difficult to asses. At the same time, one wonders if any alternative, linear-time equivalence exists that might replace bisimulation-based equivalences in certain situations. On ordinary LTS's, language equivalence, aka may testing [9], is known to be more generous than bisimulation and appropriate to reason on certain classes of properties, like safety ones. Unfortunately, it is also much more complex to decide than bisimilarity [16]. In practice, one often uses bisimulation as an incomplete proof technique for language equivalence. One wonders to what extent this state of things carries over to the weighted setting.

In an ordinary automaton, each state is associated with a recognized language. Likewise, in a weighted automaton, each state q is associated with a recognized *weighted* language $\sigma(q)$, that is, a set of words each coming with a weight (probability/multiplicity/cost...). Classically, weighted languages are known as *formal power series* [2]. Two states q and q' are weighted language

* Author's e-mail: boreale@dsi.unifi.it. Address: Dipartimento di Sistemi e Informatica, Viale Morgagni 65, I-50134, Firenze. Work partially supported by EU within the FET-GC2 initiative, project SENSORIA.

equivalent if $\sigma(q) = \sigma(q')$. It is worth to notice that, in the case of probabilistic transition systems, $\sigma(q)$ has a very natural interpretation: the weight of a word $x = a_1 \cdots a_n$ in $\sigma(q)$ is just the probability of observing the word x if the execution of the system starts from q. Or, to phrase it in the testing equivalence jargon [9], it is the probability that the system passes the test $\overline{a}_1.\cdots\overline{a}_n.\omega$ starting its execution from q. In other weighted settings, like the counting automata of [19], language equivalence enjoys an equally natural interpretation. Now, on ordinary automata, one way of computing the minimal language equivalent automaton is to first make the original automaton deterministic and then quotient the result of this operation by the largest bisimulation. It is also known that the first operation, the powerset construction, takes exponential time. One wonders what is an equivalent construction in the weighted setting. Or, in other words, what is, if any, the form of bisimulation underpinned by language-preserving minimization in weighted automata. Note that a polynomial weighted-language-preserving minimization procedure for weighted automata has been known for more than forty years [22]. This leads us to the second motivation for our study, that is, to clarify the connections of bisimulation for weighted transition systems and similar structures to classical results in Automata and Language Theory.

We undertake our study by first introducing a linear algebraic representation of weighted automata, *linear weighted automata* (LWA, Section 2). In the familiar representation, transitions of a weighted automaton can be viewed as maps taking each individual state into a set of states, each having its own weight. It is useful to view this set as a formal linear combination of states. It is then natural to extend the transition maps so that they take linear combinations to linear combinations (of states). This leads to the notion of LWA, where the state-space of an automaton is a vector space – the set of linear combinations of states – and the transitions are linear maps. In this formulation, it is natural to define a *linear weighted bisimulation* (Section 3) as a sub-space that is invariant under the transition maps and is included in the kernel of the weight function. This definition retains the nice coinductive proof technique found in ordinary bisimulation: to prove two stated related, it is sufficient to exhibit a "small" bisimulation relation containing them as a pair. We show that the largest linear weighted bisimulation equivalence exists and coincides with coincides with weighted language equivalence. Moreover, it can be effectively computed by a geometrical version of the partition refinement algorithm (Section 4). Or, more accurately, a basis of the corresponding sub-space can be effectively computed. The resulting algorithm is polynomial in the dimension, i.e. the number of states, of the underlying weighted automaton. We next show that taking the quotient of a LWA by a bisimulation corresponds, geometrically, to projecting the original state-space onto the *(orthogonal) complement* of the sub-space representing the bisimulation (Section 5). When the chosen bisimulation is the largest one, this operation results into a practical method for constructing the minimal language-equivalent LWA out of a given one.

The overall construction resembles that for ordinary automata, with determinization corresponding to building the LWA. The important difference is that

here there is no exponential blow-up involved. When we specialize this construction to automata with an initial state (Section 7), we re-discover essentially the original minimization algorithm proposed by Schützenberger [22]. The minimal form is canonical, in the sense that minimal LWA's representing the same weighted language are isomorphic. We also compare linear weighted bisimilarity to the probabilistic bisimulation of Larsen and Skou [15] and find the latter to be to be strictly finer than the former (Section 6).

Our work is related to recent and less recent work by Buchholz and Kemper [5,6], by Rutten [20,19,21] and by Stark [24], who have studied some of the issues we consider here (see Section 8).

In the paper, we will make use of a few concepts from elementary Linear Algebra, whose description can be found in any introductory textbook, such as [14]. Due to lack of space, most proofs have been omitted in this short version: they can be found in the full version available online [3].

2 Linear weighted automata

In the sequel, we fix a finite, non-empty alphabet A of *actions*. V will denote a finite-dimensional inner product space, that is a vector space over $\mathbb{R}$ equipped with a inner product $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ (in fact, inner product will not come into use until Section 4). We will often refer to elements of a vector space as "vectors"; the null vector will be always denoted by 0, as the context will be always sufficient to avoid confusion with the zero scalar. We shall often omit brackets surrounding function arguments, writing e.g. Tv rather than $T(v)$, unless this jeopardizes readability. "Homomorphism" is shortened as "morphism".

Definition 1 (weighted automaton in linear form). *A* linear weighted automaton *(*LWA*, for short) is a triple $L = (V, \{T_a\}_{a \in A}, \phi)$, where V is a inner product space over $\mathbb{R}$, and $T_a : V \to V$, for each $a \in A$, and $\phi : V \to \mathbb{R}$ are morphisms. The* dimension *of L is* $\dim(L) \triangleq \dim(V)$.

