Featured Weighted Automata

Uli Fahrenberg*
Ecole polytechnique, Palaiseau, France
uli@lix.polytechnique.fr

Axel Legay
Inria Rennes, France
axel.legay@inria.fr

ABSTRACT

A featured transition system is a transition system in which the transitions are annotated with feature expressions: Boolean expressions on a finite number of given features. Depending on its feature expression, each individual transition can be enabled when some features are present, and disabled for other sets of features. The behavior of a featured transition system hence depends on a given set of features. There are algorithms for featured transition systems which can check their properties for all sets of features at once, for example for LTL or CTL properties.

Here we introduce a model of featured weighted automata which combines featured transition systems and (semiring-) weighted automata. We show that methods and techniques from weighted automata extend to featured weighted automata and devise algorithms to compute quantitative properties of featured weighted automata for all sets of features at once. We show applications to minimum reachability and to energy properties.

1 INTRODUCTION

A *featured transition system* [6] is a transition system in which the transitions are annotated with *feature expressions*: Boolean expressions involving a finite number of given features. Depending on its feature expression, each individual transition can be enabled when some features are present, and disabled for other sets of features. For any set of features, a given featured transition system *projects* to a transition system which contains precisely the transitions which are enabled for that set of features.

Standard problems such as reachability or safety can be posed for featured transition systems, where the interest now is to check these properties *for all sets of features at once*. Hence, for example for reachability, given a featured transition system and a set of accepting states, one wants to construct a feature expression ϕ such that an accepting state is reachable iff the set of features satisfies ϕ.

*Most of this work was carried out while this author was still employed at Inria Rennes.

For *quantitative* properties of transition systems, the model of *(semiring-) weighted automata* has proven useful [10]. This provides a uniform framework to treat problems such as minimum reachability, maximum flow, energy problems [13], and others. Here we extend techniques from weighted automata to *featured weighted automata, i.e.,* weighted automata in which the transitions are annotated with feature expressions. This extension makes it possible to check quantitative properties for all sets of features at once.

To be precise, a featured transition system induces a (projection) function from sets of features to transition systems, mapping each set of features to the behavior under these features. Similarly, we will define projections of featured weighted automata, mapping sets of features to weighted automata. *Values* of weighted automata are an abstract encoding of their behavior; we will see how to compute values of featured weighted automata as functions from feature expressions to behaviors.

We also develop an application of our techniques to featured *energy problems*. Energy problems are important in areas such as embedded systems or autonomous systems. They are concerned with the question whether a given system admits infinite schedules during which (1) certain tasks can be repeatedly accomplished and (2) the system never runs out of energy (or other specified resources). Starting with [3], formal modeling and analysis of such problems has attracted some attention [2, 4, 5, 9, 14, 18, 22].

Featured transition systems have applications in *software product lines*, where they are used as abstract representations of the behaviors of variability models [23]. This representation allows one to analyze all behaviors of a software product line at once, as opposed to analyzing each product on its own. Similarly, featured weighted automata can be used as abstract representations of quantitative behaviors of software product lines, and the present work enables analysis of quantitative behaviors of all products in a software product line at once.

Contributions and structure of the paper. We start in Sect. 2 by revisiting minimum reachability in featured transition systems with transitions weighted by real numbers. This has to some extent already been done in [8], but we reformulate it in order to prepare for the generalization in the following sections.

In Sect. 3, we introduce featured weighted automata and show some first examples. Instead of semirings, we will work with (featured) automata weighted in *∗-continuous Kleene algebras*; this is for convenience of presentation only, and all our work (except for Sect. 5) can be extended to a more general (for example non-idempotent) setting. In Sect. 4, we then show how methods and techniques from weighted automata can be transferred to featured weighted automata.

In the last Sect. 5, we extend our results to develop an application to featured energy problems. This is based on the recent result in [13] that energy problems can be stated as Büchi problems in automata weighted in *∗-continuous Kleene ω-algebras*, which are

certain types of semimodules over *-continuous Kleene algebras; hence we need to extend our results to such semimodules.

The paper is followed by a separate appendix which contains some of the proofs of our results.

2 MINIMUM REACHABILITY IN REAL-WEIGHTED FEATURED AUTOMATA

A *real-weighted automaton* $\mathcal{S} = (S, I, F, T)$ consists of a finite set S of states, subsets $I, F \subseteq S$ of initial and accepting states, and a finite set $T \subseteq S \times \mathbb{R}_{\geq 0} \times S$ of weighted transitions. Here $\mathbb{R}_{\geq 0}$ denotes the set of non-negative real numbers.

A *finite path* in such a real-weighted automaton $\mathcal{S}$ is a finite alternating sequence $\pi = (s_0, x_0, s_1, x_1, \ldots, x_k, s_{k+1})$ of transitions $(s_0, x_0, s_1), \ldots, (s_k, x_k, s_{k+1}) \in T$. The *weight* of π is the sum $w(\pi) = x_0 + \cdots + x_k \in \mathbb{R}_{\geq 0}$. A finite path π as above is said to be *accepting* if $s_0 \in I$ and $s_{k+1} \in F$. The *minimum reachability problem* for real-weighted automata asks, given a real-weighted automaton $\mathcal{S}$ as above, to compute the value

$$|\mathcal{S}| = \inf\{w(\pi) \mid \pi \text{ accepting finite path in } \mathcal{S}\}.$$

That is, $|\mathcal{S}|$ is the minimum weight of all finite paths from an initial to an accepting state in $\mathcal{S}$. This being a multi-source-multi-target shortest path problem, it can for example be solved using the Floyd-Warshall relaxation algorithm.

Let N be a set of *features* and $px \subseteq 2^N$ a set of *products* over N. A *feature guard* is a Boolean expression over N, and we denote the set of these by $\mathbb{B}(N)$. We write $p \models \gamma$ if $p \in px$ satisfies $\gamma \in \mathbb{B}(N)$ and $[\![\gamma]\!] = \{p \in px \mid p \models \gamma\}$. Note that $[\![\gamma]\!]$ is a set of sets of features.

Definition 2.1. A *real-weighted featured automaton* (S, I, F, T, γ) consists of a finite set S of states, subsets $I, F \subseteq S$ of initial and accepting states, a finite set $T \subseteq S \times \mathbb{R}_{\geq 0} \times S$ of weighted transitions, and a feature guard mapping $\gamma : T \to \mathbb{B}(N)$.

The *projection* of a real-weighted featured automaton $\mathcal{F} = (S, I, F, T, \gamma)$ to a product $p \in px$ is the real-weighted automaton $\text{proj}_p(\mathcal{F}) = (S, I, F, T')$ with $T' = \{t \in T \mid p \models \gamma(t)\}$.

For each product $p \in px$, we could solve the shortest path problem in $\text{proj}_p(\mathcal{F})$ by computing $|\text{proj}_p(\mathcal{F})|$. Instead, we develop an algorithm which computes all these values at the same time. Its output will, thus, be a function $|\mathcal{F}| : px \to \mathbb{R}_{\geq 0}$, with the property that for every $p \in px$, $|\mathcal{F}|(p) = |\text{proj}_p(\mathcal{F})|$.

As a symbolic representation of functions $px \to \mathbb{R}_{\geq 0}$, we use injective functions from *guard partitions* to $\mathbb{R}_{\geq 0}$. Intuitively, a guard partition is a set of feature guards which partitions px into classes such that within each class, f has the same value for all products, and between different classes, f has different values.

Definition 2.2. A *guard partition* of px is a set $P \subseteq \mathbb{B}(N)$ such that $[\![\bigvee P]\!] = px$, $[\![\gamma]\!] \neq \emptyset$ for all $\gamma \in P$, and $[\![\gamma_1]\!] \cap [\![\gamma_2]\!] = \emptyset$ for all $\gamma_1, \gamma_2 \in P$ with $\gamma_1 \neq \gamma_2$. The set of all guard partitions of px is denoted $GP \subseteq 2^{\mathbb{B}(N)}$.

