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, over the state-space, the largest bisimulation equivalence coincides with weighted language equivalence and can be computed by a linear algebraic version of the partition refinement algorithm. Quotienting an automaton corresponds to taking the orthogonal projection of the original state-space onto a bisimulation. In the case of the largest bisimulation, this yields the minimal weighted-language-preserving automaton. We also clarify the relations of our notion of bisimulation to other definitions found in the literature, including probabilistic bisimulation, and to certain classical constructions in Automata Theory.

Keywords: bisimulation, weighted automata, minimization, linear representations, probabilistic bisimulation.

1 Introduction

Weighted automata [29, 4] unify various computational models of practical relevance, arising in such diverse areas of application as modelling of probabilistic and stochastic systems (e.g. [20, 5, 15]), language and speech processing (e.g. [23]), Enumerative Combinatorics [25], Control Theory (e.g. [16]). In this type of automaton, transitions are labelled with both a letter from a finite alphabet, or *action*, and a *weight* taken from a semiring. In this paper, we will be exclusively concerned with the case when the semiring is actually a field, and particularly with the field of real numbers. This weight may have different interpretations, such as probability, transition speed, cost or multiplicity. In particular, Markov chains [18], probabilistic transition systems [20] and stochastic automata [5, 15] can all be seen as instances of this class of automata. Bisimulation, introduced in Concurrency theory by Park and Milner [24, 22], is a well-understood concept for ordinary automata and transition systems. It serves a conceptual and a pragmatical purpose, both of paramount importance. Conceptually, it provides a sensible notion of equivalence that can be used to reason about systems. Pragmatically, it allows one to reduce the number of state of a system by aggregation of bisimilar states, being assured that the reduction preserves certain

*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.

properties. Moreover, bisimulation comes equipped with nice coinductive proof techniques and partition refinement algorithms.

We study bisimulation , 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. [12, 20, 5, 1, 6] 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 [10], 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 [21]. 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* [4]. Two states q and q' are weighted language 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 [10], 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 [25], 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 [29]. 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. If the automaton has a final weight function, this too can be seen as a linear map from the state-space to the reals. Compared to the usual representation [4, 6], LWA's identify automata up to similarity of their transition matrices, thus providing one with a basic level of abstraction, somewhat analogous to that provided by structural congruence in process calculi. 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 [29]. 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 [20] 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 [6, 7], by Rutten [26, 25, 27] and by Stark [31], who have studied some of the issues we consider here.A detailed comparison with their work and proposals, as well as with the classical constructions in Automata theory [4], is deferred to Section 9.

In summary, we make the following contributions:

- we give a simple linear algebraic formulation of linear weighted bisimulation; this enjoys the usual coinductive proof technique and is correct and complete for weighted language equivalence;
- we give a feasible partition refinement algorithm in linear algebraic form for computing the largest bisimulation;
- we give an account of minimization as orthogonal projection, this results into a feasible minimization algorithm and canonical forms;
- we clarify the relationship between linear weighted bisimulation and other equivalences, like probabilistic bisimulation, and classical minimization algorithms in Automata Theory.

Structure of the paper Linear weighted automata are introduced in Section 2 and linear weighted bisimulation in Section 3, where the coincidence with weighted-language equivalence is also discussed. The partition refinement algorithm is discussed in Section 4. Quotients are the subject of Section 5. In Section 6 we study the relationship between weighted and probabilistic bisimulation. For technical convenience, LWA's are presented in the preceding sections as not featuring an initial distribution/weighting; when this is considered, we obtain *rooted* LWA's, which are the subject of Section 7. An extension to the case of weights in a generic field is outlined in Section 8.A detailed discussion of related and further work is found in Section 9. To enhance readability, a few technical definitions and proofs have been confined to a few separate appendices (A, B and C).

Background and notation We will make use of a few basic concepts from Linear Algebra, such as vector space and sub-space, span of a set of vectors, basis, dimension, vector space homomorphism, kernel of a homomorphism, matrix representation, orthogonality, orthogonal complement, inner product, inner product space, free vector space. These concepts are described in any introductory textbook in Linear Algebra, e.g. [19]. For the reader's convenience, a brief summary of the most relevant definitions and results is also provided in Appendix A. We will often refer to elements of a vector space as "vectors"; the null vector will be always denoted by $\mathbf{0}$, 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".

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).

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 $\sigma : A^* \to \mathbb{R}$. The set of all FPS's over A and $\mathbb{R}$ is denoted by $\mathbb{R}\langle\langle A \rangle\rangle$.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 $\sigma_L(u) = \sigma_L(v)$.*

In passing, we note that the preceding two definitions do not depend on the fact that V is finite-dimensional, so that they can be extended without modifications to the infinite-dimensional case (we will use this fact at some point in Section 7). 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[1]; 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. [6]). 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

[1]This component can be dispensed with, if it is stipulated that the set of states is always an integer interval $1..n$.

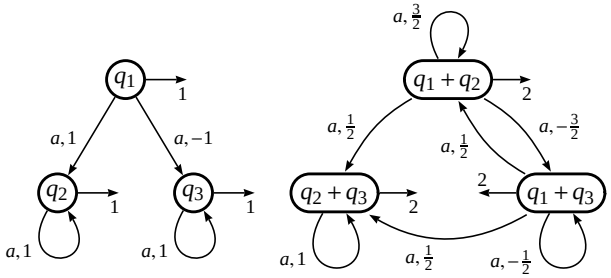

Figure 1: Two weighted automata representing the same linear weighted automaton.

by W is given by: $\sigma_W(s)(x) \stackrel{\triangle}{=} fM_x s$, 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 ($g \cdot h \stackrel{\triangle}{=} \sum_{q \in Q} g(q) \cdot h(q)$), 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[2] Q. Again, this correspondence preserves the weighted-language semantics. The two constructions are inverse of one another, e.g. one has $W_{L_W,Q} = W$. Note that two WA's derived from the same LWA w.r.t. different bases need not be have isomorphic transitions, although they have the same number of states. In particular, they have *similar* transition matrices. This is discussed in the next example and in a bit more detail in Appendix B.

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 $L = L_W = L_{W'}$. Hence, the two automata in the figure are similar, despite the fact that the one on the right has a more complicated transition structure. This difference may be practically important, though: for computational purposes, one will generally prefer the representation on the left to the one on the right.

3 Linear weighted bisimulation

We first show how to represent binary relations over V as sub-spaces of V, following [31].

[2] 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 $Q \subseteq \mathbb{R}^Q$.

Definition 3 (linear relation) *Let U be a sub-space of V. The binary relation R_U over V is defined by*

$$u\, R_U\, v \text{ if and only if } u - v \in U.$$

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 [6]. 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 $u\, R\, v$ 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 $\sigma_L(u) = \sigma_L(v)$.*

PROOF Suppose that $u \sim_L v$. It is easy to prove by induction on $|x|$ that $\sigma_L(u)(x) = \sigma_L(v)(x)$, for each $x \in A^*$. Conversely, consider the relation $R = \{(u,v) | \sigma_L(u) = \sigma_L(v)\}$. It is easy to see that this is a linear relation. Moreover, it is a L-bisimulation. Concerning clause (a) of Definition 4, note that $\phi(u) = \sigma_L(u)(\epsilon) = \sigma_L(v)(\epsilon) = \phi(v)$. Concerning (b), note that, for each a and $x \in A^*$: $\sigma_L(T_a u)(x) = \sigma_L(u)(ax) = \sigma_L(v)(ax) = \sigma_L(T_a v)(x)$, so that $\sigma_L(T_a u) = \sigma_L(T_a v)$, that is, $T_a u \, R \, T_a v$. □

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 ${}^{\mathrm{t}}T$, be the endomorphism $V \to V$ represented by ${}^{\mathrm{t}}M$ in the given basis.*

Remark 1 It is easy to check that the definition of ${}^{\mathrm{t}}T$ 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 ($N^{-1} = {}^{\mathrm{t}}N$). The transpose operator is of course an involution, in the sense that ${}^{\mathrm{t}}({}^{\mathrm{t}}T) = T$.