The three elements of a LWA L are referred to as: the *state-space* V, the *transition functions* T_a, $a \in A$, and the *(final) weight function* ϕ, respectively. We do not consider yet initial distributions on states, that will be the subject of Section 7. In the rest of the section, L will denote a generic LWA $(V, \{T_a\}_{a \in A}, \phi)$. A family of morphisms indexed over A, $\{T_a\}_{a \in A}$, induces a family of morphisms indexed over A^*, $\{T_x\}_{x \in A^*}$, defined as follows: for each $v \in V$, $T_\epsilon v \triangleq v$, $T_{ax} v \triangleq T_x T_a v$. Recall that a *formal power series* (FPS for short) over A and $\mathbb{R}$ is a function $\sigma : A^* \to \mathbb{R}$. We view a FPS as the same thing as a *weighted language*, a generalization of the usual notion of language where which word comes with a real multiplicity, possibly 0 to indicate absence.

Definition 2 (weighted language semantics). *The* weighted language *associated by L to any $v \in V$ is the* FPS $\sigma_L(v) : A^* \to \mathbb{R}$ *defined by: $\sigma_L(v)(x) \triangleq \phi(T_x v)$ for each $x \in A^*$. We say u and v are* weighted-language equivalent *if $\sigma_L(u) = \sigma_L(v)$.*

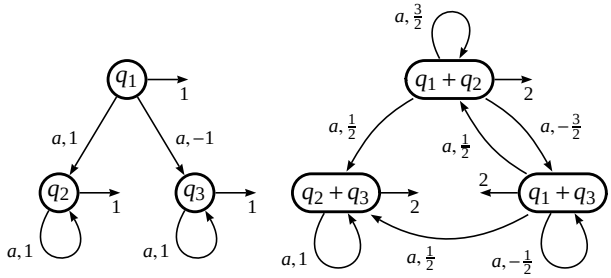

Fig. 1. Two weighted automata representing the same linear weighted automaton.

Weighted automata are often represented in matrix form. Ignoring for the moment initial states, a weighted automaton (WA, for short) W is often defined as a triple $(Q, \{M_a\}_{a\in A}, f)$, where: $Q = (q_1, ..., q_n)$ is an ordered finite set of states; each $M_a \in \mathbb{R}^{n\times n}$ is a real-valued square matrix, with $M_a(i,j)$ specifying the weight of the a-transition from q_j to q_i; and $f \in \mathbb{R}^{1\times n}$ is a real-valued row vector, describing the final weights assigned to the q_i's (see e.g. [5]). The weighted language semantics of W can be described as follows. Given an initial distribution on states specified by a column vector $s \in \mathbb{R}^{n\times 1}$, the FPS associated to s by W is given by: $\sigma_W(s)(x) \triangleq fM_x s$, where for $x = a_1 \cdots a_k$, M_x is the product matrix $M_{a_n} \cdots M_{a_1}$ (with $M_\epsilon = I$). The matrix representation corresponds, up to the ordering of states, to the familiar graphical representation (see the next example).

It should be evident that the matrix formulation is equivalent to the one given in Definition 1. More precisely, given a LWA L, by fixing an ordered basis $Q = (e_1, ..., e_n)$ of V one determines a WA $W_{L,Q} = (Q, \{M_a\}_{a\in A}, f)$, where M_a (resp. f) is the matrix (resp. row vector) representing T_a (resp. ϕ) in the basis Q. This correspondence preserves weighted-language semantics, that is, for each $v \in V$, $\sigma_L(v) = \sigma_{W_{L,Q}}(s)$, where s is the column vector of coordinates of v in Q. Conversely, a WA $W = (Q, \{M_a\}_{a\in A}, f)$ determines a LWA $L_W = (\mathbb{R}^Q, \{T_a\}_{a\in A}, \phi)$, by considering $\mathbb{R}^Q$ as the (free) vector space with the expected inner product $(g \cdot h \triangleq \sum_{q\in Q} g(q) \cdot h(q))$, and taking T_a (resp. ϕ) to be the linear morphism on $\mathbb{R}^Q$ represented by the matrix M_a (resp. row-vector f) in the basis[1] Q. Again, this correspondence preserves the weighted-language semantics. The two constructions are inverse of one another, e.g. one has $W_{L_W,Q} = W$.

Example 1. Let $A = \{a\}$ be a singleton alphabet and $Q = (q_1, q_2, q_3)$. Consider the WA $W = (Q, \{M_a\}, f)$ represented by the graph in Fig. 1, on the left. Transitions having weight 0 are not displayed. The standard convention is adopted to specify final weights: $(q_i) \longrightarrow r$ means that $f_i = r$ (this graphical representation alone actually determines the matrix representation up to the ordering of the states q_1, q_2, q_3). The WA W gives rise to the LWA $L = L_W = (\mathbb{R}^Q, \{T_a\}, \phi)$. Another representation of the same LWA L, this time w.r.t. the basis $Q' = (q_1 + q_2, q_2 + q_3, q_1 + q_3)$, is given by the automaton W' in Fig. 2 on the right. That is, we have that

[1] As customary, we identify each $q \in Q$ with $\delta_q \in \mathbb{R}^Q$ defined as: $\delta_q(q') = 1$ if $q' = q$, $\delta_q(q') = 0$ otherwise. Under this identification, we have $Q \subseteq \mathbb{R}^Q$.

$L = L_W = L_{W'}$. Note that, in general, WA's derived from of the same LWA by fixing different bases have *similar* transition matrices [14]. The difference between them may only be relevant for computational purposes. As an example one will generally prefer to work with the representation on the left rather than with the one on the right above. More discussion on similarity can be found in [3].

3 Linear weighted bisimulation

We first show how to represent binary relations over V as sub-spaces of V, following [24].

Definition 3 (linear relation). *Let U be a sub-space of V. The binary relation R_U over V is defined by: $u\, R_U\, v$ if and only if $u - v \in U$. A relation R is* linear *if there is a subspace U such that $R = R_U$.*

Note that a linear relation is a total equivalence relation on V. Let now R be *any* binary relation over V. There is a canonical way of turning R into a linear relation, which we describe in the following. The *kernel* of R is defined by: $\ker(R) \triangleq \{u - v \mid u R v\}$. The *linear extension* of R, denoted R^ℓ, is defined by: $u R^\ell v$ if and only if $(u - v) \in \mathsf{span}(\ker(R))$. The following lemma summarizes two useful facts about linear relations.