A guard partition is a logical analogue to a partition of the set of products px: any guard partition induces a partition of px, and any partition of px can be obtained by a guard partition. In particular, for any guard partition P and any product px, there is precisely one $\gamma \in P$ for which $px \models \gamma$.

Let $GP[\mathbb{R}_{\geq 0}] = \{f : P \to \mathbb{R}_{\geq 0} \mid P \in GP, \forall \gamma_1, \gamma_2 \in P : \gamma_1 \neq \gamma_2 \Rightarrow f(\gamma_1) \neq f(\gamma_2)\}$ denote the set of injective functions from guard partitions to $\mathbb{R}_{\geq 0}$.

We use *injective* functions $P \to \mathbb{R}_{\geq 0}$ as symbolic representations of functions $px \to \mathbb{R}_{\geq 0}$, because they provide the most *concise* such representation. Indeed, if a function $f : P \to \mathbb{R}_{\geq 0}$ is not injective, then there are feature guards $\gamma_1, \gamma_2 \in P$ for which $f(\gamma_1) = f(\gamma_2)$, so we can obtain a more concise representation of f by letting $P' = P \setminus \{\gamma_1, \gamma_2\} \cup \{\gamma_1 \vee \gamma_2\}$ and $f' : P' \to \mathbb{R}_{\geq 0}$ be defined by $f'(\delta) = f(\delta)$ for $\delta \neq \gamma_1 \vee \gamma_2$ and $f'(\gamma_1 \vee \gamma_2) = f(\gamma_1)$.

We show in Fig. 1 an algorithm to compute a symbolic representation of $|\mathcal{F}|$. The algorithm performs, in lines 13 to 16, a symbolic Floyd-Warshall relaxation to compute a matrix D which as entries $D(i, j)$ has functions in $GP[\mathbb{R}_{\geq 0}]$ that for each product return the shortest path from state s_i to state s_j.

The relaxation procedure $\text{RELAX}(i, j, k)$ is performed by comparing $D(i, j)$ to the sum $D(i, k) + D(k, j)$ and updating $D(i, j)$ if the sum is smaller. The result of the comparison depends on the products for which the different paths are enabled, hence the comparison and update are done for each feature expression γ_1 in the partition for $D(i, j)$ and all feature expressions γ_2, γ_3 in the partitions for $D(i, k)$ and $D(k, j)$, respectively. The comparison has to be done only if these partitions overlap (line 30), and in case the sum is smaller, $D(i, j)$ is updated in a call to a split-and-combine procedure.

Using the procedure SPLIT, in lines 32 to 41, $D(i, j)$ is updated at the $\gamma_1 \wedge (\gamma_2 \wedge \gamma_3)$ part of its partition. If $[\![\gamma_1 \wedge (\gamma_2 \wedge \gamma_3)]\!]$ is not smaller than $[\![\gamma_1]\!]$ (line 33), then $D(i, j)(\gamma_1)$ is set to its new value. Afterwards, we need to call a COMBINE procedure to see whether $D(i, j)$ has the same value at any other part δ of its partition (line 44) and, in the affirmative case, to update the partition of $D(i, j)$ by joining the two parts (line 46f).

If the feature expression $\gamma_1 \wedge (\gamma_2 \wedge \gamma_3)$ on which to update $D(i, j)$ is a strict subset of γ_1 (line 36), then the γ_1 part of the partition of $D(i, j)$ needs to be split into two parts: $\gamma_1 \wedge (\gamma_2 \wedge \gamma_3)$, on which $D(i, j)$ is to be updated, and $\gamma_1 \wedge \neg(\gamma_2 \wedge \gamma_3)$, on which its value stays the same. Again, we need to call the COMBINE procedure afterwards to potentially combine feature expressions in the partition of $D(i, j)$.

Once relaxation has finished in line 17, we need to find $f := \min\{D(i, j) \mid s_i \in I, s_j \in F\}$. As this again depends on which features are present, we need to compute this minimum in a way similar to what we did in the RELAX procedure: for each feature expression in the partition P of f and each overlapping feature expression in the partition of $D(i, j)$, we compare the two values and use the SPLIT procedure to update f if $D(i, j)$ is smaller.

A variant of the algorithm in Fig. 1 has been implemented in [21], as part of an effort to compute minimum limit-average cost in real-weighted featured automata. Several experiments in [21] show that our algorithm is significantly faster than an approach which separately solves the minimum reachability problem for each product.

3 FEATURED WEIGHTED AUTOMATA

We proceed to introduce a generalization of the setting in the previous section. Here $\mathbb{R}_{\geq 0}$ is replaced by an abstract *-*continuous Kleene algebra*. This allows us to develop an abstract setting for

1: **Input:** real-weighted featured automaton $\mathcal{F} = (S, I, F, T, \gamma)$
   with $S = \{s_1, \ldots, s_n\}$
2: **Output:** function $|\mathcal{F}| \in GP[\mathbb{R}_{\geq 0}]$

3: **var** $D : \{1, \ldots, n\} \times \{1, \ldots, n\} \rightarrow GP[\mathbb{R}_{\geq 0}]$
4: **var** P, f
5: **for** $i \leftarrow 1$ to n **do**
6:    **for** $j \leftarrow 1$ to n **do**
7:       $\mathrm{dom}(D(i,j)) \leftarrow \{\mathbf{tt}\}$
8:       $D(i,j)(\mathbf{tt}) \leftarrow \infty$
9:       **for all** $(s_i, x, s_j) \in T$ **do**
10:          **for all** $\gamma \in \mathrm{dom}(D(i,j))$ **do**
11:             **if** $[\![\gamma \wedge \gamma(s_i, x, s_j)]\!] \neq \emptyset$ and $D(i,j)(\gamma) > x$ **then**
12:                $\textsc{Split}(D(i,j), \gamma, \gamma(s_i, x, s_j), x)$
13: **for** $i \leftarrow 1$ to n **do**
14:    **for** $j \leftarrow 1$ to n **do**
15:       **for** $k \leftarrow 1$ to n **do**
16:          $\textsc{Relax}(i, j, k)$
17: $P \leftarrow \{\mathbf{tt}\}; f(\mathbf{tt}) \leftarrow \infty$
18: **for all** $s_i \in I$ **do**
19:    **for all** $s_j \in F$ **do**
20:       **for all** $\gamma_1 \in P$ **do**
21:          **for all** $\gamma_2 \in \mathrm{dom}(D(i,j))$ **do**
22:             **if** $[\![\gamma_1 \wedge \gamma_2]\!] \neq \emptyset$ and $f(\gamma_1) > D(i,j)(\gamma_2)$ **then**
23:                $\textsc{Split}(f, \gamma_1, \gamma_2, D(i,j)(\gamma_2))$
24: **return** f

25: **procedure** $\textsc{Relax}(i, j, k)$
26:    **for all** $\gamma_1 \in \mathrm{dom}(D(i,j))$ **do**
27:       **for all** $\gamma_2 \in \mathrm{dom}(D(i,k))$ **do**
28:          **for all** $\gamma_3 \in \mathrm{dom}(D(k,j))$ **do**
29:             **if** $[\![\gamma_1 \wedge \gamma_2 \wedge \gamma_3]\!] \neq \emptyset$ **then**
30:                **if** $D(i,j)(\gamma_1) > D(i,k)(\gamma_2) + D(k,j)(\gamma_3)$ **then**
31:                   $\textsc{Split}(D(i,j), \gamma_1, \gamma_2 \wedge \gamma_3, D(i,k)(\gamma_2) + D(k,j)(\gamma_3))$