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 ${}^{\mathrm{t}}T$-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 ${}^{\mathrm{t}}T$-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 ${}^{\mathrm{t}}T_a$ rather than with T_a. Taking the complement of a relation actually means taking its orthogonal complement.

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:*

$$R_0 \quad = \quad \ker(\phi)^{\perp} \qquad R_{i+1} \quad = \quad R_i + \textstyle\sum_{a\in A} {}^{\mathrm{t}}T_a(R_i)\,.$$

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, ${}^{\mathrm{t}}T_a(R_j) \subseteq {}^{\mathrm{t}}T_a(R_j) + R_j \subseteq R_{j+1} = R_j$ implies, by Lemma 4, that ${}^{\mathrm{t}}({}^{\mathrm{t}}T_a(R_j^{\perp})) = T_a(R_j^{\perp}) \subseteq R_j^{\perp}$; 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 ${}^{\mathrm{t}}T_a(S^{\perp}) \subseteq S^{\perp}$ by Lemma 4. Hence $S^{\perp} \supseteq {}^{\mathrm{t}}T_a(S^{\perp}) \supseteq {}^{\mathrm{t}}T_a(R_i)$, where the last inclusion stems from $S^{\perp} \supseteq R_i$. Since this holds for each a, we have that $S^{\perp} \supseteq \sum_a {}^{\mathrm{t}}T_a(R_i) + R_i = R_{i+1}$. 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 for at least two reasons. First, $\ker(\phi)$ 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(see equality (4) in Appendix A), we have that $\dim(\ker(\phi)) \geq \dim(V) - 1$. Second, the backward algorithm returns at no additional cost the orthogonal complement of $\sim_L$, which is anyway necessary to build the minimal automaton (see next section), this is not the case for the forward version.

To end the section, we briefly discuss some practical aspects involved in the implementation of the algorithm. 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 independence of $u - v$. 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, fix any orthonormal basis B of V and let f and M_a $(a \in A)$ be the row-vector and matrices representing the weight and transition

functions of the LWA in this basis. All the computations are carried out representing coordinates in this basis.

1. Compute a basis B_0 of R_0. As already discussed, $\dim(\ker(\phi)) \geq \dim(V) - 1$, hence $\dim(\ker(\phi)^\perp) \leq 1$. It is readily checked that $\ker(\phi)^\perp$ is spanned by the vector v_0 represented in B by ${}^t f$, thus we set $B_0 = \{v_0\}$.

2. For each $i \geq 0$, compute a basis B_{i+1} of R_{i+1}. This can be obtained by incrementally joining to B_i the vectors ${}^t T_a v$, for $a \in A$ and $v \in B_i$ that are linearly independent from previously joined vectors. The actual computations of ${}^t T_a v$ are carried out using the transposed matrices ${}^t M_a$.

After $j \leq n$ iterations, one finds a set B_j s.t. $B_{j+1} = B_j$: this is the basis of R_j, the orthogonal complement of $\sim_L$. We illustrate this algorithm in the example 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 ${}^t T_a$. 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 ${}^t T_a v$ can be carried out by looking at the transitions of the WA with arrows reversed. Since ${}^t T_a(q_1 + q_2 + q_3) = q_1 + q_2 - q_1 + q_3 = q_2 + q_3$ and ${}^t T_a(q_1 + q_2 + q_3) = q_2 + q_3$, 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 $q_1 \sim_L q_1 + q_2 + q_3$.

Remark 3 The space R_j corresponds to what is known in numerical Linear Algebra as *Krylov space*. There exist numerically stable methods to build a basis of such spaces in $O(n^3)$ floating-point operations, such as the Arnoldi iteration, see [13, Ch. 9]. The Arnoldi iteration returns at no additional cost the projection of the transition matrix on the space R_j, thus automatically providing the minimized automaton (see next section).

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 $\pi(u+w) \stackrel{\triangle}{=} w$. 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 $(\pi T)_a \stackrel{\triangle}{=} \pi \circ T_a$, 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 $\phi^{\mathsf{q}} = \phi_{|V/R}$.*

The following lemma states a simple property of invariant relations and projections.

Lemma 6 *Let R be a T-invariant subspace and π be the projection function onto V/R. Then $\pi \circ T \circ \pi = \pi \circ T$.*

PROOF Let $v \in V$. Then $v - \pi v \in R$ (Lemma 5(b)) implies $T(v-\pi v) \in R$ (R-invariance) implies $\pi T(v-\pi v) = 0$ (by definition of projection) that is $\pi T v = \pi T \pi v$. □

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 $\pi u = \pi v$.*

PROOF First observe that for each $x \in A^*$, R is T_x-invariant. Exploiting this fact and relying on Lemma 6, it is easy to check that $\sigma_L(v)(x) = \sigma_{L/R}(\pi v)(x)$ for each x and $v \in V$. Moreover, by definition of orthogonal complement, we have that $\dim(V/R) = \dim(R^\perp) = \dim(V) - \dim(R) \leq \dim(V)$. These facts say that $(\pi, L/R)$ is a reduction of L and its dimension is as stated in part (a). Part (b) is a consequence of $(\pi, L/R)$ being a reduction of L and of the characterization of the largest bisimulation in terms of weighted language equivalence (Theorem 1).

Suppose now that $R = \sim_L$, the largest L-bisimulation. Let us first consider the coinduction principle: this is in fact an instance of Lemma 5(a). Also note that, combining coinduction and part (b) above, we find that the largest $(L/\sim_L)$-bisimulation, that is $\sim_{L/\sim_L}$, is $Id_{V/\sim_L}$, the identity relation over $V/\sim_L$. Let us now prove minimality of $L/\sim_L$. Let (h,L') be any reduction of $L/\sim_L$: we show that $\dim(h(V/\sim_L)) = \dim(V/\sim_L)$, which implies that $L/\sim_L$ is minimal. To this purpose, consider the equivalence relation on $V/\sim_L$ induced by $\ker(h)$. That is, more explicitly, the relation on $V/\sim_L$ defined thus $u\ R_{\ker(h)}\ v$ if and only if $u - v \in \ker(h)$. Now, $u\ R_{\ker(h)}\ v$ implies, as a consequence of (h,L') being a reduction, that $\sigma_{L/\sim_L}(u) = \sigma_{L/\sim_L}(v)$. This is in turn equivalent to $u \sim_{L/\sim_L} v$ (Theorem 1), that is, as noted above, $u = v$. In other words, it holds that $R_{\ker(h)} \subseteq Id_{V/\sim_L}$. This is equivalent to $\ker(h) \subseteq \ker(Id_{V/\sim_L}) = \{0\}$, hence $\ker(h) = \{0\}$. Now, applying the fundamental identity (4), we find $\dim(h(V/\sim_L)) = \dim(V/\sim_L) - \dim(\ker(h)) = \dim(V/\sim_L)$. □

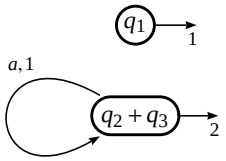

Figure 2: A minimal weighted automaton.