Lemma 1. *(1) Let U be a sub-space of V, then $\ker(R_U) = U$. (2) Given any binary relation R, R^ℓ is the smallest linear relation containing R.*

According to the first part of the above lemma, a linear relation R is completely described by its kernel, which is a sub-space, that is

$$u R v \quad \text{ if and only if } \quad (u - v) \in \ker(R). \tag{1}$$

Conversely, to any sub-space $U \subseteq V$ there corresponds, by definition, a linear relation R_U whose kernel is U. Hence, without loss of generality, *we can identify linear relations on V with sub-spaces of V*. For example, by slight abuse of notation, we can write $u\, U\, v$ instead of $u\, R_U\, v$; and conversely, we will sometime denote by R the sub-space $\ker(R)$, for a linear relation R. The context will be sufficient to tell whether we are actually referring to a linear relation or to the corresponding sub-space (kernel). Note that the sub-space $\{0\}$ corresponds to the identity relation on V, that is $R_{\{0\}} = Id_V$. In fact: $u\, Id_V\, u$ iff $u = v$ iff $u - v = 0$. Similarly, the space V itself corresponds the universal relation on V. Another consequence of the preceding lemma, part (2), is that it is not restrictive to confine ourselves, as we do below, to relations over V that are linear. Note that, again by virtue of (2), $R^\ell = R$ if R is linear, hence $(\cdot)^\ell$ is idempotent: $(R^\ell)^\ell = R^\ell$.

We are now set to define linear weighted bisimulation. The definition relies on the familiar step-by-step game on transitions, plus an initial condition requiring that two related states have the same weight. We christen this form of bisimulation *linear* to stress the difference with other forms of bisimulation proposed for WA's [5]. In the rest of the section, we let L denote a generic LWA $(V, \{T_a\}_{a \in A}, \phi)$.

Definition 4 (linear weighted bisimulation). *Let L be a* LWA*. A linear relation R over V is a* linear weighted L-bisimulation *(L-bisimulation, for short) if whenever uRv then: (a) $\phi(u) = \phi(v)$ and (b) $T_a u \; R \; T_a v$ for each $a \in A$.*

That a largest L-bisimulation, denoted $\sim_L$, exists, is quite obvious: in analogy with the ordinary case, one takes the *span* of the union of all L-bisimulations, and checks that it is in turn a L-bisimulation, the largest one. Note that the mentioned union is non-empty, as e.g. the identity relation is a L-bisimulation. We shall give two characterizations of $\sim_L$, one in terms of language equivalence (Theorem 1) and one in algorithmic terms (Theorem 2). A useful property of bisimulation on ordinary transition systems is that, to prove two states related, exhibiting a "small" relation containing the given pair is sufficient. This property is preserved in the present setting, despite the fact that Definition 4 mentions linear, hence total, relations on V.

Lemma 2. *Let L be a* LWA *and R be a binary relation over V satisfying clauses (a) and (b) of Definition 4. Then R^ℓ is the smallest weighted L-bisimulation containing R.*

The following lemma provides a somewhat handier characterization of linear weighted bisimulation. Let us say that a sub-space U is *T-invariant* if $T(U) \subseteq U$. Bisimulations are transition-invariant relations that refine the kernel of ϕ.

Lemma 3. *Let L be a* LWA *and R be linear relation over V. R is a L-bisimulation if and only if (a) $\ker(\phi) \supseteq R$, and (b) R is T_a-invariant for each $a \in A$.*

The largest L-bisimulation $\sim_L$ coincides with the weighted-language equivalence.

Theorem 1. *For any $u, v \in V$, we have that $u \sim_L v$ if and only if $\sigma_L(u) = \sigma_L(v)$.*

4 Partition refinement

Two well-known concepts from Linear Algebra, orthogonal complements and transpose morphisms, will be used to describe geometrically two basic operations of the algorithm: the complement of a relation and the operation of "reversing arrows" in an automaton, respectively.

Let U, W be sub-spaces of V. We recall that the orthogonal complement $U^\perp$ enjoys the following properties: (i) $U^\perp$ is a sub-space of V; (ii) $(\cdot)^\perp$ reverses inclusions, i.e. if $U \subseteq W$ then $W^\perp \subseteq U^\perp$; (iii) $(\cdot)^\perp$ is an involution, that is $(U^\perp)^\perp = U$. These three properties suggest that $U^\perp$ can be regarded as a *complement*, or negation, of U seen as a relation. Another useful property is: (iv) $\dim(U^\perp) + \dim(U) = \dim(V)$. Concerning transpose morphisms, we have the following definition. The need for ortho-normal bases is explained in the remark below.

Definition 5 (transpose morphism). *Let $T : V \to V$ be any endomorphism on V. Fix any ortho-normal basis of V and let M be the square matrix representing T in this basis. We let the* transpose of T*, written ${}^{\mathsf{t}}T$, be the endomorphism $V \to V$ represented by ${}^{\mathsf{t}}M$ in the given basis.*

Remark 1. It is easy to check that the definition of ${}^{\mathsf{t}}T$ does *not* depend on the choice of the ortho-normal basis: this is a consequence of fact that the change of basis matrix N between two ortho-normal bases is unitary ($N^{-1} = {}^{\mathsf{t}}N$). The transpose operator is of course an involution, in the sense that ${}^{\mathsf{t}}({}^{\mathsf{t}}T) = T$.

Transpose morphisms and orthogonal spaces are connected via the following property, which is crucial to the development of the partition refinement algorithm. It basically asserts that T-invariance of R corresponds to ${}^{\mathsf{t}}T$-invariance of the complementary relation $R^{\perp}$.