32: **procedure** $\textsc{Split}(f : P \rightarrow \mathbb{R}_{\geq 0}, \gamma_1, \gamma_2 \in \mathbb{B}(N), x \in \mathbb{R}_{\geq 0})$
33:    **if** $[\![\gamma_1]\!] = [\![\gamma_1 \wedge \gamma_2]\!]$ **then**
34:       $f(\gamma_1) \leftarrow x$
35:       $\textsc{Combine}(f, \gamma_1)$
36:    **else**
37:       $y \leftarrow f(\gamma_1)$
38:       $P \leftarrow P \setminus \{\gamma_1\} \cup \{\gamma_1 \wedge \gamma_2, \gamma_1 \wedge \neg\gamma_2\}$
39:       $f(\gamma_1 \wedge \neg\gamma_2) \leftarrow y$
40:       $f(\gamma_1 \wedge \gamma_2) \leftarrow x$
41:       $\textsc{Combine}(f, \gamma_1 \wedge \gamma_2)$

42: **procedure** $\textsc{Combine}(f : P \rightarrow \mathbb{R}_{\geq 0}, \gamma \in \mathbb{B}(N))$
43:    $x \leftarrow f(\gamma)$
44:    **for all** $\delta \in P \setminus \{\gamma\}$ **do**
45:       **if** $f(\delta) = f(\gamma)$ **then**
46:          $P \leftarrow P \setminus \{\delta, \gamma\} \cup \{\delta \vee \gamma\}$
47:          $f(\delta \vee \gamma) \leftarrow x$
48:          **break**

Figure 1: Algorithm to compute $|\mathcal{F}|$ for a real-weighted featured automaton $\mathcal{F}$.

analysis of *featured weighted automata*, and to re-use our techniques developed in the previous section to solve quantitative problems in other concrete settings.

3.1 Weighted Automata

Recall that a *semiring* [10] $K = (K, \oplus, \otimes, 0, 1)$ consists of a commutative monoid $(K, \oplus, 0)$ and a monoid $(K, \otimes, 1)$ such that the distributive and zero laws

$$x(y \oplus z) = xy \oplus xz \qquad (y \oplus z)x = yx \oplus zx \qquad 0 \otimes x = 0 = x \otimes 0$$

hold for all $x, y, z \in K$ (here we have omitted the multiplication sign $\otimes$ in some expressions, and we shall also do so in the future). It follows that the product distributes over all finite sums.

A (finite) *weighted automaton* [10] over a semiring K (or a K-*weighted automaton* for short) is a tuple $\mathcal{S} = (S, I, F, T)$ consisting of a finite set S of states, a subset $I \subseteq S$ of initial states, a subset $F \subseteq S$ of accepting states, and a finite set $T \subseteq S \times K \times S$ of transitions.

A *finite path* in such a K-weighted automaton $\mathcal{S} = (S, I, F, T)$ is a finite alternating sequence $\pi = (s_0, x_0, s_1, \ldots, x_k, s_{k+1})$ of transitions $(s_0, x_0, s_1), \ldots, (s_k, x_k, s_{k+1}) \in T$. The *weight* of π is the product $w(\pi) = x_0 \cdots x_k \in K$. A finite path π as above is said to be *accepting* if $s_0 \in I$ and $s_{k+1} \in F$. The *reachability value* $|\mathcal{S}|$ of $\mathcal{S}$ is defined to be the sum of the weights of all its accepting finite paths:

$$|\mathcal{S}| = \bigoplus \{w(\pi) \mid \pi \text{ accepting finite path in } \mathcal{S}\}$$

As the set of accepting finite paths generally will be infinite, one has to assume that such sums exist in K for this definition to make sense. This is the subject of Sect. 3.4 below.

3.2 Examples

The *Boolean semiring* is $\mathbb{B} = (\{\mathbf{ff}, \mathbf{tt}\}, \vee, \wedge, \mathbf{ff}, \mathbf{tt})$, with disjunction as $\oplus$ and conjunction as $\otimes$. A $\mathbb{B}$-weighted automaton $\mathcal{S}$ hence has its transitions annotated with $\mathbf{ff}$ or $\mathbf{tt}$. For a finite path $\pi = (s_0, x_0, s_1, \ldots, x_k, s_{k+1})$, we have $w(\pi) = \mathbf{tt}$ iff all $x_0 = \cdots = x_k = \mathbf{tt}$. Hence $|\mathcal{S}| = \mathbf{tt}$ iff there exists an accepting finite path in $\mathcal{S}$ which involves only $\mathbf{tt}$-labeled transitions. That is, $\mathbb{B}$-weighted automata are equivalent to ordinary (unlabeled) automata, where the equivalence consists in removing all $\mathbf{ff}$-labeled transitions.

The *tropical semiring* is $\mathbb{T} = (\mathbb{R}_{\geq 0} \cup \{\infty\}, \wedge, +, \infty, 0)$, where $\mathbb{R}_{\geq 0} \cup \{\infty\}$ denotes the set of extended real numbers, with minimum as $\oplus$ and addition as $\otimes$. The weight of a finite path is now the sum of its transition weights, and the reachability value of a $\mathbb{T}$-weighted automaton is the minimum of all its accepting finite paths' weights. Hence $\mathbb{T}$-weighted automata are precisely the real-weighted automata of Sect. 2, and to compute their reachability values is to solve the *minimum reachability* problem.

The *fuzzy semiring* is $\mathbb{F} = (\mathbb{R}_{\geq 0} \cup \{\infty\}, \vee, \wedge, 0, \infty)$, with maximum as $\oplus$ and minimum as $\otimes$. Here, the weight of a finite path is the minimum of its transition weights, and the reachability value of an $\mathbb{F}$-weighted automaton is the maximum of all its accepting finite paths' weights. This value is hence the *maximum flow* in a weighted automaton: the maximum available capacity along any finite path from an initial to a accepting state.

3.3 Featured Weighted Automata

We now extend weighted automata with features, for modeling quantitative behavior of software product lines. Let K be a semiring and denote by $GP[K] = \{f : P \to K \mid P \in GP, \forall \gamma_1, \gamma_2 \in P : \gamma_1 \neq \gamma_2 \Rightarrow f(\gamma_1) \neq f(\gamma_2)\}$ the set of injective functions from guard partitions to K.

Definition 3.1. A *featured weighted automaton* over K and *px* is a tuple (S, I, F, T) consisting of a finite set S of states, subsets $I, F \subseteq S$ of initial and accepting states, and a finite set $T \subseteq S \times GP[K] \times S$ of transitions.

Similarly to what we did in Sect. 2, the transition labels in $GP[K]$ are to be seen as syntactic representations of functions from products to K; we will say more about this below.

Example 3.2. For $K = \mathbb{B}$ the Boolean semiring, featured $\mathbb{B}$-weighted automata are standard (unlabeled) featured automata: for any feature guard $\gamma \in \mathbb{B}(N)$, $\{\gamma, \neg\gamma\}$ is a guard partition of *px*, moreover, for $K = \mathbb{B}$, any mapping in $GP[K]$ is equivalent to one from such a guard partition. Hence transitions labeled with feature guards (as in standard featured automata) are the same as transitions labeled with functions from guard partitions to $\{\mathbf{ff}, \mathbf{tt}\}$.

Definition 3.3. For $f : P \to K \in GP[K]$ and $p \in px$, let $\gamma \in P$ be the unique feature guard for which $p \models \gamma$ and define $[\![f]\!](p) = f(\gamma)$. This defines the *semantic representation of f* as the function $[\![f]\!] : px \to K$.

Definition 3.4. Let $\mathcal{F} = (S, I, F, T)$ be a featured K-weighted automaton and $p \in px$. The *projection* of $\mathcal{F}$ to p is the K-weighted automaton $\mathrm{proj}_p(\mathcal{F}) = (S, I, F, T')$, where $T' = \{(s, [\![f]\!](p), s') \mid (s, f, s') \in T\}$.

The behavior of a featured K-weighted automaton is hence given relative to products: given a featured K-weighted automaton $\mathcal{F}$ and a product p, $|\mathrm{proj}_p(\mathcal{F})|$, provided that it exists, will be the behavior of $\mathcal{F}$ when restricted to the particular product p. The purpose of this paper is to show how the values $|\mathrm{proj}_p(\mathcal{F})|$ can be computed for all $p \in px$ at once.