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.

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 [19] 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

$$\begin{array}{rcccl} T^{\mathrm{q}}_a q_1 & = & \pi(q_2 - q_3) & = & 0 \\ T^{\mathrm{q}}_a (q_2 + q_3) & = & \pi(q_2 + q_3) & = & q_2 + q_3 \,. \end{array}$$

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 [20], as a generalization to probabilistic transition systems of the older notion of *lumpability* for Markov chains, due to Kemeny and Snell [18]. The notion of probabilistic transition system itself can be found in a number of variations in the literature; see e.g. [12, 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 $|f|_X \stackrel{\triangle}{=} \sum_{q \in X} f(q)$. We abbreviate $|f|_Q$ just as $|f|$. We record for future use that for any $X \subseteq Q$, the function $|\cdot|_X$ is a linear morphism $\mathbb{R}^Q \to \mathbb{R}$. We let $\mathbb{R}^+$ be the set of non-negative reals.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 $a \in A$, $|t_a(q)|_C = |t_a(q')|_C$.*

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(see Appendix A), 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.

As discussed in the Introduction, linear weighted bisimulation for probabilistic PTS has a very natural interpretation. Indeed, the weight $\sigma_L(q)(x)$ associated to each state q and string x is obtained as the sum of the probabilities of each path labelled x starting from q. In other words, $\sigma_L(q)(x)$ can be interpreted as the probability of observing the string x in an execution starting from q.

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 [31], building on an alternative characterization of probabilistic bisimulation due to Jonsson and Larsen [17]. Our proof is more direct and relies on the following lemma.

Lemma 7 *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 $a \in A$, $|t_a f|_C = |t_a g|_C$.*

PROOF For any $h \in \mathbb{R}^Q$, say $h = \sum_{q\in Q} r_q q$, and any $X \subseteq Q$, let us write h_X for $\sum_{q\in X} r_q q$. Let $C_1, ..., C_n$ be the equivalence classes of $Q/\mathcal{S}$. Now, take f and g as in the statement of the lemma. Clearly, $f = f_{C_1} + \cdots + f_{C_n}$ and $g = g_{C_1} + \cdots + g_{C_n}$. Hence, by linearity of t_a

$$\begin{aligned} t_a f &= t_a(f_{C_1}) + \cdots + t_a(f_{C_n}) \\ t_a g &= t_a(g_{C_1}) + \cdots + t_a(g_{C_n}). \end{aligned}$$

From these equalities and linearity of $|\cdot|_C$, it follows that, for each C, $|t_a f|_C = \sum_i |t_a(f_{C_i})|_C$, and similarly $|t_a g|_C = \sum_i |t_a(g_{C_i})|_C$. Therefore, to prove that $|t_a f|_C = |t_a g|_C$, it suffices to show that for each pair of classes C_i and C, $|t_a(f_{C_i})|_C = |t_a(g_{C_i})|_C$. To this purpose, write f_{C_i} as $\sum_{q\in C_i} \lambda_q q$ and g_{C_i} as $\sum_{q\in C_i} \lambda'_q q$. By linearity of t_a and $|\cdot|_X$, we have that $|t_a f_{C_i}|_C = \sum_{q\in C_i} \lambda_q |t_a q|_C = (\sum_{q\in C_i} \lambda_q) k = |f|_{C_i} k$, where k is a constant s.t. $|t_a q|_C = k$ for each $q \in C_i$: that all these quantities $|t_a q|_C$ are the same stems from the fact that C_i is an equivalence class of the probabilistic bisimulation $\mathcal{S}$. Similarly, $|g_{C_i}|_C = |g|_{C_i} k$. Since $|f|_{C_i} = |g|_{C_i}$ by assumption, the thesis follows. □

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, see Appendix C for a detailed proof) 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

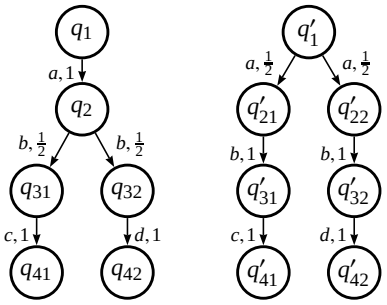

Figure 3: A probabilistic transition system.

$C \in Q/\sim_P$, $|t_a f|_C = |t_a g|_C$, is a linear weighted bisimulation. Lemma 7 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. The states of this minimal automaton will in general not be equivalence classes of states of the original automaton, but linear combinations of them. Stark, in [31], shows that probabilistic bisimilarity coincides with the the largest linear weighted bisimulation that can be obtained as a linear extension of some equivalence relation on Q.

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"[3]. The results discussed in this section carry over to this class of transition systems.

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

7 Weighted automata with an initial state

WA's are sometimes presented as featuring an initial distribution on states, see e.g. Buchholz's [6]. WA's with an initial distribution are also known as *linear representations* in Automata Theory [4]. In terms of LWA's, assuming an initial distribution on states is equivalent to choosing a distinguished initial vector, the *root*.

Definition 9 (rooted LWA) *A* rooted LWA *is a pair* (v, L)*, where* $v \in V$*. The power series* represented by (v, L) *is* $\sigma_L(v)$*. The dimension of* (v, L) *is given by* $\dim(L)$*.*

Minimization has now to take care of preserving only the semantics of the root. This fact implies that states that are not reachable from the root can be discarded right away, thus allowing for a potentially smaller reduced automaton.

Definition 10 (reduction, minimality) *Let* (v, L)*,* (u, L') *be rooted* LWA*'s representing the same power series. We say* (u, L') *is a* reduct *of* (v, L) *if* $\dim(L') \leq \dim(L)$*. We say* (v, L) *is* minimal *if it does not admit reducts of lesser dimension.*

The minimization 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, spanned by the vectors $\{T_x v | x \in A^*\}$, is equivalent to computing the largest bisimulation of a LWA with transitions reversed, and having v as a basis of $\ker(\psi)^\perp$, where ψ is its final weight function. This construction can be seen essentially as the original two-phase minimization algorithm given by Schützenberger [29], 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.

Below, we describe briefly this construction. We first make the symmetry between the first and second step of the algorithm precise. First, given any (v, L) we show how to describe a "reverse" rooted automaton $(v, L)^{\mathrm{op}}$ with final and initial distributions on states swapped, and transitions reversed. Recall from Linear Algebra that, for any morphism $\phi : V \to \mathbb{R}$, there is a unique $u \in V$ that represents[4] ϕ w.r.t. to inner product $\langle \cdot, \cdot \rangle$, in the sense that $\phi = w \mapsto \langle u, w \rangle$.

Definition 11 (duality) *Let* (v, L) *be a rooted* LWA*. The* dual of (v, L)*, written* $(v, L)^{\mathrm{op}}$*, is the rooted* LWA $(u, (V, \{{}^{\mathrm{t}}T_a\}_{a \in A}, \psi))$*, where* $u \in V$ *represents* ϕ *and* $\psi(w) \triangleq \langle v, w \rangle$*, for any* $w \in V$*.*

Note that, if ψ is the final weight function of $(v, L)^{\mathrm{op}}$, then $\ker(\psi)^\perp = \mathrm{span}\{v\}$. Also note that $(\cdot)^{\mathrm{op}}$ is an involution, that is $((v, L)^{\mathrm{op}})^{\mathrm{op}} = (v, L)$. In what follows, for $(v, L)^{\mathrm{op}} = (u, L')$, we shall denote by $\sim_L^{\mathrm{op}}$ the largest L'-bisimulation.