Lemma 4. *Let U be a sub-space of V and T be an endomorphism on V. If U is T-invariant then $U^{\perp}$ is ${}^{\mathsf{t}}T$-invariant.*

An informal preview of the algorithm is as follows. Rather than computing directly the sub-space representing $\sim_L$, the algorithm computes the sub-space representing the complementary relation. To this end, the algorithm starts from a relation R_0 that is the complement of the relation identifying vectors with equal weights, then incrementally computes the space of all states that are *backward* reachable from R_0. The largest bisimulation is obtained by taking the complement of this space. Geometrically, "going backward" means working with the transpose transition functions ${}^{\mathsf{t}}T_a$ rather than with T_a. Taking the complement of a relation actually means taking its orthogonal complement. Recall that $U + W \triangleq \mathsf{span}(U \cup W)$.

Theorem 2 (partition refinement). *Let L be a* LWA*. Consider the sequence $(R_i)_{i\geq 0}$ of sub-spaces of V inductively defined by: $R_0 = \ker(\phi)^{\perp}$ and $R_{i+1} = R_i + \sum_{a\in A} {}^{\mathsf{t}}T_a(R_i)$. Then there is $j \leq \dim(L)$ s.t. $R_{j+1} = R_j$. The largest L-bisimulation is $\sim_L = R_j^{\perp}$.*

Proof. Since $R_0 \subseteq R_1 \subseteq R_2 \subseteq \cdots \subseteq V$, the sequence of the dimensions of these spaces is non-decreasing. As a consequence, for some $j \leq \dim(V)$, we get $\dim(R_j) = \dim(R_{j+1})$. Since $R_j \subseteq R_{j+1}$, this implies $R_j = R_{j+1}$.

We next show that $R_j^{\perp}$ is a L-bisimulation. Indeed, by the properties of the orthogonal complement: (a) $\ker(\phi)^{\perp} \subseteq R_j$ implies $(\ker(\phi)^{\perp})^{\perp} = \ker(\phi) \supseteq R_j^{\perp}$. Moreover: (b) for any action a, ${}^{\mathsf{t}}T_a(R_j) \subseteq {}^{\mathsf{t}}T_a(R_j) + R_j \subseteq R_{j+1} = R_j$ implies, by Lemma 4, that ${}^{\mathsf{t}}({}^{\mathsf{t}}T_a(R_j^{\perp})) = T_a(R_j^{\perp}) \subseteq R_j^{\perp}$; by (a), (b) and Lemma 3, we conclude that $R_j^{\perp}$ is an L-bisimulation.

We finally show that any L-bisimulation S is included in $R_j^{\perp}$. We do so by proving that for each i, $S \subseteq R_i^{\perp}$, thus, in particular $S \subseteq R_j^{\perp}$. We proceed by induction on i. Again by Lemma 3, we know that $R_0^{\perp} = \ker(\phi) \supseteq S$. Assume now $S \subseteq R_i^{\perp}$, that is, $S^{\perp} \supseteq R_i$. For each action a, by Lemma 3 we have that $T_a(S) \subseteq S$, which implies ${}^{\mathsf{t}}T_a(S^{\perp}) \subseteq S^{\perp}$ by Lemma 4. Hence $S^{\perp} \supseteq {}^{\mathsf{t}}T_a(S^{\perp}) \supseteq {}^{\mathsf{t}}T_a(R_i)$, where

the last inclusion stems from $S^\perp \supseteq R_i$. Since this holds for each a, we have that $S^\perp \supseteq \sum_a {}^tT_a(R_i) + R_i = R_{i+1}$. Taking the orthogonal complement on both sides reverses the inclusion and yields the wanted result.

Remark 2. What is being "refined" in the algorithm above are not, of course, the sub-spaces R_i, but their orthogonal complements: $R_0^\perp \supseteq R_1^\perp \supseteq \cdots \supseteq R_j^\perp = {\sim_L}$. One could also devise a version of the above algorithm that starts from $\ker(\phi)$ and refines it working forward, operating with intersections of sub-spaces rather than with sums. This "forward" version appears to be less convenient computationally as $\ker(\phi)$ is a large sub-space: since $\phi : V \to \mathbb{R}$ with $\dim(\mathbb{R}) = 1$, by virtue of the fundamental identity relating the dimensions of the kernel and of the image of a morphism, we have that $\dim(\ker(\phi)) \geq \dim(V) - 1$.

By virtue of (1), checking $u \sim_L v$, for any pair of vectors u and v, is equivalent to checking $u - v \in \ker(\sim_L)$. This can be done by first computing a basis of $\sim_L$ and then checking for linear (in)dependence of $u - v$ from this basis. Alternatively, and more efficiently, one can check whether $u - v$ is in the orthogonal complement of R_j, by showing that $u - v$ is orthogonal to each element of a basis of R_j. Thus, our task reduces to computing one such basis. To do so, we fix any orthonormal basis B of V: all the computations are carried out representing coordinates in this basis. Let f and M_a ($a \in A$) be the row-vector and matrices, respectively, representing the weight and transition functions of the LWA in this basis. Then a sequence of basis $B_0, ..., B_j$ for the sub-spaces $R_0, ..., R_j$ can be iteratively computed starting with $B_0 = \{v_0\}$, where v_0 is the vector represented by f, which is a basis for $\ker(\phi)^\perp$. The resulting algorithm requires in the worst case a cubic number of floating point operations in the dimension of the automaton. The algorithm is illustrated below.