3.4 *-Continuous Kleene Algebras

We finish this section by introducing extra structure and properties into our semiring K which will ensure that the infinite sums $|\mathcal{S}|$ always exist. This is for convenience only, and all our work can be extended to a more general (for example non-idempotent) setting. Recall that a semiring $K = (K, \oplus, \otimes, 0, 1)$ is *idempotent* [10] if $x \oplus x = x$ for every $x \in K$.

A *-continuous Kleene algebra* [20] is an idempotent semiring $K = (K, \oplus, \otimes, 0, 1)$ in which all infinite sums of the form $\bigoplus_{n \geq 0} x^n$, $x \in K$, exist, and such that

$$x\Big(\bigoplus_{n \geq 0} y^n\Big)z = \bigoplus_{n \geq 0} xy^n z \tag{1}$$

for all $x, y, z \in K$. Intuitively, automata weighted over a *-continuous Kleene algebra allow for *loop abstraction*, in that the global effects of a loop (right-hand side of (1)) can be computed locally (left-hand side of (1)). In any *-continuous Kleene algebra K one can define a unary *star* operation $^* : K \to K$ by $x^* = \bigoplus_{n \geq 0} x^n$.

For any semiring K and $n \geq 1$, we can form the *matrix semiring* $K^{n \times n}$ whose elements are n-by-n matrices of elements of K and whose sum and product are given as the usual matrix sum and product. It is known [19] that when K is a *-continuous Kleene algebra, then $K^{n \times n}$ is also a *-continuous Kleene algebra, with the *-operation defined by $M_{i,j}^* = \bigoplus_{m \geq 0} \bigoplus \{M_{k_1,k_2} \cdots M_{k_{m-1},k_m} \mid 1 \leq k_1, \ldots, k_m \leq n, k_1 = i, k_m = j\}$ for all $M \in K^{n \times n}$ and $1 \leq i, j \leq n$. Also, if $n \geq 2$ and $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, where a and d are square matrices of dimension less than n, then

$$M^* = \begin{bmatrix} (a \oplus bd^*c)^* & (a \oplus bd^*c)^*bd^* \\ (d \oplus ca^*b)^*ca^* & (d \oplus ca^*b)^* \end{bmatrix}. \tag{2}$$

The *matrix representation* [10] of a K-weighted automaton $\mathcal{S} = (S, I, F, T)$, with $n = \#S$ the number of states, is given by the triple (α, M, κ), where $\alpha \in \{0, 1\}^n$ is the *initial vector*, $M \in K^{n \times n}$ is the *transition matrix*, and $0 \leq k \leq n$. These are given as follows: order $S = \{1, \ldots, n\}$ such that $i \in F$ iff $i \leq k$, *i.e.*, such that the first k states are accepting, and define α and M by $\alpha_i = 1$ iff $i \in I$ and $M_{i,j} = \bigoplus\{x \mid (i, x, j) \in T\}$.

It can be shown [17] that if $\mathcal{S}$ is a weighted automaton over a *-continuous Kleene algebra, then the reachability value of $\mathcal{S}$ is defined and $|\mathcal{S}| = \alpha M^* \kappa$, where $\kappa \in \{0, 1\}^n$ is the vector given by $\kappa_i = 1$ for $i \leq k$ and $\kappa_i = 0$ for $i > k$.

Example 3.5. Our example semirings $\mathbb{B}, \mathbb{T}$ and $\mathbb{F}$ share the property of being *bounded*. In general terms, a semiring K is said to be bounded [10] if $x \oplus 1 = 1$ for all $x \in K$. Note that this implies idempotency: for all $x \in K$, $x \oplus x = x(1 \oplus 1) = x \otimes 1 = x$. If K is bounded, then $x^* = 1 \oplus \cdots = 1$ for all $x \in K$, and K is a *-continuous Kleene algebra [15].

In $\mathbb{B}$, $x \oplus 1 = x \vee \mathbf{tt} = \mathbf{tt}$; in $\mathbb{T}$, $x \oplus 1 = x \wedge 0 = 0$; and in $\mathbb{F}$, $x \oplus 1 = x \vee \infty = \infty$; so these three semirings are indeed bounded. Operationally, the fact that $x^* = 1$ means that *loops can be disregarded*: for all $x, y, z \in K$, $\bigoplus_{n \geq 0} xy^n z = xy^* z = xz$. In lieu of the examples in Sect. 3.2, it is clear that this property holds for $\mathbb{B}$-, $\mathbb{T}$- and $\mathbb{F}$-weighted automata: for reachability, loops are unimportant; for minimum reachability, likewise; and for maximum flow, taking a loop can only decrease the flow, hence would be disadvantageous.

4 ANALYSIS OF FEATURED WEIGHTED AUTOMATA

Let K be a *-continuous Kleene algebra. In this section we take a closer look at the functions in $GP[K]$ and define semiring operations on them. We show that with these operations, $GP[K]$ itself is a *-continuous Kleene algebra. This means that we can treat featured K-weighted automata as $GP[K]$-weighted automata.

We first need to define an operation on partitions which turns functions $f : P \to K$ from a partition $P \in GP$ into *injective* functions, providing the most concise representation, by changing their domain. Intuitively, this *canonicalization* of f changes the partition P into a coarser one by forming disjunctions of feature guards on which f has the same value:

Definition 4.1. Let $P \in GP$ and $f : P \to K$. Introduce an equivalence relation $\sim \subseteq P \times P$ by $\gamma_1 \sim \gamma_2$ iff $f(\gamma_1) = f(\gamma_2)$ and let $P' = P/\sim$ be the quotient. Let $\tilde{P} = \{\bigvee \Gamma \mid \Gamma \in P'\}$, then $\tilde{P} \in GP$. For every $\tilde{\gamma} \in \tilde{P}$ there is an equivalence class $\Gamma \in P'$ for which $\tilde{\gamma} = \bigvee \Gamma$,

```
1: function KCombine(f : P → K): GP[K]
2:     var f̃, P̃
3:     P̃ ← ∅
4:     while P ≠ ∅ do
5:         Pick and remove γ from P
6:         x ← f(γ)
7:         for all δ ∈ P do
8:             if f(δ) = x then
9:                 γ ← γ ∨ δ
10:                P ← P \ {δ}
11:        P̃ ← P̃ ∪ {γ}
12:        f̃(γ) ← x
13:    return f̃ : P̃ → K
```

Figure 2: Function which computes canonicalization.

and f passes to these equivalence classes by definition, so we can define $\tilde{f} : \tilde{P} \to K$, the *canonicalization* of f, by $\tilde{f}(\tilde{\gamma}) = f(\Gamma)$.

We show an algorithm which implements canonicalization in Fig. 2. The function KCombine takes as input a function $f : P \to K$ and builds its canonicalization $\tilde{f} : \tilde{P} \to K$ by taking disjunctions of feature expressions in the partition P. Note the similarity of its inner loop to the Combine procedure of Fig. 1: the procedure in Fig. 1 only updates the partition of f in one place, whereas KCombine needs to check the whole partition.

LEMMA 4.2. *Let $P \in GP$, $f : P \to K$, and $\tilde{f} : \tilde{P} \to K$ the canonicalization of f. Then $\tilde{f}$ is injective, hence $\tilde{f} \in GP[K]$. Also, for any $\gamma \in P$ there is a unique element $\tilde{\gamma} \in \tilde{P}$ such that $[\![\gamma]\!] \subseteq [\![\tilde{\gamma}]\!]$.*

Definition 4.3. Let $P_1, P_2 \in GP$. The *intersection* of P_1 and P_2 is the partition $P = P_1 \wedge P_2 \in GP$ given as $P = \{\gamma_1 \wedge \gamma_2 \mid \gamma_1 \in P_1, \gamma_2 \in P_2, [\![\gamma_1 \wedge \gamma_2]\!] \neq \emptyset\}$.

