Noname manuscript No.
(will be inserted by the editor)

Quantitative Properties of Featured Automata

Uli Fahrenberg · Axel Legay

the date of receipt and acceptance should be inserted later

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.

For *quantitative* properties of transition systems, the model of *(semiring-) weighted automata* has proven useful [11]. This provides a uniform framework to treat problems such as minimum reachability, maximum flow, energy problems [14], 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 speci-

Uli Fahrenberg
École polytechnique, France

Axel Legay
Université catholique de Louvain, Belgium
Aalborg University, Denmark

fied resources). Starting with [3], formal modeling and analysis of such problems has attracted some attention [2, 4, 5, 10, 15, 20, 24].

Featured transition systems have applications in *software product lines*, where they are used as abstract representations of the behaviors of variability models [25]. 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 [9], 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. Section 4 is then devoted to the main conceptual result of this paper, namely that featured weighted automata are weighted automata (over a different semiring).

In Sect. 5, we apply this result to family-based analysis of featured weighted automata. The following Section 6 solves a technical problem related to infinite sums by introducing extra structure in the base semiring; we then show that this extra structure lifts to the featured semiring and how to use matrix calculations for family-based analysis.

In the last Sect. 7, we extend our results to develop an application to featured energy problems. This is based on our results in [14, 16, 17] 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.

This paper is based on [21], presented at the 5th IEEE/ACM International FME Workshop on Formal Methods in Software Engineering. Compared to this workshop paper, the algebraic setting presented here is more general, as we develop most of the paper for general semirings instead of *-continuous Kleene algebras. In particular, this entails that the current paper applies to non-idempotent settings.

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 a real-weighted automaton $\mathcal{S}$ is a finite alternating sequence $\pi = (s_0, x_0, s_1, x_1, \dots, x_k, s_{k+1})$ of transitions $(s_0, x_0, s_1), \dots, (s_k, x_k, s_{k+1}) \in T$. 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 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 $\gamma : T \to \mathbb{B}(N)$.

We refer to [23, Sects. 3 and 5] for examples of real-weighted featured automata.

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 $\mathsf{proj}_p(\mathcal{F}) = (S, I, F, T')$ with $T' = \{t \in T \mid p \models \gamma(t)\}$.

For each product $p \in px$, we could solve the shortest path problem in $\mathsf{proj}_p(\mathcal{F})$ by computing $|\mathsf{proj}_p(\mathcal{F})|$. 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 $p \in px$, $|\mathcal{F}|(p) = |\mathsf{proj}_p(\mathcal{F})|$.

As a symbolic representation of functions $f : px \to \mathbb{R}_{\geq 0}$, 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 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 $\textsc{Relax}(i,j,k)$ 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 29), 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 33 to 42, $D(i,j)$ 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 34), 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 45) and, in the affirmative case, to update the partition of $D(i,j)$ by joining the two parts (line 47f).