Example 2. Consider the LWA $L = (V, \{T_a\}, \phi)$, with $V = \mathbb{R}^Q$ and $Q = (q_1, q_2, q_3)$, given in Example 1. The WA describing L w.r.t. Q is the one depicted in Fig. 1, on the left. Q is an ortho-normal basis, so that it is easy to represent the transpose transitions ${}^tT_a$. According to the above outline of the algorithm, since $f = (1, 1, 1)$ represents ϕ in Q, we have that $R_0 = \ker(\phi)^\perp$ is spanned by $v_0 = q_1 + q_2 + q_3$. Next, we apply the algorithm to build the B_i's as described above. Manually, the computation of the vectors ${}^tT_a v$ can be carried out by looking at the transitions of the WA with arrows reversed. Since ${}^tT_a(q_1 + q_2 + q_3) = q_1 + q_2 - q_1 + q_3 = q_2 + q_3$ and ${}^tT_a(q_1 + q_2 + q_3) = q_2 + q_3$, we obtain $B_0 = \{q_1 + q_2 + q_3\}$, then $B_1 = \{q_1 + q_2 + q_3, q_2 + q_3\}$ and finally $B_2 = B_1$. Hence B_1 is a basis of $(\sim_L)^\perp$. As an example, let us check that $q_1 \sim_L q_1 + q_2 - q_3$. To this purpose, note that the difference vector $(q_1 + q_2 - q_3) - q_1 = q_2 - q_3$ is orthogonal to each elements of B_1, which is equivalent to $q_1 \sim_L q_1 + q_2 + q_3$.

5 Quotients

The purpose of the quotient operation is to obtain a reduced automaton that has the same semantics as the original one. Let us make the notions of reduction and minimality precise first.

Definition 6 (reduction, minimality). *Let L and L' be two* LWA*'s having V and V', respectively, as underlying state-spaces. Let $h : V \to V'$ be a morphism. We say (h, L') is a* reduction of L *if* $\dim(L') \leq \dim(L)$ *and for each $v \in V$, $\sigma_L(v) = \sigma_{L'}(hv)$. We say L is* minimal *if for every reduction (h, L') of L we have* $\dim(L) = \dim(L')$.

We now come to the actual construction of the minimal automaton. This is basically obtained by quotienting the original space by the largest bisimulation. In inner product spaces, there is a canonical way of representing quotients as orthogonal complements. Let U be any sub-space of V. Then V can be decomposed as the direct sum of two sub-spaces: $V = U \oplus U^\perp$ and $\dim(V) = \dim(U) + \dim(U^\perp)$. This means that any element $v \in V$ can be written in a *unique way* as a sum $v = u + w$ with $u \in U$ and $w \in U^\perp$. The orthogonal *projection* of V onto $U^\perp$ is the morphism $\pi : V \to U^\perp$ defined as $\pi(u + w) \triangleq w$. The following lemma says that taking the quotient of V by a linear relation (whose kernel is) U amounts to "collapsing" vectors of V along the U-direction. Or, in other words, to projecting vectors of V onto $U^\perp$, the sub-space orthogonal to U (this is a well known result in Linear Algebra).

Lemma 5. *Let U be a sub-space of V and $\pi : V \to U^\perp$ be the projection onto $U^\perp$. Then for each $u, v \in V$: (a) $u \, U \, v$ if and only if $\pi u = \pi v$; (b) $u \, U \, \pi u$.*

In view of the above lemma, we will sometimes denote the orthogonal complement of U w.r.t. V as "V/U". In what follows, L denotes a LWA $(V, \{T_a\}_{a \in A}, \phi)$. We shall make use of the morphisms $(\pi T)_a \triangleq \pi \circ T_a$, for $a \in A$.

Definition 7 (quotient automaton). *Let R be a L-bisimulation and let π be the projection function onto V/R. We let the* quotient automaton L/R *be $(V/R, \{T^{\mathsf{q}}_a\}_{a \in A}, \phi^{\mathsf{q}})$ where $T^{\mathsf{q}}_a = (\pi T_a)_{|V/R}$ and $\phi^{\mathsf{q}} = \phi_{|V/R}$.*

Theorem 3 (minimal automaton). *Let R be a L-bisimulation and π be the projection function from V onto V/R. Then $(\pi, L/R)$ is a reduction of L such that: (a)* $\dim(L/R) = \dim(L) - \dim(R)$*, and (b) for each $u, v \in V$, $u \sim_L v$ if and only if $\pi u \sim_{L/R} \pi v$. Moreover, if R is $\sim_L$, then L/R is minimal and the following* coinduction *principle holds: for each $u, v \in V$, $u \sim_L v$ if and only if $\pi u = \pi v$.*

It is well-known that, when B is an orthogonal basis, for each $v \in V$, the projection of v onto the space spanned by B, πv, can be written as

$$\pi v = \sum_{e \in B} \frac{\langle v, e \rangle}{\langle e, e \rangle} e. \tag{2}$$

One can give a (concrete) representation of the minimal LWA in terms of a WA, by first computing an orthogonal basis of the quotient space $V/\sim_L$ and then representing the transition T^{q}_a and and final weight ϕ^{q} functions in this basis using the above formula. This is illustrated in the example below.

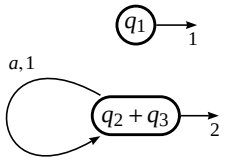

Fig. 2. A minimal weighted automaton.

Example 3. Let us consider again the LWA in Example 2. We give a representation of $L/\sim_L$ as a WA. From Example 2, we know that a basis of $V/\sim_L$ is $B = \{q_1 + q_2 + q_3, q_2 + q_3\}$. It is convenient to turn B into an orthogonal basis, applying an the Gram-Schmidt's [14] orthogonalizing method. We find that $B' = \{q_1, q_2 + q_3\}$ is an orthogonal basis of $V/\sim_L$. We now represent the transition function in B'. That is, for any $e \in B'$, we express each $T^{\mathrm{q}}_a e$ as a linear combination of elements of B'. Applying the identity (2), we find that

$$T^{\mathrm{q}}_a q_1 = \pi(q_2 - q_3) = 0$$
$$T^{\mathrm{q}}_a (q_2 + q_3) = \pi(q_2 + q_3) = q_2 + q_3 .$$

Concerning the weight function, we have: $\phi^{\mathrm{q}}(q_1) = 1$ and $\phi^{\mathrm{q}}(q_2 + q_3) = 2$. The resulting WA, which represents the LWA $L/\sim_L$ w.r.t. the basis B', is graphically represented in in Fig. 2. According to Theorem 3, the projection function π turns pairs of bisimilar elements of L into identical ones of $L/\sim_L$. As an example, the relation $q_1 \sim_L q_1 + q_2 - q_3$ becomes an identity once projected onto $V/\sim_L$: indeed, $\pi q_1 = q_1$ and $\pi(q_1 + q_2 - q_3) = \pi(q_1) + \pi(q_2 - q_3) = q_1 + 0 = q_1$.