Lemma 8 (v, L) *is minimal if and only if* $(v, L)^{\mathrm{op}}$ *is minimal.*

Lemma 9 *A rooted* LWA (v, L) *is minimal if and only if the bisimulations* $\sim_L$ *and* $\sim_L^{\mathrm{op}}$ *both coincide with the identity equivalence relation on* V*.*

[4] It is easy to check that, taken any ortho-normal basis B of V, this u can be written as $\sum_{e \in B} \phi(e) e$.

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*

$$V_*=\pi(V/\sim_L^{\mathrm{op}}) \qquad T_{*a}=(\pi T_a)_{|V_*} \qquad \phi_*=\phi_{|V_*}.$$

Then (v_*,L_*) *is a minimal reduct of* (v,L).

PROOF We first check that L_* is well-defined and is actually a reduct. First, we check that for each a and $u\in\pi(V/\sim_L^{\mathrm{op}})$, one has $T_{*a}u\in\pi(V/\sim_L^{\mathrm{op}})$. Let $u=\pi w$ for some $w\in V/\sim_L^{\mathrm{op}}$, we have: $T_{*a}u=\pi T_a\pi w=\pi T_a w$ where the last equality stems from Lemma 6. Now $T_a w\in V/\sim_L^{\mathrm{op}}$, as $T_a(V/\sim_L^{\mathrm{op}})\subseteq V/\sim_L^{\mathrm{op}}$, by ${}^{\mathrm{t}}T_a(V)\subseteq V$ (Lemma 3(b)) and Lemma 4. Hence $\pi T_a w\in\pi(V/\sim_L^{\mathrm{op}})$. Again by applying Lemma 6, it is easily checked that, for each $x\in A^*$, $T_{*x}\pi v=\pi T_x v$, hence $\sigma_{L_*}(v_*)=\phi_*(T_{*x}\pi v)=\phi(\pi T_x v)=\phi(T_x v)=\sigma_L(v)$, where the last but one equality in this chain stems from $\pi w\sim_L w$ for any w. As clearly $\dim(V_*)\leq\dim(V)$, we have thus shown that (v_*,L_*) is a reduct of (v,L).

Concerning minimality, according to Lemma 9, it will suffice to check that both the largest bisimulation over L_*, say $\sim_*$, and the largest bisimulation over the dual LWA, say $\sim_*^{\mathrm{op}}$, are the identity relation over V_*. That $\sim_*$ is the identity over V_* follows from the fact that, by construction, L_* is a sub-automaton (in the obvious sense) of the quotient automaton $L/\sim_L$: hence $\ker(\sim_*)=\ker(\sim_{L/\sim_L})\cap V_*$, and the result follows because $\sim_{L/\sim_L}$ is the identity over $V/\sim_L$ (Corollary 10). Concerning $\sim_*^{\mathrm{op}}$, we show that $V_*/\sim_*^{\mathrm{op}}=V_*$, that is, (the kernel of) $\sim_*^{\mathrm{op}}$ has dimension 0, hence, in terms of relations, is Id_{V_*}. To see this, we rely on the characterization of bisimulations orthogonal complements given by Theorem 2. Indeed, for a finite, large enough $X\subseteq A^*$ (e.g. take $X=\cup_{0\leq i\leq n}A^i$), we can write

$$\begin{aligned} V/\sim_L^{\mathrm{op}} &= \textstyle\sum_{x\in X} T_x(\mathrm{span}\{v\}) \\ V_*/\sim_*^{\mathrm{op}} &= \textstyle\sum_{x\in X} T_{*x}(\mathrm{span}\{v_*\}. \end{aligned}$$

Now, relying on the fact that $\mathrm{span}(\cdot)$ commutes with linear maps and applying Lemma 6, each summand of the second sum above can be re-written as follows: $T_{*x}(\mathrm{span}\{v_*\})=\mathrm{span}(T_{*x}\pi v)=\mathrm{span}(\pi T_x v)=\pi T_x\mathrm{span}\{v\}$. Hence we can re-write the second equation above as follows, where the third equality stems from the above equation for $V/\sim_L^{\mathrm{op}}$

$$\begin{aligned} V_*/\sim_*^{\mathrm{op}} &= \textstyle\sum_{x\in X}\pi T_x\mathrm{span}\{v\} \\ &= \pi(\textstyle\sum_{x\in X} T_x\mathrm{span}\{v\}) \\ &= \pi(V/\sim_L^{\mathrm{op}}) \\ &= V_*. \end{aligned}$$

□

Example 5 Consider again the LWA L of Example 1 and the rooted LWA given by (q_1,L). We construct the corresponding minimal rooted LWA, (v_*,L_*), and then represent it as a WA. Applying the algorithm described in Section 3, we find a basis of the sub-space $V/\sim_L^{\mathrm{op}}$, which is $\{q_1,q_2-q_3\}$. Note that this amounts to starting from state q_1 in L and collecting all reachable linear combinations of states. In Example 2 we have already identified an (orthogonal) basis of $V/\sim_L$ and the corresponding projection function, π. A basis of V_* is computed by projecting the basis of $V/\sim_L^{\mathrm{op}}$ onto the basis of $V/\sim_L$, thus obtaining: $\pi q_1=q_1$ and $\pi(q_2-q_3)=0$. Thus V_* is the sub-space spanned

by $\{q_1\}$. Let us now represent T_{*a} in this basis. Since $T_a q_1 = q_2 - q_3$, we get $T_{*a} q_1 = \pi(q_2 - q_3) = 0$; in other words, T_{*a} is the identically 0 function on V_*. Finally, $\phi_*(q_1) = \phi(q_1) = 1$. Thus, the WA representing L_* w.r.t. the basis $\{q_1\}$ is hence given by just:

We finally show that any two minimal LWA's representing the same FPS are isomorphic, in particular they have the same dimension. This shows that (minimal) dimension is a feature of FPS's rather than of rooted LWA's. Formally, we say that two rooted LWA's (v, L) and (v', L') are *isomorphic* if there is a vector-space isomorphism $\tau : V \to V'$ that preserves the roots and commutes with the transition and weight functions of the two automata, that is: $\tau v = v'$, $\tau T_a u = T'_a \tau u$ and $\phi(u) = \phi'(\tau u)$, for each $a \in A$ and $u \in V$. Isomorphism is clearly an equivalence relation over the class of rooted LWA's.

In the following theorem, we will use the fact that $\mathbb{R}\langle\langle A \rangle\rangle$ is an (infinite-dimensional) vector space (see Appendix C) that can be naturally endowed with a LWA structure, as follows. Σ is the infinite-dimensional LWA $(\mathbb{R}\langle\langle A \rangle\rangle, \{\delta_a\}_{a \in A}, \theta)$, where

- $\delta_a(\sigma) \triangleq \lambda x.\sigma(ax)$ *(a-derivative)*
- $\theta(\sigma) \triangleq \sigma(\epsilon)$.

Given any $L = (V, \{T_a\}_{a \in A}, \phi)$, recall that the function $\sigma_L : V \to \mathbb{R}\langle\langle A \rangle\rangle$ is a vector-space morphism. Moreover, it is immediate to check that $\Sigma(L) \triangleq (\sigma_L(V), \{(\delta_a)_{|\sigma_L(V)}\}_{a \in A}, \phi_{|\sigma_L(V)})$ is a finite-dimensional LWA (a sub-LWA of Σ).