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 37), 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

```
 1: Input: real-weighted featured automaton
    F = (S, I, F, T, γ) with S = {s_1, ..., s_n}
 2: Output: function |F| ∈ GP[R_≥0]

 3: var D : {1,...,n} × {1,...,n} → GP[R_≥0]
 4: var P, f
 5: for i ← 1 to n do
 6:   for j ← 1 to n do
 7:     dom(D(i,j)) ← {tt}
 8:     D(i,j)(tt) ← ∞
 9:     for all (s_i, x, s_j) ∈ T do
10:       for all γ ∈ dom(D(i,j)) do
11:         if [γ ∧ γ(s_i, x, s_j)] ≠ ∅ and D(i,j)(γ) > x then
12:           Split(D(i,j), γ, γ(s_i, x, s_j), x)
13: for i ← 1 to n do
14:   for j ← 1 to n do
15:     for k ← 1 to n do
16:       Relax(i, j, k)
17: P ← {tt}; f(tt) ← ∞
18: for all s_i ∈ I do
19:   for all s_j ∈ F do
20:     for all γ_1 ∈ P do
21:       for all γ_2 ∈ dom(D(i,j)) do
22:         if [γ_1 ∧ γ_2] ≠ ∅ and f(γ_1) > D(i,j)(γ_2) then
23:           Split(f, γ_1, γ_2, D(i,j)(γ_2))
24: return f

25: procedure Relax(i, j, k)
26:   for all γ_1 ∈ dom(D(i,j)) do
27:     for all γ_2 ∈ dom(D(i,k)) do
28:       for all γ_3 ∈ dom(D(k,j)) do
29:         if [γ_1 ∧ γ_2 ∧ γ_3] ≠ ∅ then
30:           if D(i,j)(γ_1) > D(i,k)(γ_2) + D(k,j)(γ_3) then
31:             Split(D(i,j), γ_1, γ_2 ∧ γ_3,
32:                   D(i,k)(γ_2) + D(k,j)(γ_3))

33: procedure Split(f : P → R_≥0, γ_1, γ_2 ∈ B(N), x ∈ R_≥0)
34:   if [γ_1] = [γ_1 ∧ γ_2] then
35:     f(γ_1) ← x
36:     Combine(f, γ_1)
37:   else
38:     y ← f(γ_1)
39:     P ← P \ {γ_1} ∪ {γ_1 ∧ γ_2, γ_1 ∧ ¬γ_2}
40:     f(γ_1 ∧ ¬γ_2) ← y
41:     f(γ_1 ∧ γ_2) ← x
42:     Combine(f, γ_1 ∧ γ_2)

43: procedure Combine(f : P → R_≥0, γ ∈ B(N))
44:   x ← f(γ)
45:   for all δ ∈ P \ {γ} do
46:     if f(δ) = f(γ) then
47:       P ← P \ {δ, γ} ∪ {δ ∨ γ}
48:       f(δ ∨ γ) ← x
49:       break
```

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

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.

Like the standard Floyd-Warshall algorithm, our algorithm solves minimum reachability in a number of steps which is cubic in the size of the input automaton. However, each step may lead to a feature expression being split in two, increasing the size of the representation of the featured automaton. Hence a theoretical upper bound for the running time is exponential.

A variant of the algorithm in Fig. 1 has been implemented in [23], as part of an effort to compute minimum limit-average cost in real-weighted featured automata. Several experiments in [23] show that, despite its theoretical exponential complexity, 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 *semiring*. This allows us to develop an abstract setting for 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* [11] $K = (K, \oplus, \otimes, 0, 1)$ 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* [11] 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

$$\pi = (s_0, x_0, s_1, \dots, x_k, s_{k+1})$$

of transitions $(s_0, x_0, s_1), \dots, (s_k, x_k, s_{k+1}) \in T$. 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. 6 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 $\pi = (s_0, x_0, s_1, \dots, x_k, s_{k+1})$, 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.

As a matter of syntactic difference only, we label the transitions of featured weighted automata with functions from guard partitions to weights instead of annotating transitions with weights and feature guards, as we did in Sect. 2.

Let K be an arbitrary semiring.

Definition 3 We 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 4 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.

We use *injective* functions from guard partitions to K for conciseness: if a function $f : P \to K$ from a partition $P \in GP$ is not injective, then it takes the same value on two different feature guards, hence there is a coarser partition on which an equivalent function may be defined; see Def. 7 below.

Example 1 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 5 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 $[\![f]\!] : px \to K$.

Lemma 1 *Let $f_1, f_2 \in GP[K]$. Then $f_1 = f_2$ iff $[\![f_1]\!] = [\![f_2]\!]$.*

Proof It is clear that $f_1 = f_2$ implies $[\![f_2]\!] = [\![f_2]\!]$. For the other direction, write $f_1 : P_1 \to K$ and $f_2 : P_2 \to K$. 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 $f_1', f_2' : P \to K$ 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 $f_1 = f_2$. □

```
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
```

Fig. 2 Function which computes canonicalization.

Definition 6 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 $\mathsf{proj}_p(\mathcal{F}) = (S, I, F, T')$, 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, $|\mathsf{proj}_p(\mathcal{F})|$, 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 $|\mathsf{proj}_p(\mathcal{F})|$ can be computed for all $p \in px$ at once.