6 Probabilistic bisimulation

The notion of probabilistic bisimulation was introduced by Larsen and Skou [15], as a generalization to probabilistic transition systems of the older notion of *lumpability* for Markov chains, due to Kemeny and Snell [13]. The notion of probabilistic transition system itself can be found in a number of variations in the literature; see e.g. [11,1] and references therein. Below, we deal with the the one called *reactive* probabilistic transition system. We comment on another version, the *generative* one, at the end of this section.

In this section, for any finite set Q, $f \in \mathbb{R}^Q$ and $X \subseteq Q$, we let $|f|_X \triangleq \sum_{q \in X} f(q)$. We abbreviate $|f|_Q$ just as $|f|$. A probabilistic transition system is just a weighted automaton with all final weights implicitly set to 1, and where the weights of arcs outgoing a node satisfy certain restrictions.

Definition 8 (probabilistic bisimulation). *A* (finite, reactive) probabilistic transition system *(*PTS*, for short) is a pair $P = (Q, \{t_a\}_{a \in A})$, where Q is finite set of states and $\{t_a\}_{a \in A}$ is a family of functions $Q \to (\mathbb{R}^+)^Q$, such that for each $a \in A$ and $q \in Q$, $|t_a(q)|$ equals 1 or 0. An equivalence relation $\mathcal{S}$ over Q is a* probabilistic bisimulation *on P if whenever $q \mathcal{S} q'$ then, for each equivalence class $C \in Q/\mathcal{S}$ and each $a \in A$, $|t_a(q)|_C = |t_a(q')|_C$.*

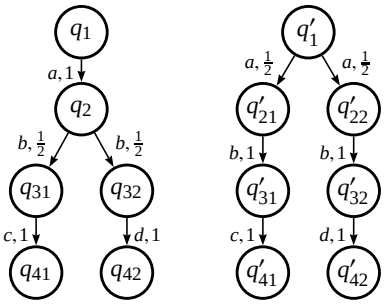

Fig. 3. A probabilistic transition system.

It is not difficult to see that a largest probabilistic bisimulation on P, denoted $\sim_P$, exists. Let $P = (Q, \{t_a\}_{a\in A})$ be a PTS. Any transition function t_a, being defined over Q, which is a basis of $\mathbb{R}^Q$ seen as a free vector space, is extended linearly to an endomorphism on the whole $\mathbb{R}^Q$: we denote this extension by the same name, t_a. With this notational convention, every PTS P determines a LWA $\hat{P} = (\mathbb{R}^Q, \{t_a\}_{a\in A}, \phi)$, where ϕ takes on the value 1 on each element of Q and is extended linearly to the whole space. Note that the semantics of a PTS is independent of final weights on states; we achieve the same effect here by setting $\phi(q) = 1$, the neutral element of product.

We establish below that, over Q, the largest linear weighted bisimulation, $\sim_{\hat{P}}$, is coarser than the largest probabilistic bisimulation, $\sim_P$. A similar result was already proven by Stark in [24], building on an alternative characterization of probabilistic bisimulation due to Jonsson and Larsen [12]. Our proof is more direct and relies on the following lemma.

Lemma 6. *Let $\mathcal{S}$ be a probabilistic bisimulation on the* PTS $(Q, \{t_a\}_{a\in A})$. *Let $f, g \in \mathbb{R}^Q$ s.t. for each $C \in Q/\mathcal{S}$, $|f|_C = |g|_C$. Then for each $C \in Q/\mathcal{S}$ and $a \in A$, $|t_a f|_C = |t_a g|_C$.*

Theorem 4. *Let $P = (Q, \{t_a\}_{a\in A}, \phi)$ be a* PTS. *If $q \sim_P q'$ in P then $q \sim_{\hat{P}} q'$ in $\hat{P}$.*

Proof. (Sketch) It is shown that that the linear relation $\sim_P^\ell$ over $\mathbb{R}^Q$ defined by: $f \sim_P^\ell g$ if and only if (i) $|f| = |g|$, and (ii) for all $a \in A$ and all equivalence classes $C \in Q/\sim_P$, $|t_a f|_C = |t_a g|_C$, is a linear weighted bisimulation. Lemma 6 is used to check requirement (b) of the definition.

To show that $\sim_P$ makes *less* identifications than $\sim_{\hat{P}}$, we consider the following example.

Example 4. The WA P in Fig. 3, with final weights all implicitly set to 1, is a PTS. Let $L = L_P$ be the corresponding LWA. It is easy to check that $q_1 \sim_L q'_1$. Indeed, consider the "small" relation R, defined thus

$$R = \{(q_1, q'_1), (q_2, \frac{1}{2}(q'_{21} + q'_{22})), (\frac{1}{2}(q_{31} + q_{32}), \frac{1}{2}(q'_{31} + q'_{32})), (\frac{1}{2}(q_{41} + q_{42}), \frac{1}{2}(q'_{41} + q'_{42})), (0, 0)\}.$$

One checks that this relation satisfies the clauses of bisimulation. Applying Lemma 2, one thus finds that R^ℓ is a linear weighted bisimulation, hence $q_1 \sim_L q'_1$.