Theorem 6 (canonicity) *Let (v, L) and (v', L') be two minimal rooted* LWA *'s representing the same* FPS *σ. Then (v, L) and (v', L') are isomorphic.*

PROOF It is straightforward to check that σ_L makes (v, L) isomorphic to the rooted LWA $(\sigma, \Sigma(L))$; in particular, the fact that $\ker(\sigma_L) = \{0\}$ stems from $\ker(\sigma_L) = \ker(\sim_L)$ and from $\sim_L$ being the identity over V (Lemma 9). Similarly, $\sigma_{L'}$ makes (v', L') isomorphic to the rooted LWA $(\sigma, \Sigma(L'))$. Now, we show that $\Sigma(L) = \Sigma(L')$, which will imply the thesis, by transitivity of isomorphism. To see this, it is sufficient to check that $\sigma_L(V) = \sigma_{L'}(V')$. Now, since (v, L) is minimal, we know that $\sim_L^{\text{op}}$ is the identity over V (again Lemma 9). Or, equivalently, $V / \sim_L^{\text{op}} = V$; hence, by the characterization in Theorem 2, $V = \text{span}\{T_x v | x \in X\}$, for some finite $X \subseteq A^*$ (e.g. take $X = \cup_{0 \le i \le n} A^i$). Therefore, $\sigma_L(V)$ is spanned by $H = \{\sigma_L(T_x v) | x \in X\}$. Analogously, $\sigma_{L'}(V')$ is spanned by $H' = \{\sigma_{L'}(T'_x v') | x \in X\}$. Now, it is easy to check by induction on x that for each x, $\sigma_L(T_x v) = \sigma_{L'}(T'_x v')$. This implies $H = H'$, hence $\sigma_L(V) = \sigma_{L'}(V')$. □

Practically, the above theorem entails that, no matter what rooted LWA one chooses to represent a given σ, after minimizing one ends up into one and the same minimal representation. Also note that the this result can be used to check whether two rooted LWA's represent the same FPS: just check whether the corresponding minimal LWA's are isomorphic. When a matrix representation of the LWA's is adopted, checking isomorphism reduces to checking similarity, for which standard algorithms from Linear Algebra can be used. Another, more direct method, to check equivalence of two rooted LWA's, (v, L) and (v', L'), does not require minimization, but relies on checking whether v and v' are bisimilar in the direct-sum LWA $L \oplus L'$.

Intuitively, this automaton is formed by laying L and L' side by side. Formally, recall that the cartesian product of two vector spaces V and V' is still a vector space, called direct sum of V and

V' and denoted by $V \oplus V'$. Note that $V \times \{0_{V'}\}$ is a sub-space of $V \oplus V'$ that can be identified, via isomorphism, with V. Similarly for $\{0_V\} \times V'$ and V'. The *direct sum of* L *and* L', written $L \oplus L'$, is the LWA $(V \oplus V', \{T_a^{\oplus}\}_{a \in A}, \phi^{\oplus})$ where $T_a^{\oplus}(u,u') \triangleq (T_a u, T'_a u')$ and $\phi^{\oplus}(u,u') \triangleq \phi(u) + \phi(u')$.

Proposition 1 *Let* (v,L) *and* (v',L') *be two rooted* LWA*'s.* (v,L) *and* (v',L') *represent the same* FPS *if and only if* $v \sim_{L \oplus L'} v'$.

PROOF It is straightforward to check that, for any $(u,u') \in V \oplus V'$, $\sigma_{L \oplus L'}(u,u') = \sigma_L(u) + \sigma_{L'}(u')$. Applying this fact to $(v, 0_{V'})$ and $(0_V, v')$ and using Theorem 1 we have the thesis. □

8 The case of a generic field

We have restricted our presentation to LWA over the field of real numbers equipped with a inner product, so that we can easily represent both negations and quotients as orthogonal complements. In fact, while having orthogonal complements is handy, it is not at all necessary. In the case of a vector space over a generic field $\mathbb{K}$, we can replace orthogonal complements by annihilators when representing negation and by (ordinary) complement spaces when representing quotients. Doing so requires the introduction of dual spaces, which we quickly review below. An in-depth treatment cab be found in e.g. [14].

Given any vector space V over $\mathbb{K}$, its *dual space* $V^\star$ is the set of all morphisms $V \to \mathbb{K}$, with $\mathbb{K}$ seen as a 1-dimensional vector space. The elements of $V^\star$ are often called *functionals*. The sum of two functionals $\psi_1 + \psi_2$ and the scalar multiplication $k \cdot \psi$ ($k \in \mathbb{K}$) are defined point-wise as expected, and turn $V^\star$ into a vector space over $\mathbb{K}$. It is customary to denote functional application $\psi(v)$ as $[v,\psi]$, the bracket notation intending to emphasize certain analogies with inner products. Fix an ordered basis $B = (e_1, ..., e_n)$ of V and consider $B^\star \triangleq (e_1^\star, ..., e_n^\star)$, where the functionals $e_i^\star$ are specified by $[e_j, e_i^\star] \triangleq \delta_{ij}$ for each i and j – here δ_{ij} denotes the Kronecker symbol, which equals 1 if $i = j$ and 0 otherwise. It is easy to check that $B^\star$ forms a basis of $V^\star$, referred to as the *dual basis* of B. Hence $\dim(V^\star) = \dim(V)$. In particular, the morphism $(\cdot)^\star : V \to V^\star$ that sends each e_i into $e_i^\star$ is an isomorphism between V and $V^\star$. Note that if a and b are the column-vectors representing v and ψ in the respective bases B and $B^\star$, then $\psi(v)$ can be computed as the ordinary scalar product of these vectors, that is

$$[v,\psi] = \langle a,b \rangle \triangleq \sum_i a_i \cdot b_i . \tag{3}$$

The definition of transpose morphism must be extended as follows.

Definition 12 (transpose morphism, general case) *Let* $T : V \to V$ *be an endomorphism on* V*. We let the* transpose of T*, written* ${}^{\mathrm{t}}T$*, be the endomorphism* $V^\star \to V^\star$ *defined by* ${}^{\mathrm{t}}T(\psi) \triangleq \psi \circ T$ *for each* $\psi \in V^\star$*.*

It is easy to check that that if M is the matrix representing T in V w.r.t. to B, then the transpose matrix ${}^{\mathrm{t}}M$ represents ${}^{\mathrm{t}}T$ in $V^\star$ w.r.t. $B^\star$, whence the terminology and the notation. It is quite expected that, by taking the transpose twice one gets back the original morphism. In fact this is the case, although one has to take care of identifying things up to isomorphism. Denote by $V^{\star\star}$ the space $(V^\star)^\star$, called double dual of V. There is a natural isomorphism F between V and $V^{\star\star}$, given by $F : v \mapsto [v, \cdot]$ (note that this isomorphism does not depend on the choice of a basis).

In the sequel, we shall freely identify V and $V^{\star\star}$ up to this isomorphism, i.e. identify v and $[v,\cdot]$ for each $v \in V$. With this identification, one has that ${}^{\mathsf{t}}({}^{\mathsf{t}}T) = T$.

We need another concept from duality theory. Given $U \subseteq V$, we denote by U^o the *annihilator* of U, the subset of functionals that vanish on U.