LEMMA 4.4. *Let $P_1, P_2 \in GP$ and $\gamma \in P_1 \wedge P_2$. There are unique elements $\gamma_1 \in P_1$, $\gamma_2 \in P_2$ such that $\gamma = \gamma_1 \wedge \gamma_2$.*

We can hence write the elements of $P_1 \wedge P_2$ as $\gamma_1 \wedge \gamma_2$ without ambiguity. We are ready to define operations $\oplus$, $\otimes$ and * on functions in $GP[K]$.

Definition 4.5. Let $f_1 : P_1 \to K, f_2 : P_2 \to K \in GP[K]$. Define functions $s', p' : P_1 \wedge P_2 \to K$ and $t' : P_1 \to K$ by $s'(\gamma_1 \wedge \gamma_2) = f_1(\gamma_1) \oplus f_2(\gamma_2)$, $p'(\gamma_1 \wedge \gamma_2) = f_1(\gamma_1) \otimes f_2(\gamma_2)$, and $t'(\gamma_1) = f_1(\gamma_1)^*$. Let $s, p, t \in GP[K]$ be the canonicalizations of s', p' and t', respectively, then we define $f_1 \oplus f_2 = s$, $f_1 \otimes f_2 = p$, and $f_1^* = t$.

Figure 3 shows algorithms to compute these operations in $GP[K]$. Note how these are similar to the Split procedure in Fig. 1.

Let $\mathbb{0}, \mathbb{1} : \{\mathbf{tt}\} \to K$ be the functions given by $\mathbb{0}(\mathbf{tt}) = 0$ and $\mathbb{1}(\mathbf{tt}) = 1$. Then $\mathbb{0}, \mathbb{1} \in GP[K]$.

LEMMA 4.6. *Let $f_1, f_2 \in GP[K]$ and $p \in px$. Then $[\![f_1 \oplus f_2]\!](p) = [\![f_1]\!](p) \oplus [\![f_2]\!](p)$, $[\![f_1 \otimes f_2]\!](p) = [\![f_1]\!](p) \otimes [\![f_2]\!](p)$, and $[\![f_1^*]\!](p) = [\![f]\!]_1(p)^*$.*

LEMMA 4.7. *Let $f_1, f_2 \in GP[K]$. Then $f_1 = f_2$ iff $[\![f_1]\!] = [\![f_2]\!]$.*

PROPOSITION 4.8. *The structure $(GP[K], \oplus, \otimes, \mathbb{0}, \mathbb{1})$ forms a *-continuous Kleene algebra.*

```
1:  function KSum(f_1 : P_1 → K, f_2 : P_2 → K): GP[K]
2:      var f', P'
3:      P' ← ∅
4:      for all γ_1 ∈ P_1 do
5:          for all γ_2 ∈ P_2 do
6:              if [[γ_1 ∧ γ_2]] ≠ ∅ then
7:                  P' ← P' ∪ {γ_1 ∧ γ_2}
8:                  f'(γ_1 ∧ γ_2) ← f_1(γ_1) ⊕ f_2(γ_2)
9:      return KCombine(f')

10: function KProd(f_1 : P_1 → K, f_2 : P_2 → K): GP[K]
11:     var f', P'
12:     P' ← ∅
13:     for all γ_1 ∈ P_1 do
14:         for all γ_2 ∈ P_2 do
15:             if [[γ_1 ∧ γ_2]] ≠ ∅ then
16:                 P' ← P' ∪ {γ_1 ∧ γ_2}
17:                 f'(γ_1 ∧ γ_2) ← f_1(γ_1) ⊗ f_2(γ_2)
18:     return KCombine(f')

19: function KStar(f : P → K): GP[K]
20:     var f'
21:     for all γ ∈ P do
22:         f'(γ) ← f(γ)*
23:     return KCombine(f')
```

Figure 3: Functions which compute $\oplus$, $\otimes$ and * in $GP[K]$.

LEMMA 4.9. *For $n \geq 1$, $M \in GP[K]^{n \times n}$, and $p \in px$, $[\![M^*]\!](p) = [\![M]\!](p)^*$.*

We are ready to give the central result of this paper, stating that for a given featured weighted automaton $\mathcal{F}$, computing $|\mathcal{F}|$ suffices to obtain all projected values.

THEOREM 4.10. *Let $\mathcal{F}$ be a featured weighted automaton over K and $p \in px$. Then $|\text{proj}_p(\mathcal{F})| = [\![|\mathcal{F}|]\!](p)$.*

PROOF. We have $[\![|\mathcal{F}|]\!](p) = [\![\alpha M^* \kappa]\!](p) = [\![\alpha]\!](p)[\![M]\!](p)^*[\![\kappa]\!](p)$ by Lemmas 4.6 and 4.9. Noting that the matrix representation of $\text{proj}_p(\mathcal{F})$ is $([\![\alpha]\!](p), [\![M]\!](p), k)$, the proof is finished. □

5 FEATURED ENERGY PROBLEMS

In this final section we apply the theoretical results of this paper to featured energy problems.

5.1 Energy Problems

The *energy semiring* [14] is the structure $\mathbb{E} = (\mathcal{E}, \vee, \circ, \bot, \top)$. Here $\mathcal{E}$ is the set of *energy functions*, which are partial functions $f : \mathbb{R}_{\geq 0} \cup \{\bot, \infty\} \to \mathbb{R}_{\geq 0} \cup \{\bot, \infty\}$ on extended real numbers ($f(x) = \bot$ meaning that f is undefined at x) with the property that

$$\text{for all } x \leq y : f(y) - f(x) \geq y - x . \tag{3}$$

These have been introduced in [14] as a general framework to handle formal energy problems as below. The operations in the semiring are (pointwise) maximum as $\oplus$ and function composition as $\otimes$,

and the neutral elements are the functions $\bot$, id given by $\bot(x) = \bot$ and $\mathrm{id}(x) = x$ for all $x \in \mathbb{R}_{\geq 0} \cup \{\bot, \infty\}$.

Definition 5.1. An *energy automaton* is a tuple (S, I, F, T) consisting of a finite set S of states, subsets $I, F \subseteq S$ of initial and accepting states, and a finite set $T \subseteq S \times \mathcal{E} \times S$ of transitions.

Hence the transition labels in energy automata are functions which proscribe how a real-valued variable evolves along a transition. An *energy problem* asks, then, whether some state is reachable when given a certain *initial energy*, or whether the automaton admits infinite accepting runs from some initial energy:

A *global state* of an energy automaton is a pair $q = (s, x)$ with $s \in S$ and $x \in \mathbb{R}_{\geq 0}$. A transition between global states is of the form $((s, x), f, (s', x'))$ such that $(s, f, s') \in T$ and $x' = f(x)$. A (finite or infinite) *run* of the automaton is a (finite or infinite) path in the graph of global states and transitions.

As the input to a decision problem must be in some way finitely representable, we will state them for subclasses $\mathcal{E}' \subseteq \mathcal{E}$ of *computable* energy functions (but note that we give no technical meaning to the term "computable" other that "finitely representable"); an $\mathcal{E}'$-automaton is an energy automaton (S, I, F, T) with $T \subseteq S \times \mathcal{E}' \times S$.

Problem 1 (Reachability). Given a subset $\mathcal{E}' \subseteq \mathcal{E}$ of computable functions, an $\mathcal{E}'$-automaton $\mathcal{S} = (S, I, F, T)$ and a computable initial energy $x_0 \in \mathbb{R}_{\geq 0}$: do there exist $s_0 \in I$ and a finite run of $\mathcal{S}$ from (s_0, x_0) which ends in a state in F?

Problem 2 (Büchi acceptance). Given a subset $\mathcal{E}' \subseteq \mathcal{E}$ of computable functions, an $\mathcal{E}'$-automaton $\mathcal{S} = (S, I, F, T)$ and a computable initial energy $x_0 \in \mathbb{R}_{\geq 0}$: do there exist $s_0 \in I$ and an infinite run of $\mathcal{S}$ from (s_0, x_0) which visits F infinitely often?

As customary, a run such as in the statements above is said to be *accepting*.