In an even more direct fashion, one just checks that $\sigma_L(q_1) = \sigma_L(q'_1)$ and applies Theorem 1. On the other hand, q_1 and q'_1 are not related by any probabilistic bisimulation. In fact, any such relation should group e.g. q_{31} and q_{32} in the same equivalence class: but this is impossible, because q_{31} has a c-transition, whereas q_{32} has not.

The above example highlights the fundamental difference between probabilistic and linear weighted bisimulations. After each step, linear weighted bisimulation may relate "point" states to linear combinations of states: e.g., starting from q_1 and q'_1 and taking an a-step, q_2 and $\frac{1}{2}(q'_{21} + q'_{22})$ are related. This is not possible, by definition, in probabilistic bisimulation. A practical consequence of these results is that quotienting by the largest linear weighted bisimulation yields a minimal automaton that may be smaller than the one obtained when quotienting by probabilistic bisimilarity.

As hinted at the beginning of this section, a different version of probabilistic transition systems exists, the *generative* one. In this version, the requirement "for each $a \in A$ and $q \in Q$, $|t_a(q)|$ equals 1 or 0" is replaced by "for each $q \in Q$, $\sum_{a \in A} |t_a(q)|$ equals 1 or 0"[2]. The results discussed in this section carry over to this class of transition systems.

7 Weighted automata with an initial state

WA's are sometimes presented as featuring an initial distribution on states, see e.g. Buchholz's [5]. WA's with an initial distribution are also known as *linear representations* in Automata Theory [2]. In terms of LWA's, assuming an initial distribution on states is equivalent to choosing a distinguished initial vector, or root, so that we can define a *rooted* LWA as a pair (v, L), where $v \in V$. When minimizing, now one has now to take care of preserving only the semantics of the root. In particular, states that are not reachable from the root can be discarded right away, thus allowing for a potentially smaller reduced automaton.

We give here only a brief outline the minimization procedure, which is explained in detail in the full version [3]. The algorithm now consists of two steps: first, the sub-space reachable from the root is computed; second, the sub-automaton obtained by restricting the original one to this sub-space is minimized, according to the method described in Section 5. The second step is clearly a quotient operation involving bisimulation. But in fact, also the first step can be seen as a quotient operation. Indeed, if one looks at the minimization algorithm described in Theorem 2, one sees that computing the sub-space reachable from the root v, and spanned by the vectors $\{T_x v | x \in A^*\}$, is equivalent to computing the largest bisimulation of a LWA with transitions reversed and with a final weight function ψ such that $\mathrm{span}(v) = \ker(\psi)^\perp$. Let us denote by $\sim_L^{\mathrm{op}}$ the largest bisimulation in this reverse LWA. More formally, we have the following

[2] Modulo the addition of self-loops to sink states, generative and reactive transition systems correspond to Markov Chains and to Markov Decision Processes, respectively.

Theorem 5 (minimal rooted LWA). *Let (v, L) be a rooted LWA with $L = (V, \{T_a\}_a, \phi)$. Let π be the projection function $V \to V/\sim_L$. Consider the rooted LWA (v_*, L_*), where $v_* = \pi v$ and $L_* = (V_*, \{T_{*a}\}_a, \phi_*)$ is given by $V_* = \pi(V/\sim_L^{\mathrm{op}})$, $T_{*a} = (\pi T_a)_{|V_*}$ and $\phi_* = \phi_{|V_*}$. Then (v_*, L_*) is a minimal reduct of (v, L).*

This construction can be seen essentially as the original two-phase minimization algorithm given by Schützenberger [22], modulo the fact that here the two phases (computing the reachable sub-space and then reducing it) are both described in terms of bisimulation quotients.

When we apply this procedure to the rooted LWA (q_1, L), where L is the LWA of Example 1, we get a minimal rooted LWA that can be represented by the following WA:

Finally, one can show that any two minimal LWA's representing the same FPS are isomorphic, in particular they have the same dimension. This shows that the (minimal) dimension is a feature of FPS's rather than of rooted LWA's. We omit the details here.

8 Related and further work

Our formulation of linear weighted bisimulation is primarily related to the definition of *Σ-congruence* put forward by Stark [24]. Σ-congruence is introduced in order to provide a simple formulation of behavioural equivalence in a model of stochastic systems, *Probabilistic Input/Output Automata*, and relate this notion to standard probabilistic bisimulation. In the form studied by Stark, weighted automata do not feature final (nor initial) weights. This form is subsumed by ours once we assign the final weight 1 to all elements of the basis. In this special case, Σ-congruence and linear weighted bisimulation coincide. The representation of linear relations in terms of their kernels is already present in [24]. Partition refinement and quotient/minimization are not tackled, though. A related equivalence for stochastic systems, under the name behaviour equivalence, is studied in [26,25] (while weighted equivalence indicates there yet another equivalence).

Buchholz and Kemper have put forward a definition of bisimulation for weighted automata over a generic semiring [5,6]. A largest such bisimulation can be computed by a partition refinement algorithm that works on a matrix representation of the automata [5]; both a forward and a backward version of the equivalence and of the algorithm are investigated. A definition of "aggregated" automaton, corresponding to a quotient, is presented, but a notion of canonical representation is not put forward. Akin to the probabilistic one of Larsen and Skou, and differently from ours and Stark's, Buchholz and Kemper's bisimulations never relate a "point" state to a linear combinations of states. As a consequence, when instantiating the semiring in their framework to $\mathbb{R}$, their notion of equivalence is stricter than ours – and than weighted language equivalence – for the same reasons discussed in Example 4.

Weighted automata and formal power series play a central role in a few recent and less recent papers of Rutten [20,19,21] on coinduction and (multivariate) streams – another name for FPS's. In [20], weighted automata are used to

provide a more compact representation for streams than deterministic (Moore) automata do. Closely related to ours is also [21], where linear representations very similar to our LWA's are considered. Bisimulation is defined over streams – seen as deterministic Moore automata – and two states of a weighted automaton are related iff they generate the same stream. This approach is also taken in [19], where it is shown that infinite weighted automata can be used to enumerate a wide class of combinatorial objects. The stream-based definition can be used to prove an infinite automaton equivalent to a "small" one. The latter can be directly mapped to a closed expressions for the generating function of the enumerated objects.