Definition 13 (annihilator) *For any $U \subseteq V$, we let $U^o \triangleq \{\psi \in V^\star \mid [u,\psi] = 0$ for each $u \in U\}$.*

Once again, the notation makes the analogy with inner products explicit. We use the following properties of annihilators, where U, W are a sub-spaces of V: (i) U^o is a sub-space of $V^\star$; (ii) $(\cdot)^o$ reverses inclusions, i.e. if $U \subseteq W$ then $W^o \subseteq U^o$; (iii) $(\cdot)^o$ is an involution, that is $(U^o)^o = U$ up to the natural isomorphism between V and its double dual. These three properties suggest that U^o can be regarded as a *complement*, or negation, of U seen as a relation. Another useful property is: (iv) $\dim(U^o) + \dim(U) = \dim(V)$.

Now, consider LWA's over a generic field $\mathbb{K}$. The definitions and results of sections 2, 3 and 4 carry over essentially unchanged when replacing $U^\perp$ by U^o and the definition of transpose morphism with the general one given above. The only difference is that the R_i's given in Theorem 2 are now sub-spaces of the dual space $V^\star$, rather than V. Concerning the quotient construction in Section 5, the definitions and results carry over if identifying V/R with any complement of R w.r.t. V, i.e. with any W s.t. $V = R \oplus W$, and identifying π with the projection onto W. Concerning Section 7, the duality operation is extended as follows: fix any basis B of V and denote by $(\cdot)^\star : V \to V^\star$ the induced dualization morphism. Then $(v, L)^{\mathrm{op}}$ is the rooted LWA $(\phi, (V^\star, \{{}^{\mathsf{t}}T_a\}_{a\in A}, v^\star))$. Finally, in the construction of the minimal rooted automaton (Theorem 5) one must define V_* as the sub-space $(\sim_L^{\mathrm{op}})^o$, which is a sub-space of V up to the isomorphism between V and its double dual. With these changes, the results carry over.

9 Related and further work

Related work Our formulation of linear weighted bisimulation is primarily related to the definition of Σ-*congruence* put forward by Stark [31]. 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 [31]. Partition refinement and quotient/minimization are not tackled, though. A related equivalence for stochastic systems, under the name behaviour equivalence, is studied in [34, 32] (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 [6, 7]. A largest such bisimulation can be computed by a partition refinement algorithm that works on a matrix representation of the automata [6]; 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 [26, 25, 27] on coinduction and (multivariate) streams – another name for FPS's. The coinduction principle we state in Theorem 3 has been inspired to us by a similar principle formulated, for Moore automata, in [26]. We also note that Rutten endows FPS's with a Moore automaton structure, while here we endow them with a weighted automaton structure – see Theorem 6. In [26], weighted automata are used to provide a more compact representation for streams than deterministic (Moore) automata do. Closely related to ours is also [27], 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 [25], 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 [29], where a minimization algorithm was also discussed. This algorithm has been reformulated in a more algebraic fashion in Berstel and Reutenauer's book [4]. Other descriptions of the algorithm can be found in [8, 11, 28, 33]. Here we explicitly connect this algorithm to the notion of bisimulation. Hopefully, this connection will make the algorithm itself accessible to a larger audience.

Conclusion and further work We have introduced a notion of bisimulation for weighted automata over the field of real numbers, provided partition refinement and minimization algorithms, and studied its relationship to existing notions of equivalence on probabilistic systems and to classical constructions in Automata Theory. Both the presentation and the technical development draw some benefit from a geometrical representation called LWA. 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 [31, Section 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 [25]. Indeed, going from an infinite to a "small" automaton as described in the preceding paragraph is a quotient operation, that would be interesting to cast in our framework. Second, it would be interesting to generalize the present approach to (semi)rings, so as to capture a greater variety of equivalences, e.g. ordinary language equivalence. This would imply moving from vector spaces to *(semi)modules* [19]. Some work in this vein has already been done, see [2, 3], which consider Euclidean domains.Also, it would be interesting to cast the present results in a more explicit coalgebraic 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 [30, 9] 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] M.P. Beal, S. Lombardy and J. Sakarovitch. On the equivalence of Z-automata. *ICALP 2005*, LNCS 3580, 397–409, 2005.

[3] M.P. Beal, S. Lombardy and J. Sakarovitch. Conjugacy and equivalence of weighted automata and functional transducers. *CSR 2006*, LNCS 3967, 58–69, 2006.

[4] 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`.

[5] P. Buchholz. Exact Performance Equivalence: An Equivalence Relation for Stochastic Automata. *Theoretical Computer Science*, 215(1-2): 263-287, 1999.

[6] P. Buchholz. Bisimulation relations for weighted automata. *Theoretical Computer Science* 393(1-3): 109-123, 2008.

[7] P. Buchholz, P. Kemper. Quantifying the Dynamic Behavior of Process Algebras. *PAPM-PROBMIV* 2001: 184-199, 2001.

[8] A. Cardon and M. Crochemore. Determination de la representation standard d'une serie reconnaissable. *RAIRO Theor. Informatics and Appl.* 14: 371–379, 1980.

[9] 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.

[10] R. De Nicola, Matthew Hennessy. Testing Equivalences for Processes. *Theoretical Compututer Science* 34: 83-133, 1984.

[11] M. Flouret and E. Laugerotte. Noncommutative minimization algorithms. *Inform. Process. Lett.* 64: 123–126, 1997.

[12] 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.

[13] G. H. Golub, C.F. Van Loan. *Matrix Computations*, 2/e. The John Hopkins University Press, 1989.

[14] P. R. Halmos. *Finite-Dimensional Vector Spaces*. Springer, 1987.

[15] J. Hillston. Compositional Markovian modelling using a process algebra. In: W.J. Stewart (Ed.), *Computations with Markov Chains*, Kluwer Academic Publisher, 177-196, 1995.

[16] A. Isidori. *Nonlinear Control Systems*. Lecture Notes in Control and Information Sciences, Springer-Verlag, 1989.

[17] B. Jonsson, K. G. Larsen. Specification and Refinement of Probabilistic Processes, *LICS 1991*: 266-277, 1991.

[18] J.G. Kemeny, J.L. Snell. *Finite Markov Chains*. Springer-Verlag, 1976.

[19] S.A. Lang. *Introduction to Linear Algebra*, 2/e. Springer-Verlag, 1997.

[20] K.G. Larsen, A. Skou. Bisimulation through Probabilistic Testing. *Information and Compututation*, 94(1): 1-28, 1991.

[21] A. R. Meyer, L. J. Stockmeyer. Word problems requiring exponential time. In *STOC 1973*: 1-9, 1973.

[22] R. Milner. *A Calculus of Communicating Systems*. Prentice-Hall, 1989.

[23] M. Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics* 23(2): 269-311, 1997.

[24] D. Park. Concurrency and Automata on Infinite Sequences. *Theoretical Computer Science* 1981: 167-183.

[25] J.J.M.M. Rutten. Coinductive counting with weighted automata. *Journal of Automata, Languages and Combinatorics* 8(2): 319-352, 2003.

[26] 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.

[27] J.J.M.M. Rutten. Rational streams coalgebraically. *Logical Methods in Computer Science*, 4(3), 2008.

[28] J. Sakarovitch. *Elements de theorie des automates*, Vuibert, 2003.

[29] M. P. Schützenberger. On the Definition of a Family of Automata. *Information and Control*, 4(2-3): 245-270, 1961.

[30] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT. 1995.

[31] 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.

[32] E. W. Stark, R. Cleaveland, S.A. Smolka: Probabilistic I/O Automata: Theories of Two Equivalences. *CONCUR 2006*, LNCS 4137:343-357, 2006.

[33] E. W. Stark, W. Song. Linear Decision Diagrams, manuscript, 2004. Available from `http://bsd7.starkhome.cs.sunysb.edu/~stark/REPORTS/ldd.pdf`.