4 Featured Weighted Automata as Weighted Automata

In this section we define operations of addition and multiplication on functions in $GP[K]$ which, as we shall see, turn $GP[K]$ into a semiring itself.

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 7 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 $\tilde{\gamma} = \bigvee \Gamma$, 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 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 $[\![\gamma]\!] \subseteq [\![\tilde{\gamma}]\!]$.*

Proof 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 $\tilde{\gamma}_1 = \tilde{\gamma}_2$. □

Definition 8 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 3 *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 $\gamma = \gamma_1 \wedge \gamma_2$.*

Proof 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. □

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 addition and multiplication on functions in $GP[K]$.

Definition 9 Let $f_1 : P_1 \to K, f_2 : P_2 \to K \in GP[K]$. Define functions $s', p' : P_1 \wedge P_2 \to K$ by $s'(\gamma_1 \wedge \gamma_2) = f_1(\gamma_1) \oplus f_2(\gamma_2)$ and $p'(\gamma_1 \wedge \gamma_2) = f_1(\gamma_1) \otimes f_2(\gamma_2)$. Let $s : P_s \to K$ and $p : P_p \to K$ be the canonicalizations of s' and p', then we define $f_1 \oplus f_2 = s$ and $f_1 \otimes f_2 = p$.

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

Lemma 4 *Let $f_1, f_2 \in GP[K]$ and $p \in px$. Then $[\![f_1 \oplus f_2]\!](p) = [\![f_1]\!](p) \oplus [\![f_2]\!](p)$ and $[\![f_1 \otimes f_2]\!](p) = [\![f_1]\!](p) \otimes [\![f_2]\!](p)$.*

Proof Let $f_1 : P_1 \to K$ and $f_2 : P_2 \to K$. 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. 9, $s'(\gamma_1 \wedge \gamma_2) = f_1(\gamma_1) \oplus f_2(\gamma_2) = [\![f_1]\!](p) \oplus [\![f_2]\!](p)$. Write $s : P \to K$ and let $\tilde{\gamma} \in P$ be such that $[\![\gamma_1 \wedge \gamma_2]\!] \subseteq [\![\tilde{\gamma}]\!]$, *cf.* Lemma 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 proof for $\otimes$ is similar. □

```
1: function KSum(f1 : P1 → K, f2 : P2 → K): GP[K]
2:   var f', P'
3:   P' ← ∅
4:   for all γ1 ∈ P1 do
5:     for all γ2 ∈ P2 do
6:       if [[γ1 ∧ γ2]] ≠ ∅ then
7:         P' ← P' ∪ {γ1 ∧ γ2}
8:         f'(γ1 ∧ γ2) ← f1(γ1) ⊕ f2(γ2)
9:   return KCombine(f')

10: function KProd(f1 : P1 → K, f2 : P2 → K): GP[K]
11:   var f', P'
12:   P' ← ∅
13:   for all γ1 ∈ P1 do
14:     for all γ2 ∈ P2 do
15:       if [[γ1 ∧ γ2]] ≠ ∅ then
16:         P' ← P' ∪ {γ1 ∧ γ2}
17:         f'(γ1 ∧ γ2) ← f1(γ1) ⊗ f2(γ2)
18:   return KCombine(f')
```

Fig. 3 Functions which compute $\oplus$ and $\otimes$ in $GP[K]$.

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

Proposition 1 *The structure $(GP[K], \oplus, \otimes, \mathbf{0}, \mathbf{1})$ forms a semiring.*

Proof We show that the set K^{px} of functions from px to K forms a semiring; the theorem is then clear from Lemmas 1 and 4. For functions $\phi_1, \phi_2 : px \to K$, define $\phi_1 \oplus \phi_2$ and $\phi_1 \otimes \phi_2$ by $(\phi_1 \oplus \phi_2)(p) = \phi_1(p) \oplus \phi_2(p)$ and $(\phi_1 \otimes \phi_2)(p) = \phi_1(p) \otimes \phi_2(p)$. Let $0, 1 : px \to K$ be the functions $0(p) = 0$, $1(p) = 1$. Then $(K^{px}, \oplus, \otimes, 0, 1)$ forms a semiring. □

Hence a featured K-weighted automaton is the same as a $GP[K]$-weighted automaton.