5.2 *-Continuous Kleene ω-Algebras

We need a few algebraic notions connected to infinite runs in weighted automata before we can continue. An *idempotent semiring-semimodule pair* [1,16] (K, V) consists of an idempotent semiring $K = (K, \oplus, \otimes, 0, 1)$ and a commutative idempotent monoid $V = (V, \oplus, 0)$ which is equipped with a left K-action $K \times V \to V$, $(x, v) \mapsto xv$, satisfying the following axioms for all $x, y \in K$ and $u, v \in V$:

$$(x \oplus y)v = xv \oplus yv \qquad x(u \oplus v) = xu \oplus xv$$
$$(xy)v = x(yv) \qquad 0 \otimes x = 0$$
$$x \otimes 0 = 0 \qquad 1 \otimes v = v$$

Also non-idempotent versions of these are in use, but we will only need the idempotent one here.

A *generalized *-continuous Kleene algebra* [12] is an idempotent semiring-semimodule pair (K, V) where K is a *-continuous Kleene algebra such that for all $x, y \in K$ and for all $v \in V$,

$$xy^*v = \bigoplus_{n \geq 0} xy^n v.$$

A *-continuous Kleene ω-algebra* [12] consists of a generalized *-continuous Kleene algebra (K, V) together with an *infinite product* operation $K^\omega \to V$ which maps every infinite sequence $x_0, x_1, \ldots$

in K to an element $\prod_n x_n$ of V. The infinite product is subject to the following conditions:

- For all $x_0, x_1, \ldots \in K$, $\prod_n x_n = x_0 \prod_n x_{n+1}$.
- Let $x_0, x_1, \ldots \in K$ and $0 = n_0 \leq n_1 \leq \cdots$ a sequence which increases without a bound. Let $y_k = x_{n_k} \cdots x_{n_{k+1}-1}$ for all $k \geq 0$. Then $\prod_n x_n = \prod_k y_k$.
- For all $x_0, x_1, \ldots, y, z \in K$, we have $\prod_n (x_n(y \oplus z)) = \bigoplus_{x_0', x_1', \ldots \in \{y, z\}} \prod_n x_n x_n'$.
- For all $x, y_0, y_1, \ldots \in K$, $\prod_n x^* y_n = \bigoplus_{k_0, k_1, \ldots \geq 0} \prod_n x^{k_n} y_n$.

For any idempotent semiring-semimodule pair (K, V) and $n \geq 1$, we can form the matrix semiring-semimodule pair $(K^{n \times n}, V^n)$ whose elements are $n \times n$-matrices of elements of K and n-dimensional (column) vectors of elements of V, with the action of $K^{n \times n}$ on V^n given by the usual matrix-vector product.

When (K, V) is a *-continuous Kleene ω-algebra, then $(K^{n \times n}, V^n)$ is a generalized *-continuous Kleene algebra [12]. By [12, Lemma 17], there is an ω-operation on $K^{n \times n}$ defined by

$$M_i^\omega = \bigoplus_{1 \leq k_1, k_2, \ldots \leq n} M_{i, k_1} M_{k_1, k_2} \cdots$$

for all $M \in K^{n \times n}$ and $1 \leq i \leq n$. Also, if $n \geq 2$ and $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, where a and d are square matrices of dimension less than n, then

$$M^\omega = \begin{bmatrix} (a \oplus bd^*c)^\omega \oplus (a \oplus bd^*c)^* bd^\omega \\ (d \oplus ca^*b)^\omega \oplus (d \oplus ca^*b)^* ca^\omega \end{bmatrix}.$$

We also need another matrix-ω-power below. Let $n \geq 2$, $k < n$ and $M \in K^{n \times n}$, and write $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ as above, with $a \in K^{k \times k}$ top left k-by-k part of M. We define

$$M^{\omega_k} = \begin{bmatrix} (a \oplus bd^*c)^\omega \\ d^*c(a \oplus bd^*c)^\omega \end{bmatrix}.$$

Let (K, V) be a *-continuous Kleene ω-algebra and $\mathcal{S} = (S, I, F, T)$ a K-weighted automaton. An *infinite path* in $\mathcal{S}$ is an infinite alternating sequence $\pi = (s_0, x_0, s_1, x_1, s_2, \ldots)$ of transitions (s_0, x_0, s_1), $(s_1, x_1, s_2), \ldots \in T$. The *weight* of π is the infinite product $w(\pi) = \prod_n x_n \in V$.

An infinite path $\pi = (s_0, x_0, s_1, x_1, \ldots)$ in $\mathcal{S}$ is said to be *Büchi accepting* if $s_0 \in I$ and the set $\{n \in \mathbb{N} \mid s_n \in F\}$ is infinite. The *Büchi value* $\|\mathcal{S}\|$ of $\mathcal{S}$ is defined to be the sum of the weights of all its Büchi accepting infinite paths:

$$\|\mathcal{S}\| = \bigoplus \{w(\pi) \mid \pi \text{ Büchi accepting infinite path in } \mathcal{S}\}$$

Let (α, M, k) be the matrix representation of $\mathcal{S}$. It can be shown [12] that

$$\|\mathcal{S}\| = \alpha M^{\omega_k}.$$

5.3 Featured Energy Problems

Recall that $\mathcal{E}$ denotes the set of energy functions: functions $f : \mathbb{R}_{\geq 0} \cup \{\bot, \infty\} \to \mathbb{R}_{\geq 0} \cup \{\bot, \infty\}$ with the property (3) that whenever $x \leq y$, then $f(y) - f(x) \geq y - x$; and that $\mathbb{E} = (\mathcal{E}, \vee, \circ, \bot, \top)$ is the semiring of energy functions.

LEMMA 5.2 ([13]). $\mathbb{E}$ *is a *-continuous Kleene algebra.*

Let $\mathbb{B} = \{\mathbf{ff}, \mathbf{tt}\}$ be the Boolean lattice. We say that a function $f : \mathbb{R}_{\geq 0} \cup \{\bot, \infty\} \to \mathbb{B}$ is ∞-*continuous* if $f = \bot$ or for all $X \subseteq \mathbb{R}_{\geq 0} \cup \{\bot, \infty\}$ with $\bigvee X = \infty$, $\bigvee f(X) = \mathbf{tt}$.

Let $\mathcal{V}$ be the set of ∞-continuous functions $f : \mathbb{R}_{\geq 0} \cup \{\bot, \infty\} \to \mathbb{B}$. With operation $\vee$ defined by $(f \vee g)(x) = f(x) \vee g(x)$ and unit $\bot$ given by $\bot(x) = \mathbf{ff}$ for all $x \in \mathbb{R}_{\geq 0} \cup \{\bot, \infty\}$, $\mathbb{V} = (\mathcal{V}, \vee, \bot)$ forms a commutative idempotent monoid. Then $(\mathbb{E}, \mathbb{V})$ is an idempotent semiring-semimodule pair.

Define an infinite product $\mathcal{E} \to \mathcal{V}$ as follows: Let $f_0, f_1, \dots \in \mathcal{E}$ be an infinite sequence and $x \in \mathbb{R}_{\geq 0} \cup \{\bot, \infty\}$. Let $x_0 = f_0(x)$ and, for each $k \geq 1$, $x_k = f_k(x_{k-1})$. Thus $x_0, x_1, \dots$ is the infinite sequence of values obtained by application of finite prefixes of the function sequence $f_0, f_1, \dots$. Then $(\prod_n f_n)(x) = \mathbf{ff}$ if there is an index k for which $x_k = \bot$ and $(\prod_n f_n)(x) = \mathbf{tt}$ otherwise.

It can be shown [13] that $\prod_n f_n$ is ∞-continuous for any infinite sequence $f_0, f_1, \dots \in \mathcal{E}$, hence this defines indeed a mapping $\mathcal{E}^\omega \to \mathcal{V}$.

LEMMA 5.3 ([13]). $(\mathbb{E}, \mathbb{V})$ *is a* **-continuous Kleene* ω*-algebra.*

Hence the energy problems stated at the end of Sect. 5.1 can be solved by computing reachability and Büchi values of energy automata:

PROPOSITION 5.4 ([13]). *Let* $\mathcal{S} = (S, I, F, T)$ *be an energy automaton and* $x_0 \in \mathbb{R}_{\geq 0}$.