[34] 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.

A Linear algebraic background

Let V, W be two vector spaces and let $T : V \to W$ be a morphism between them (we shall also call T an *endomorphism* if $V = W$). $T(V)$ is a sub-space of W and $\ker(T) \triangleq \{v \in V \mid T(v) = 0\}$ is a sub-space of V. The following fundamental identity relates the dimensions of these spaces in case $\dim(V)$ is finite:

$$\dim(V) = \dim(T(V)) + \dim(\ker(T)). \tag{4}$$

For any subset $U \subseteq V$, the sub-space *spanned by* U, denoted $\mathrm{span}(U)$, is the set of linear combinations of elements from U; this is the smallest sub-space containing U. The *sum* of n sub-spaces $V_1, ..., V_n$ is defined as $\sum_{i=1}^{n} V_i \triangleq \mathrm{span}(\cup_{i=1}^{n} V_i)$. Note that $\mathrm{span}(\cdot)$ commutes with T in the sense that $T(\mathrm{span}(U)) = \mathrm{span}(T(U))$; in particular, $T(\sum_{i=1}^{n} V_i) = \sum_{i=1}^{n} T(V_i)$. Suppose V has finite dimension n. An *ordered basis* B for V is just a n-tuple $B = (e_1, \cdots, e_n)$ of distinct vectors that form a basis of V. If $T : V \to W$ and B and B' are ordered bases of V and W, of cardinality m and n, respectively, then T is *represented by* the matrix $M \in \mathbb{R}^{n \times m}$ that has as its i^{th} column the column-vector of the coordinates of $T(e_i)$ in the basis B'; the coordinates of $T(v)$ in the basis B' are obtained by the matrix multiplication sM, where s is the row-vector of the coordinates of v in B.

A *inner product* over V is a symmetric, bi-linear application $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ that is *definite-positive*, that is $\langle v, v \rangle > 0$ for each $v \neq 0$. A vector space over $\mathbb{R}$ equipped with an inner product is called *inner product space*. Let V be a inner product space. Two vectors u and v are *orthogonal* if $\langle u, v \rangle = 0$. An *orthogonal basis* of V is a basis of V whose elements are pairwise orthogonal. An *ortho-normal basis* of V is an orthogonal basis B whose elements have norm 1, that is $\langle e, e \rangle = 1$ for each $e \in B$. Any finite-dimensional inner product space – hence any sub-space of it – is proven to have an ortho-normal basis. Given any sub-space $U \subseteq V$, the set $U^{\perp} \triangleq \{v \in V \mid \langle v, u \rangle = 0 \text{ for each } u \in U\}$ is itself a sub-space of V, called the *orthogonal complement* of U w.r.t. V. If V is finite-dimensional, then each $v \in V$ can be written in *a unique way* as a sum $v = u + w$ with $u \in U$ and $w \in U^{\perp}$; this is also written as $V = U \oplus U^{\perp}$ and, in this case, $\dim(V) = \dim(U) + \dim(U^{\perp})$. Moreover, in finite-dimensional inner products, the orthogonal complement operation is an involution, that is $(U^{\perp})^{\perp} = U$, and reverses inclusions, that is, if $U \subseteq W$ then $U^{\perp} \supseteq W^{\perp}$.

Let Q be a finite set. Denote by $\mathbb{R}^Q$ the set of all functions $f : Q \to \mathbb{R}$, called *measures* over Q. We observe that $\mathbb{R}^Q$ is a vector space over $\mathbb{R}$ of dimension $|Q|$, the *free vector space* generated by Q, once we define the sum of two measures, $f + g$, and the scalar product $r \cdot f$ ($r \in \mathbb{R}$) point-wise as expected. Q itself can be seen as a subset of this space, once every $q \in Q$ is identified with the "point-measure" f_q defined by: $f_q(q') = 1$ if $q' = q$ and $f_q(q') = 0$ otherwise; in the sequel, we will make no notational distinction between q and f_q. In fact, Q is a ortho-normal basis of $\mathbb{R}^Q$ w.r.t. to the inner product defined as $\langle \sum_{q \in Q} r_q \cdot q, \sum_{q \in Q} r'_q \cdot q \rangle \triangleq \sum_{q \in Q} r_q \cdot r'_q$.

B Similarity

We have argued in Section 2 that the matrix-based representation of WA is basically equivalent to that in terms of LWA. It should be noted, however, that, when passing from a LWA to a WA, choosing two different bases of V, say Q and Q', will in general determine two different WA's, $W_{L,Q}$ and $W_{L,Q'}$. These two automata have the same number of states, but need not have isomorphic transition graphs. Rather, these two automata are similar, in the following sense.

Definition 14 (similarity) *Given two* WA*'s,* $W = (Q, \{M_a\}_{a\in A}, f)$ *and* $W' = (Q', \{M'_a\}_{a\in A}, f')$*, we say* W *is* N-similar to W' *if* N *is an invertible square matrix such that* $M_a = NM'_aN^{-1}$*, for each* $a \in A$*, and* $f = Nf'$*.* W *and* W' *are* similar *if they are* N*-similar for some* N*.*

Similarity is, evidently, an equivalence relation on the class of WA's. In case W and W' are derived from the same L via different bases Q and Q', N is just the change of basis matrix from Q to Q'. The states of two similar automata have the same weighted language semantics, modulo a transformation of coordinates. These considerations are made precise and recorded in the proposition below.

Proposition 2 *(1) Let* W *be a* n*-dimensional* WA N*-similar to* W'*. For each row-vector* $s \in \mathbb{R}^{1\times n}$*, it holds that* $\sigma_W(s) = \sigma_{W'}(sN)$*. (2) Let* L *be a* LWA *and* Q*,* Q' *be two ordered bases of* V*. Then* $W_{L,Q}$ *and* $W_{L,Q'}$ *are similar. (3) Conversely, two* WA*'s* W *and* W' *over the same ordered set of states are similar if and only if* $L_W = L_{W'}$*.*

Part (1) of Proposition 2 says that similar automata have the same weighted language semantics, modulo a change coordinates. Parts (2) and (3) say that working with LWA's means essentially working with WA's up to similarity. Working up to similarity provides a basic level of abstraction – somewhat analogous to the use of structural congruence in process calculi – which is technically convenient. On the other hand, when it comes to represent concretely a LWA, one has a certain freedom as to the choice of the basis, hence of the corresponding WA.