Weighted automata were first introduced in Schützenberger's classical paper [22], where a minimization algorithm was also discussed. This algorithm has been reformulated in a more algebraic fashion in Berstel and Reutenauer's book [2]. Other descriptions of the algorithm can be found in [7,10]. Here we explicitly connect this algorithm to the notion of bisimulation. Hopefully, this connection will make the algorithm itself accessible to a larger audience.

Due to lack of space, we have not presented results concerning composition of automata. Indeed, it is quite easy to prove that both direct sum (juxtaposition) and tensor product (synchronization) of LWA's preserve bisimulation equivalence (see also Stark's [24, Section 3]). Also, the results presented here can be extended to the case of vector spaces over a generic field, relying on the concept of dual space (see [3]).

There are several possible directions for future work. One would like to extend the present approach to the case of infinite WA's. This would provide proof techniques, if not effective algorithms, that could be used to reason in a more systematic manner on the counting automata of [19]. Also, it would be interesting to cast the present results in a more explicit co-algebraic setting: this would put them in a deeper perspective and possibly help to explain certain aspects not clear at moment, such as, why the blow up of the ordinary case goes away. It would also be practically relevant to identify classes of properties preserved by linear weighted bisimilarity on probabilistic systems: a preliminary investigation shows that reachability is one such class. The relationship of linear weighted bisimilarity with other notions of equivalences/preorders [23,8] that also relate distributions, rather than individual states, deserves further attention.

References

1. C. Baier, B. Engelen, M. E. Majster-Cederbaum. Deciding Bisimilarity and Similarity for Probabilistic Processes. *Journal of Computer and System Sciences*, 60(1): 187-231, 2000.
2. J. Berstel, C. Reutenauer. *Rational Series and Their Languages.* EATCS Monograph Series, Springer-Verlag, 1988. New edition, *Noncommutative Rational Series With Applications*, 2008, available from `http://www-igm.univ-mlv.fr/~berstel/LivreSeries/LivreSeries.html`.
3. M. Boreale. Weighted bisimulations in linear algebraic form. Full version of the present paper, 2009. Available from `http://rap.dsi.unifi.it/~boreale/papers/WBG.pdf`.

4. P. Buchholz. Exact Performance Equivalence: An Equivalence Relation for Stochastic Automata. *Theoretical Computer Science*, 215(1-2): 263-287, 1999.
5. P. Buchholz. Bisimulation relations for weighted automata. *Theoretical Computer Science* 393(1-3): 109-123, 2008.
6. P. Buchholz, P. Kemper. Quantifying the Dynamic Behavior of Process Algebras. *PAPM-PROBMIV* 2001: 184-199, 2001.
7. A. Cardon and M. Crochemore. Determination de la representation standard d'une serie reconnaissable. *RAIRO Theor. Informatics and Appl.* 14: 371–379, 1980.
8. Y. Deng, R.J. van Glabbeek, M. Hennessy and C.C. Morgan. Characterising testing preorders for finite probabilistic processes. *Logical Methods in Computer Science* 4(4:4), 2008.
9. R. De Nicola, Matthew Hennessy. Testing Equivalences for Processes. *Theoretical Compututer Science* 34: 83-133, 1984.
10. M. Flouret and E. Laugerotte. Noncommutative minimization algorithms. *Inform. Process. Lett.* 64: 123–126, 1997.
11. R. J. van Glabbeek, S. A. Smolka, B. Steffen, C. M. N. Tofts. Reactive, Generative, and Stratified Models of Probabilistic Processes, *LICS 1990*, 130-141, 1990.
12. B. Jonsson, K. G. Larsen. Specification and Refinement of Probabilistic Processes, *LICS 1991*: 266-277, 1991.
13. J.G. Kemeny, J.L. Snell. *Finite Markov Chains*. Springer-Verlag, 1976.
14. S.A. Lang. *Introduction to Linear Algebra*, 2/e. Springer-Verlag, 1997.
15. K.G. Larsen, A. Skou. Bisimulation through Probabilistic Testing. *Information and Compututation*, 94(1): 1-28, 1991.
16. A. R. Meyer, L. J. Stockmeyer. Word problems requiring exponential time. In *STOC 1973*: 1-9, 1973.
17. R. Milner. *A Calculus of Communicating Systems*. Prentice-Hall, 1989.
18. D. Park. Concurrency and Automata on Infinite Sequences. *Theoretical Computer Science* 1981: 167-183.
19. J.J.M.M. Rutten. Coinductive counting with weighted automata. *Journal of Automata, Languages and Combinatorics* 8(2): 319-352, 2003.
20. J.J.M.M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theoretical Computer Science*, 308(1–3): 1–53, 2003.
21. J.J.M.M. Rutten. Rational streams coalgebraically. *Logical Methods in Computer Science*, 4(3), 2008.
22. M. P. Schützenberger. On the Definition of a Family of Automata. *Information and Control*, 4(2-3): 245-270, 1961.
23. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT. 1995.
24. E.W. Stark. On Behavior Equivalence for Probabilistic I/O Automata and its Relationship to Probabilistic Bisimulation. *Journal of Automata, Languages and Combinatorics* 8(2): 361-395, 2003.
25. E. W. Stark, R. Cleaveland, S.A. Smolka: Probabilistic I/O Automata: Theories of Two Equivalences. *CONCUR 2006*, LNCS 4137:343-357, 2006.
26. S.-H. Wu, S. A. Smolka, and E. W. Stark. Composition and behaviors of probabilistic I/O automata. *Theoretical Computer Science*, 176(1-2):1-38, 1997.