- *There exist* $s_0 \in I$ *and a finite run of* $\mathcal{S}$ *from* (s_0, x_0) *which ends in a state in F iff* $|\mathcal{S}|(x_0) \neq \bot$.
- *There exist* $s_0 \in I$ *and an infinite run of* $\mathcal{S}$ *from* (s_0, x_0) *which visits F infinitely often iff* $\|\mathcal{S}\|(x_0) = \mathbf{tt}$.

We now define energy problems for featured automata. Recall that N denotes a set of features and $px \subseteq 2^N$ a set of products over N.

Definition 5.5. A *featured energy automaton* over px is a tuple (S, I, F, T) consisting of a finite set S of states, subsets $I, F \subseteq S$ of initial and accepting states, and a finite set $T \subseteq S \times GP[\mathcal{E}] \times S$ of transitions.

Hence transitions in featured energy automata are labeled with (injective) functions from guard partitions to energy functions.

LEMMA 5.6. *For* $f \in \mathcal{E}$, $f^\omega \in \mathcal{V}$ *is given by*

$$f^\omega(x) = \begin{cases} \mathbf{ff} & \text{if } x = \bot \text{ or } f(x) < x, \\ \mathbf{tt} & \text{otherwise}. \end{cases}$$

Definition 5.7. Let $f : P \to \mathcal{E} \in GP[\mathcal{E}]$ and define $w' : P \to \mathcal{V}$ by $w'(\gamma) = f(\gamma)^\omega$. Let $w \in GP[\mathcal{V}]$ be the canonicalization of w', then we define $f^\omega = w$.

LEMMA 5.8. *For* $n \geq 1$, $k < n$, $M \in GP[\mathcal{E}]^{n \times n}$, *and* $p \in px$, $[\![M^\omega]\!](p) = [\![M]\!](p)^\omega$ *and* $[\![M^{\omega k}]\!](p) = [\![M]\!](p)^{\omega k}$.

THEOREM 5.9. *Let* $\mathcal{F}$ *be a featured energy automaton and* $p \in px$. *Then* $\|\text{proj}_p(\mathcal{F})\| = [\![\|\mathcal{F}\|]\!](p)$.

PROOF. We have $[\![\|\mathcal{F}\|]\!](p) = [\![\alpha M^{\omega k}]\!](p) = [\![\alpha]\!](p)[\![M]\!](p)^{\omega k}$ by Lemmas 4.6 and 5.8. As the matrix representation of $\text{proj}_p(\mathcal{F})$ is $([\![\alpha]\!](p), [\![M]\!](p), k)$, the result follows. □

6 CONCLUSION

We have introduced featured (semiring-) weighted automata and shown that, essentially, verification of their properties can be reduced to checking properties of weighted automata. This is because, from a mathematical point of view, a featured weighted automaton over a semiring K is the same as a weighted automaton over the semiring of functions from products (sets of features) to K.

Representing functions from products to K as injective functions from partitions of the set of products to K, we have exposed algorithms which will compute featured weighted reachability in case K is a *-continuous Kleene algebra. It is easy to see that these extend to the non-idempotent case of K being a *Conway semiring*. The essence in our approach does not lie in these technical details, but in the fact that we pass from K to a semiring of functions into K; this typically preserves properties one is interested in.

We have also seen that energy properties are preserved when passing from the weighted to the featured weighted setting; generally, if (K, V) is a *-continuous Kleene ω-algebra, then the semiring-semimodule pair of functions from products to K and V, respectively, will also be such.

We are interested in extending the setting of this paper to other weighted structures beyond semirings, for example the valuation monoids of [11]. This will enable feature-based treatment of properties such as limit-average cost and will be useful for an extension to the timed setting of [7]. From a practical point of view, we have shown in [21] that efficient algorithms are available for the limit-average setting.

REFERENCES

[1] S. L. Bloom and Z. Ésik. *Iteration Theories: The Equational Logic of Iterative Processes.* EATCS monographs on theoretical computer science. Springer, 1993.
[2] P. Bouyer, U. Fahrenberg, K. G. Larsen, and N. Markey. Timed automata with observers under energy constraints. In *HSCC*. ACM, 2010.
[3] P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey, and J. Srba. Infinite runs in weighted timed automata with energy constraints. In *FORMATS*, vol. 5215 of *LNCS*. Springer, 2008.
[4] P. Bouyer, K. G. Larsen, and N. Markey. Lower-bound constrained runs in weighted timed automata. In *QEST*. IEEE Computer Society, 2012.
[5] K. Chatterjee and L. Doyen. Energy parity games. In *ICALP (2)*, vol. 6199 of *LNCS*. Springer, 2010.
[6] A. Classen, M. Cordy, P. Schobbens, P. Heymans, A. Legay, and J. Raskin. Featured transition systems: Foundations for verifying variability-intensive systems and their application to LTL model checking. *IEEE Trans. Software Eng.*, 39(8):1069–1089, 2013.
[7] M. Cordy, P. Schobbens, P. Heymans, and A. Legay. Behavioural modelling and verification of real-time software product lines. In *SPLC*. ACM, 2012.
[8] M. Cordy, P. Schobbens, P. Heymans, and A. Legay. Beyond boolean product-line model checking: dealing with feature attributes and multi-features. In *ICSE*. IEEE / ACM, 2013.
[9] A. Degorre, L. Doyen, R. Gentilini, J.-F. Raskin, and S. Torunczyk. Energy and mean-payoff games with imperfect information. In *CSL*, 2010.
[10] M. Droste, W. Kuich, and H. Vogler. *Handbook of Weighted Automata.* Springer, 2009.
[11] M. Droste and I. Meinecke. Weighted automata and regular expressions over valuation monoids. *Int. J. Found. Comput. Sci.*, 22(8):1829–1844, 2011.
[12] Z. Ésik, U. Fahrenberg, and A. Legay. *-continuous Kleene ω-algebras. In *DLT*, vol. 9168 of *LNCS*. Springer, 2015.
[13] Z. Ésik, U. Fahrenberg, and A. Legay. *-continuous Kleene ω-algebras for energy problems. In *FICS*, vol. 191 of *EPTCS*, 2015.
[14] Z. Ésik, U. Fahrenberg, A. Legay, and K. Quaas. Kleene algebras and semimodules for energy problems. In *ATVA*, vol. 8172 of *LNCS*. Springer, 2013.
[15] Z. Ésik and W. Kuich. Locally closed semirings. *Monatsh. Math.*, 137(1):21–29, 2002.
[16] Z. Ésik and W. Kuich. On iteration semiring-semimodule pairs. *Semigroup Forum*, 75:129–159, 2007.

[17] Z. Ésik and W. Kuich. Finite automata. In *Handbook of Weighted Automata* [10].
[18] U. Fahrenberg, L. Juhl, K. G. Larsen, and J. Srba. Energy games in multiweighted automata. In *ICTAC*, vol. 6916 of *LNCS*. Springer, 2011.
[19] D. Kozen. On Kleene algebras and closed semirings. In *MFCS*, vol. 452 of *LNCS*. Springer, 1990.
[20] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Inf. Comput.*, 110(2):366–390, 1994.
[21] R. Olaechea, U. Fahrenberg, J. M. Atlee, and A. Legay. Long-term average cost in featured transition systems. In *SPLC*. ACM, 2016.
[22] K. Quaas. On the interval-bound problem for weighted timed automata. In *LATA*, vol. 6638 of *LNCS*. Springer, 2011.
[23] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake. A classification and survey of analysis strategies for software product lines. *ACM Comput. Surv.*, 47(1):6:1–6:45, 2014.