C Proofs

Lemma 10 *Suppose* L *is minimal. Then* $\sim_L$ *is the identity over* V*.*

PROOF By minimality of L and $L/\sim_L$ and part (a) of Theorem 3: $\dim(L) = \dim(L/\sim_L) = \dim(L) - \dim(\sim_L)$, hence $\dim(\sim_L) = 0$. □

Lemma 11 (Lemma 7) *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* $a \in A$*,* $|t_a f|_C = |t_a g|_C$*.*

PROOF For any $h \in \mathbb{R}^Q$, say $h = \sum_{q\in Q} r_q q$, and any $X \subseteq Q$, let us write h_X for $\sum_{q\in X} r_q q$. Let $C_1, ..., C_n$ be the equivalence classes of $Q/\mathcal{S}$. Now, take f and g as in the statement of the lemma. Clearly, $f = f_{C_1} + \cdots + f_{C_n}$ and $g = g_{C_1} + \cdots + g_{C_n}$. Hence, by linearity of t_a

$$\begin{aligned} t_a f &= t_a(f_{C_1}) + \cdots + t_a(f_{C_n}) \\ t_a g &= t_a(g_{C_1}) + \cdots + t_a(g_{C_n}). \end{aligned}$$

From these equalities and linearity of $|\cdot|_C$, it follows that, for each C, $|t_a f|_C = \sum_i |t_a(f_{C_i})|_C$, and similarly $|t_a g|_C = \sum_i |t_a(g_{C_i})|_C$. Therefore, to prove that $|t_a f|_C = |t_a g|_C$, it suffices to show that for each pair of classes C_i and C, $|t_a(f_{C_i})|_C = |t_a(g_{C_i})|_C$. To this purpose, write f_{C_i} as $\sum_{q\in C_i} \lambda_q q$ and g_{C_i} as $\sum_{q\in C_i} \lambda'_q q$. By linearity of t_a and $|\cdot|_X$, we have that $|t_a f_{C_i}|_C = \sum_{q\in C_i} \lambda_q |t_a q|_C = (\sum_{q\in C_i} \lambda_q)k = |f|_{C_i} k$, where k is a constant s.t. $|t_a q|_C = k$ for each $q \in C_i$: that all these quantities $|t_a q|_C$ are the same stems from the fact that C_i is an equivalence class of the probabilistic bisimulation $\mathcal{S}$. Similarly, $|g_{C_i}|_C = |g|_{C_i} k$. Since $|f|_{C_i} = |g|_{C_i}$ by assumption, the thesis follows. □

Theorem 7 (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 Let $\sim_P^\ell$ to be the linear relation 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 $C \in Q/\sim_P$, $|t_a f|_C = |t_a g|_C$. By definition, $q \sim_P q'$ implies $q \sim_P^\ell q'$. We will show that $\sim_P^\ell$ is included in $\sim_{\hat{P}}$, the largest linear weighted bisimulation on $\hat{P}$, this will prove the theorem. To this aim, it suffices to show that $\sim_P^\ell$ is a $\hat{P}$-bisimulation. Assume $f, g \in \mathbb{R}^Q$ and $f \sim_P^\ell g$. We check the obligations (a) and (b) of the definition of linear weighted bisimulation.

(a) $\phi(f) = \phi(g)$. Note that, by definition of ϕ, $\phi(h) = |h|$ for each $h \in \mathbb{R}^Q$. The thesis then follows from the definition of $\sim_P^\ell$.

(b) Let $a \in A$, we have to show that $t_a f \sim_P^\ell t_a g$. Since $f \sim_P^\ell g$, we know that for each $C \in Q/\sim_P$, $|t_a f|_C = |t_a g|_C$. Hence: (i) $|t_a f| = \sum_C |t_a f|_C = \sum_C |t_a g|_C = |t_a g|$; and, (ii) by virtue of Lemma 7 $|t_b t_a f|_C = |t_b t_a g|_C$, for every C and $b \in A$. This proves that $t_a f \sim_P^\ell t_a g$.

□

Lemma 12 (Lemma 8) *(v, L) is minimal if and only if $(v, L)^{\mathrm{op}}$ is minimal.*

PROOF Let $(u, L') = (v, L)^{\mathrm{op}}$. For any string $x = a_1 \cdots a_n \in A^*$, denote by ${}^{\mathrm{t}}x$ the reverse string $a_n \cdots a_1$. It is immediate to check, representing the involved morphisms in any ortho-normal basis of V, that for each $x \in A^*$, $\sigma_L(v)(x) = \sigma_{L'}(u)({}^{\mathrm{t}}x)$. This implies that if (w, L'') is a reduct of (v, L) then $(w, L'')^{\mathrm{op}}$ is a reduct of $(v, L)^{\mathrm{op}}$. The thesis follows by idempotency of $(\cdot)^{\mathrm{op}}$. □

In the lemma below, we use the fact that the set of FPS's, $\mathbb{R}\langle\langle A\rangle\rangle$, is a vector space over $\mathbb{R}$, once the operations of sum and scalar product are defined point-wise as expected: $(\sigma + \sigma')(x) \triangleq \sigma(x) + \sigma'(x)$ and $(r \cdot \sigma)(x) = r \cdot \sigma(x)$ $(r \in \mathbb{R})$. Moreover, for any L with V as a space-state, the map $v \mapsto \sigma_L(v)$ is a morphism $V \to \mathbb{R}\langle\langle A\rangle\rangle$.

Lemma 13 (Lemma9) *A rooted* LWA *(v, L) is minimal if and only if the bisimulations $\sim_L$ and $\sim_L^{\mathrm{op}}$ both coincide with the identity equivalence relation on V.*

PROOF Assume first (v, L) is minimal. Let (h, L') be any reduction of L. By definition, (hv, L') is a reduct of (v, L). Hence it must be $\dim(L) = \dim(L')$. In other words, L is a minimal LWA. Hence, by Lemma 10, $\sim_L$ is the identity over V. On the other hand, also $(v, L)^{\mathrm{op}}$ is minimal (Lemma 8), hence, by the same argument, also $\sim_L^{\mathrm{op}}$ is the identity over V.

Conversely, assume that both $\sim_L$ and $\sim_L^{\mathrm{op}}$ are the identity relation over V. Hence $\ker(\sim_L^{\mathrm{op}}) = 0$, and $V = V/\sim_L^{\mathrm{op}}$. By the characterization given in Theorem 2, and recalling that for the final weight function of the dual automaton we have $\ker(\psi)^\perp = \mathrm{span}\{v\}$, it must be for some finite $X \subseteq A^*$ (e.g. take $X = \cup_{0\le i\le n} A^i$)

$$V \;=\; V/\sim_L^{\mathrm{op}} \;=\; \mathrm{span}\{T_x v \mid x \in X\}.$$

Consider the map σ_L as a morphism $V \to \mathbb{R}\langle\langle A\rangle\rangle$. By the equation above for V, we get that the sub-space $\sigma_L(V) \subseteq \mathbb{R}\langle\langle A\rangle\rangle$ is spanned by $H = \{\sigma_L(T_x v) \mid x \in X\}$. Now, let (v', L') be any reduct of (v, L), with $L' = (V', \{T'_a\}_{a\in A}, \phi')$. We want to show that $\dim(V') = \dim(V)$. To this purpose, first observe that, for any string x, we have that $\sigma_L(T_x v) = \sigma_{L'}(T'_x v')$: this is easily shown by induction on x. Hence, $H = \{\sigma_L(T_x v) \mid x \in X\} = \{\sigma_{L'}(T'_x v') \mid x \in X\} \subseteq \sigma_{L'}(V')$. Since H spans $\sigma_L(V)$, we get that $\dim(\sigma_L(V)) \le \dim(\sigma_{L'}(V'))$. Moreover, it is immediate to check (via Theorem 1) that $\ker(\sigma_L) = \sim_L$.

But $\sim_L$ is by assumption the identity, that is $\dim(\sim_L) = 0$. We now can use the facts collected thus far and the fundamental identity (4) to compare $\dim(V)$ and $\dim(V')$, as follows

$$\begin{aligned}
\dim(V) &= \dim(\sigma_L(V)) + \dim(\ker(\sigma_L)) \;=\; \dim(\sigma_L(V)) \\
&\leq \dim(\sigma_{L'}(V')) \;\leq\; \dim(\sigma_{L'}(V')) + \dim(\ker(\sigma_{L'})) \\
&= \dim(V') \;\leq\; \dim(V)
\end{aligned}$$

that is, $\dim(V) = \dim(V')$. □