5 Family-Based Analysis of Featured Weighted Automata

We have shown in the preceding section that featured weighted automata over K are weighted automata over $GP[K]$. Here we see how this is useful for family-based analysis of featured weighted automata.

Theorem 1 *Let S be a featured weighted automaton over K and px, let $p \in px$, and assume that only finite sums are used in the computation of $|S|$. Then $|\mathsf{proj}_p(S)| = [\![|S|]\!](p)$.*

Proof This follows almost directly from Lemma 4. By definition, $|S|$ is the sum of the weights of all accepting paths in S. For any accepting path π in S, $w(\pi)$ is a function in $GP[K]$, and $[\![w(\pi)]\!](p) = w(\mathsf{proj}_p(\pi))$ by Lemma 4. Here $\mathsf{proj}_p(\pi)$ denotes the projection of π to p, which is an accepting path in $\mathsf{proj}_p(S)$. By assumption, the reachability value $|S|$ is a finite sum, hence the claim follows by Lemma 4. □

Example 2 Let $K = \mathbb{T} = (\mathbb{R}_{\geq 0} \cup \{\infty\}, \wedge, +, \infty, 0)$ be the tropical semiring. We have seen that $\mathbb{T}$-weighted automata model the minimum reachability problem. That is, if S is a $\mathbb{T}$-weighted automaton, then $|S|$ is the minimum weight of all paths from an initial to an accepting state.

Let now S be a featured $\mathbb{T}$-weighted automaton. As all weights are non-negative, we see that the weight of any path which contains a loop can be made smaller by removing the loop. Hence only simple paths need be taken into account, that is, all paths in the sum $|S| = \bigoplus\{w(\pi) \mid \pi \text{ accepting path in } S\}$ can be assumed to be simple paths, of which there are finitely many. Thus Thm. 1 applies.

We have thus shown that if S is a featured $\mathbb{T}$-weighted automaton, then $[\![|S|]\!](p)$, for every product p, is the minimum weight of all paths from an initial to an accepting state in $\mathsf{proj}_p(S)$.

Example 3 Similarly to the above, if $K = \mathbb{F} = (\mathbb{R}_{\geq 0} \cup \{\infty\}, \vee, \wedge, 0, \infty)$ is the fuzzy semiring, that is, we are solving the maximum-flow problem, then the capacity of a path cannot increase by taking a loop. Hence also here we only need take into account simple paths, implying that Thm. 1 applies. Modified versions of Dijkstra's or the Floyd-Warshall algorithm can be used to compute reachability values of featured weighted automata over $\mathbb{F}$.

6 Featured Weighted Automata over Conway Semirings

One way of dealing with the problem that to compute the reachability value

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

of a K-weighted automaton S, one may have to compute an infinite sum, is to introduce extra structure in the semiring K. This is what we shall do in this section, afterwards showing that also with this extra structure, reachability values of featured weighted automata can be computed in a family-based manner.

Recall that a *Conway semiring* [7,11] is a semiring $K = (K, \oplus, \otimes, 0, 1)$ together with a unary *star* operation $^* : K \to K$ which satisfies the identities

$$(x \oplus y)^* = (x^* y)^* x^* \qquad (xy)^* = 1 \oplus x(yx)^* y$$

for all $x, y \in K$.

It is known [11] that if K is a Conway semiring, then for each $n \geq 1$, so is the *matrix semiring* $K^{n\times n}$ of all $n \times n$-matrices over K with the usual sum and product operations and the star operation defined by induction on n, so that if $n > 1$ and $M = \left[\begin{smallmatrix} x & y \\ z & u \end{smallmatrix}\right]$, where x and u are square matrices of dimension less than n, then

$$M^* = \begin{bmatrix} (x \oplus yu^*z)^* & (x \oplus yu^*z)^* yu^* \\ (u \oplus zx^*y)^* zx^* & (u \oplus zx^*y)^* \end{bmatrix}.$$

The above inductive definition does not depend on how M is split into submatrices.