APPENDIX: PROOFS

PROOF OF LEMMA 4.2. To see that $\tilde{f}$ is injective, let $\tilde{\gamma}_1, \tilde{\gamma}_2 \in \tilde{P}$ and assume $\tilde{f}(\tilde{\gamma}_1) = \tilde{f}(\tilde{\gamma}_2)$. Let $\Gamma_1, \Gamma_2 \in P'$ such that $\tilde{\gamma}_1 = \bigvee \Gamma_1$ and $\tilde{\gamma}_2 = \bigvee \Gamma_2$, then $f(\Gamma_1) = f(\Gamma_2)$ and hence $\Gamma_1 = \Gamma_2$, *i.e.,* $\tilde{\gamma}_1 = \tilde{\gamma}_2$.

For the second claim, let $\gamma \in P$, then $\gamma \in \Gamma$ for some $\Gamma \in P'$, hence $[\![\gamma]\!] \subseteq [\![\bigvee \Gamma]\!]$. To see uniqueness, let $\tilde{\gamma}_1, \tilde{\gamma}_2 \in \tilde{P}$ and assume $[\![\gamma]\!] \subseteq [\![\tilde{\gamma}_1]\!]$ and $[\![\gamma]\!] \subseteq [\![\tilde{\gamma}_2]\!]$. As $[\![\gamma]\!] \neq \emptyset$, this implies that $[\![\tilde{\gamma}_1]\!] \cap [\![\tilde{\gamma}_2]\!] \neq \emptyset$, hence $\tilde{\gamma}_1 = \tilde{\gamma}_2$. □

PROOF OF LEMMA 4.4. Existence of γ_1 and γ_2 is obvious by definition of $P_1 \wedge P_2$. For uniqueness, assume that there is $\gamma_1' \in P_1$ with $\gamma_1' \neq \gamma_1$ and $\gamma = \gamma_1' \wedge \gamma_2$. Then $\gamma = \gamma_1 \wedge \gamma_1' \wedge \gamma$, but as P_1 is a partition, $[\![\gamma_1 \wedge \gamma_1']\!] = [\![\gamma_1]\!] \cap [\![\gamma_1']\!] = \emptyset$, hence $[\![\gamma]\!] = \emptyset$, a contradiction. □

PROOF OF LEMMA 4.6. Let $f_1 : P_1 \rightarrow K$ and $f_2 : P_2 \rightarrow K$. Let $\gamma_1 \in P_1, \gamma_2 \in P_2$ be the unique feature guards for which $p \models \gamma_1$ and $p \models \gamma_2$, then $[\![f_1]\!](p) = f_1(\gamma_1)$ and $[\![f_2]\!](p) = f_2(\gamma_2)$.

We have $p \in [\![\gamma_1 \wedge \gamma_2]\!]$, hence $[\![\gamma_1 \wedge \gamma_2]\!] \neq \emptyset$, so that $\gamma_1 \wedge \gamma_2 \in P_1 \wedge P_2$. Using the notation of Def. 4.5, $s'(\gamma_1 \wedge \gamma_2) = f_1(\gamma_1) \oplus f_2(\gamma_2) = [\![f_1]\!](p) \oplus [\![f_2]\!](p)$. Write $s : P \rightarrow K$ and let $\tilde{\gamma} \in P$ be such that $[\![\gamma_1 \wedge \gamma_2]\!] \subseteq [\![\tilde{\gamma}]\!]$, *cf.* Lemma 4.2. Then $p \models \tilde{\gamma}$, hence $[\![f_1 \oplus f_2]\!](p) = (f_1 \oplus f_2)(\tilde{\gamma}) = s'(\gamma_1 \wedge \gamma_2)$. The proofs for $\otimes$ and * are similar. □

PROOF OF LEMMA 4.7. It is clear that $f_1 = f_2$ implies $[\![f_2]\!] = [\![f_2]\!]$. For the other direction, write $f_1 : P_1 \rightarrow K$ and $f_2 : P_2 \rightarrow K$. For each $p \in px$, let $\gamma_p = \bigwedge_{f \in p} f \wedge \bigwedge_{f \notin p} \neg f \in \mathbb{B}(N)$ denote its *characteristic feature guard*; note that $[\![\gamma_p]\!] = \{p\}$.

Let $P \in GP$ be the guard partition $P = \{\gamma_p \mid p \in px\}$, and define functions $f_1', f_2' : P \rightarrow K$ by $f_1'(\gamma_p) = [\![f_1]\!](p)$ and $f_2'(\gamma_p) = [\![f_2]\!](p)$. By definition, f_1 is the canonicalization of f_1' and f_2 the canonicalization of f_2'. By construction, $[\![f_1]\!] = [\![f_2]\!]$ implies $f_1' = f_2'$, hence $f_1 = f_2$. □

PROOF OF PROP. 4.8. We show that the set K^{px} of functions from px to K forms a *-continuous Kleene algebra; the theorem is then clear from Lemmas 4.6 and 4.7. For functions $\phi_1, \phi_2 : px \rightarrow K$, define $\phi_1 \oplus \phi_2$, $\phi_1 \otimes \phi_2$ and ϕ_1^* by $(\phi_1 \oplus \phi_2)(p) = \phi_1(p) \oplus \phi_2(p)$, $(\phi_1 \otimes \phi_2)(p) = \phi_1(p) \otimes \phi_2(p)$, and $\phi_1^*(p) = \phi_1(p)^*$. Let $0, 1 : px \rightarrow K$ be the functions $0(p) = 0$, $1(p) = 1$. Then $(K^{px}, \oplus, \otimes, 0, 1)$ forms an idempotent semiring.

We miss to show *-continuity. Let $p \in px$ and $\phi_1, \phi_2, \phi_3 : px \rightarrow K$, then

$$\begin{aligned}
(\phi_1 \phi_2^* \phi_3)(p) &= \phi_1(p)\phi_2^*(p)\phi_3(p) \\
&= \phi_1(p)\phi_2(p)^*\phi_3(p) \\
&= \phi_1(p)\big(\bigoplus_{n \geq 0} \phi_2(p)^n\big)\phi_3(p) \\
&= \bigoplus_{n \geq 0} \phi_1(p)\phi_2(p)^n\phi_3(p) \\
&= \bigoplus_{n \geq 0} \phi_1(p)\phi_2^n(p)\phi_3(p) \\
&= \big(\bigoplus_{n \geq 0} \phi_1\phi_2^n\phi_3\big)(p)
\end{aligned}$$

□

Proof of Lemma 4.9. As the formula for computing M^* involves only additions, multiplications and stars, this is clear by Lemma 4.6.

$\square$

Proof of Lemma 5.6. The claim is clear for $x = \bot$, so let $x \neq \bot$. If $f(x) \geq x$, then also $f^n(x) \geq x$ for all $n \geq 0$, hence $f^\omega(x) = \mathbf{tt}$ by definition.

If $f(x) < x$, then $f(x) \leq x - M$, with $M = x - f(x) > 0$. By (3), $f^n(x) \leq x - nM$ for all $n \geq 0$, hence there must be $k \geq 0$ for which $f^k(x) = \bot$, whence $f^\omega(x) = \mathbf{ff}$. $\square$

Proof of Lemma 5.8. The formulas for M^ω and $M^{\omega k}$ involve only additions, multiplications, stars, and ωs. Invoking Lemmas 4.6 and 4.9, we see that the proof will be finished once we show that for $f \in GP[\mathcal{E}]$, $[\![f^\omega]\!](p) = [\![f]\!](p)^\omega$.

Write $f : P \to \mathcal{E}$ and let $\gamma \in P$ be the unique feature guard for which $p \models \gamma$. Then $[\![f]\!](p) = f(\gamma)$. Using the notation of Def. 5.7, $w'(\gamma) = [\![f]\!](p)^\omega$. Write $w : P' \to \mathcal{V}$ and let $\tilde{\gamma} \in P'$ be such that $[\![\gamma]\!] \subseteq [\![\tilde{\gamma}]\!]$, *cf.* Lemma 4.2. Then $p \models \tilde{\gamma}$, hence $[\![f^\omega]\!](p) = f^\omega(\tilde{\gamma}) = w'(\gamma)$. $\square$