The *matrix representation* [11] of a K-weighted automaton $S = (S, S^0, F, T)$, with $n = \#S$ the number of states, is given by the triple (α, M, k), 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 $i \in S_0$ and $M_{i,j} = \bigoplus\{x \mid (i, x, j) \in T\}$.

It can be shown [19] that the reachability value of S can be computed as

$$|S| = \alpha M^* \kappa\,,$$

where $\kappa \in \{0,1\}^n$ is the vector given by $\kappa_i = 1$ for $i \leq k$ and $\kappa_i = 0$ for $i > k$.

Let K be a Conway semiring. We introduce a star operation in $GP[K]$:

Definition 10 Let $f : P \to K \in GP[K]$. Let $s : P \to K$ be the function defined by $s(\gamma) = f(\gamma)^*$ for all $\gamma \in P$, then we define f^* to be the canonicalization of s.

Lemma 5 *Let $f \in GP[K]$ and $p \in px$. Then $[\![f^*]\!](p) = [\![f]\!](p)^*$.*

Proof Similar to the proof of Lemma 4. □

Proposition 2 *The structure $(GP[K], \oplus, \otimes, ^*, \mathbf{0}, \mathbf{1})$ forms a Conway semiring.*

Proof Similarly to the proof of Proposition 1, this follows from Lemmas 1 and 5 and the fact that the function semiring K^{px} is a Conway semiring. □

Lemma 6 *For $n \geq 1$, $M \in GP[K]^{n\times n}$, and $p \in px$, $[\![M^*]\!](p) = [\![M]\!](p)^*$.*

Proof As the formula for computing M^* involves only additions, multiplications and stars, this is clear by Lemmas 4 and 5. □

Theorem 2 *Let S be a featured weighted automaton over K and px, and let $p \in px$. Then $|\mathsf{proj}_p(S)| = [\![|S|]\!](p)$.*

Proof This follows from Lemma 6 and the formula $|S| = \alpha M^* \kappa$. □

Example 4 It can be shown that $\mathbb{B}$, $\mathbb{T}$ and $\mathbb{F}$ are all Conway semirings, for the trivial reason that the star operation is given by $x^* = 1$, the $\otimes$-neutral element, for all x. Intuitively, this is equivalent to the argument, which we gave in Examples 2 and 3, that loops can be disregarded.

7 Featured Energy Problems

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

7.1 Energy Problems

The *energy semiring* [15–17] 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

$$\text{for all } x \leq y : f(y) - f(x) \geq y - x\,. \tag{1}$$

These have been introduced in [15] 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 $\mathsf{id}(x) = x$ for all $x \in \mathbb{R}_{\geq 0} \cup \{\bot, \infty\}$.

Definition 11 An *energy automaton* (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 $T \subseteq S \times \mathcal{E} \times S$ 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.

7.2 *-Continuous Kleene ω-Algebras

We need a few algebraic notions connected to infinite runs in weighted automata before we can continue. Recall that a semiring $K = (K, \oplus, \otimes, 0, 1)$ is *idempotent* [11] if $x \oplus x = x$ for every $x \in K$.

A **-continuous Kleene algebra* [22] 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

$$x\Big(\bigoplus_{n \geq 0} y^n\Big)z = \bigoplus_{n \geq 0} xy^n z \tag{2}$$

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 (2)) can be computed locally (left-hand side of (2)).

In any *-continuous Kleene algebra K one can define a unary star operation $^* : K \to K$ by $x^* = \bigoplus_{n \geq 0} x^n$. This turns K into a Conway semiring [19], hence the results of the preceding sections of this paper apply.

An *idempotent semiring-semimodule pair* [1,18] (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 $u, v \in V$:

$$\begin{aligned} (x \oplus y)v &= xv \oplus yv & x(u \oplus v) &= xu \oplus xv \\ (xy)v &= x(yv) & 0 \otimes x &= 0 \\ x \otimes 0 &= 0 & 1 \otimes v &= v \end{aligned}$$

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

A *generalized *-continuous Kleene algebra* [13,16] 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* [13,16] consists of a generalized *-continuous Kleene algebra (K, V) together with an *infinite product* operation $K^\omega \to V$ which maps every infinite sequence $x_0, x_1, \dots$ in K to an element $\prod_n x_n$ of V. The infinite product is subject to the following conditions:

- For all $x_0, x_1, \dots \in K$, $\prod_n x_n = x_0 \prod_n x_{n+1}$.
- Let $x_0, x_1, \dots \in K$ 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 $x_0, x_1, \dots, y, z \in K$, we have $\prod_n (x_n(y \oplus z)) = \bigoplus_{x'_0, x'_1, \dots \in \{y,z\}} \prod_n x_n x'_n$.
- For all $x, y_0, y_1, \dots \in K$,

$$\prod_n x^* y_n = \bigoplus_{k_0, k_1, \dots \geq 0} \prod_n x^{k_n} y_n\,.$$

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 [13,16]. There is an ω-operation on $K^{n\times n}$ defined by

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

for all $M \in K^{n\times n}$ and $1 \leq i \leq n$. Also, if $n \geq 2$ and $M = \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, 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 $M = \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$ 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 $\pi = (s_0, x_0, s_1, x_1, s_2, \dots)$ of transitions (s_0, x_0, s_1), $(s_1, x_1, s_2), \dots \in T$. The *weight* of π is the infinite product $w(\pi) = \prod_n x_n \in V$.

An infinite path $\pi = (s_0, x_0, s_1, x_1, \dots)$ 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 [13,17] that

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

7.3 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 (1) 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 7 ([14, 17]) *$\mathbb{E}$ 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 [14,17] 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 8 ([14, 17]) *$(\mathbb{E}, \mathbb{V})$ is a *-continuous Kleene ω-algebra.*

Hence the energy problems stated at the end of Sect. 7.1 can be solved by computing reachability and Büchi values of energy automata:

Proposition 3 ([14, 17]) *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 12 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 9 *For $f \in \mathcal{E}$, $f^\omega \in \mathcal{V}$ is given by*

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

Proof 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 (1), $f^n(x) \leq x - nM$ for all $n \geq 0$, hence there must be $k \geq 0$ for which $f^k(x) = \bot$, whence $f^\omega(x) = \mathbf{ff}$. □

Definition 13 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 10 *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 $[\![M^{\omega_k}]\!](p) = [\![M]\!](p)^{\omega_k}$.*

Proof The formulas for M^ω and M^{ω_k} involve only additions, multiplications, stars, and ωs. Invoking Lemmas 4 and 6, 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. 13, $w'(\gamma) = [\![f]\!](p)^\omega$. Write $w : P' \to \mathcal{V}$ and let $\tilde{\gamma} \in P'$ be such that $[\![\gamma]\!] \subseteq [\![\tilde{\gamma}]\!]$, *cf.* Lemma 2. Then $p \models \tilde{\gamma}$, hence $[\![f^\omega]\!](p) = f^\omega(\tilde{\gamma}) = w'(\gamma)$. □

Theorem 3 *For $\mathcal{F}$ a featured energy automaton and $p \in px$, $\|\mathsf{proj}_p(\mathcal{F})\| = [\![\|\mathcal{F}\|]\!](p)$.*

Proof We have

$$[\![\|\mathcal{F}\|]\!](p) = [\![\alpha M^{\omega_k}]\!](p) = [\![\alpha]\!](p)[\![M]\!](p)^{\omega_k}$$

by Lemmas 4 and 10. As the matrix representation of $\mathsf{proj}_p(\mathcal{F})$ is $([\![\alpha]\!](p), [\![M]\!](p), k)$, the result follows.

We have thus shown that featured energy problems can be solved in a family-based manner, for all sets of features at once, by computing M^ω.

8 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 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 [12]. 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 [8]. From a practical point of view, we have shown in [23] that efficient algorithms are available for the limit-average setting.

References

1. Stephen L. Bloom and Zoltán Ésik. *Iteration Theories: The Equational Logic of Iterative Processes*. EATCS monographs on theoretical computer science. Springer, 1993.
2. Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, and Nicolas Markey. Timed automata with observers under energy constraints. In Karl Henrik Johansson and Wang Yi, editors, *HSCC*, pages 61–70. ACM, 2010.

3. Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, and Jiří Srba. Infinite runs in weighted timed automata with energy constraints. In Franck Cassez and Claude Jard, editors, *FORMATS*, volume 5215 of *LNCS*, pages 33–47. Springer, 2008.
4. Patricia Bouyer, Kim G. Larsen, and Nicolas Markey. Lower-bound constrained runs in weighted timed automata. In *QEST*, pages 128–137. IEEE Computer Society, 2012.
5. Krishnendu Chatterjee and Laurent Doyen. Energy parity games. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP (2)*, volume 6199 of *LNCS*, pages 599–610. Springer, 2010.
6. Andreas Classen, Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, Axel Legay, and Jean-François 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. John H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.
8. Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, and Axel Legay. Behavioural modelling and verification of real-time software product lines. In Eduardo Santana de Almeida, Christa Schwanninger, and David Benavides, editors, *SPLC*, pages 66–75. ACM, 2012.
9. Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, and Axel Legay. Beyond boolean product-line model checking: dealing with feature attributes and multi-features. In David Notkin, Betty H. C. Cheng, and Klaus Pohl, editors, *ICSE*, pages 472–481. IEEE / ACM, 2013.
10. Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Torunczyk. Energy and mean-payoff games with imperfect information. In *CSL*, pages 260–274, 2010.
11. Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer, 2009.
12. Manfred Droste and Ingmar Meinecke. Weighted automata and regular expressions over valuation monoids. *Int. J. Found. Comput. Sci.*, 22(8):1829–1844, 2011.
13. Zoltán Ésik, Uli Fahrenberg, and Axel Legay. *-continuous Kleene ω-algebras. In Igor Potapov, editor, *DLT*, volume 9168 of *LNCS*, pages 240–251. Springer, 2015.
14. Zoltán Ésik, Uli Fahrenberg, and Axel Legay. *-continuous Kleene ω-algebras for energy problems. In Ralph Matthes and Matteo Mio, editors, *FICS*, volume 191 of *EPTCS*, pages 48–59, 2015.
15. Zoltán Ésik, Uli Fahrenberg, Axel Legay, and Karin Quaas. Kleene algebras and semimodules for energy problems. In Dang Van Hung and Mizuhito Ogawa, editors, *ATVA*, volume 8172 of *LNCS*, pages 102–117. Springer, 2013.
16. Zoltán Ésik, Uli Fahrenberg, Axel Legay, and Karin Quaas. An algebraic approach to energy problems I: *-continuous Kleene ω-algebras. *Acta Cybern.*, 23(1):203–228, 2017.
17. Zoltán Ésik, Uli Fahrenberg, Axel Legay, and Karin Quaas. An algebraic approach to energy problems II: The algebra of energy functions. *Acta Cybern.*, 23(1):229–268, 2017.
18. Zoltán Ésik and Werner Kuich. On iteration semiring-semimodule pairs. *Semigroup Forum*, 75:129–159, 2007.
19. Zoltán Ésik and Werner Kuich. Finite automata. In *Handbook of Weighted Automata* [11], pages 69–104.
20. Uli Fahrenberg, Line Juhl, Kim G. Larsen, and Jiří Srba. Energy games in multiweighted automata. In *ICTAC*, volume 6916 of *LNCS*, pages 95–115. Springer, 2011.
21. Uli Fahrenberg and Axel Legay. Featured weighted automata. In *FormaliSE@ICSE*, pages 51–57. IEEE, 2017.
22. Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Inf. Comput.*, 110(2):366–390, 1994.
23. Rafael Olaechea, Uli Fahrenberg, Joanne M. Atlee, and Axel Legay. Long-term average cost in featured transition systems. In Hong Mei, editor, *SPLC*, pages 109–118. ACM, 2016.
24. Karin Quaas. On the interval-bound problem for weighted timed automata. In Adrian Horia Dediu, Shunsuke Inenaga, and Carlos Martín-Vide, editors, *LATA*, volume 6638 of *LNCS*, pages 452–464. Springer, 2011.
25. Thomas Thüm, Sven Apel, Christian Kästner, Ina Schaefer, and Gunter Saake. A classification and survey of analysis strategies for software product lines. *ACM Comput. Surv.*, 47(1):6:1–6:45, 2014.