Weighted GKAT: Completeness and Complexity

Spencer Van Koevering
Cornell University, United States of America

Wojciech Różowski
University College London, United Kingdom

Alexandra Silva
Cornell University, United States of America

Abstract

We propose Weighted Guarded Kleene Algebra with Tests (wGKAT), an uninterpreted weighted programming language equipped with branching, conditionals, and loops. We provide an operational semantics for wGKAT using a variant of weighted automata and introduce a sound and complete axiomatization. We also provide a polynomial time decision procedure for bisimulation equivalence.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases Weighted Programming, Automata, Axiomatization, Decision Procedure

Funding *Spencer Van Koevering*: Partially supported by a Cornell Bowers CIS Deans Excellence Fellowship.
Wojciech Różowski: Partially supported by ERC grant Autoprobe (grant agreement 101002697).
Alexandra Silva: Partially supported by ONR grant N68335-22-C-0411, ERC grant 101002697, and a Royal Society Wolfson fellowship.

1 Introduction

Weighted programming is a recently introduced paradigm [2] in which computational branches of programs are associated with quantities. For example, the program $e \,{}_{\frac{2}{5}}\!\oplus_{\frac{3}{5}}\, f$ does e with weight $\frac{2}{5}$ and f with weight $\frac{3}{5}$. Semantically, each branch is executed, its results scaled, and then results of different branches added. If we were to interpret weights as probabilities and use addition of real numbers, then the above example would be a classic randomized program: a biased coin flip. If we interpret the weights as costs rather than a probability, and addition as minimum rather than traditional addition, then $e \,{}_{\frac{2}{5}}\!\oplus_{\frac{3}{5}}\, f$ would model selecting the cheapest resulting branch and be well suited for optimization problems.

Weighted programs can model problems from a variety of domains, including optimization, model checking, and combinatorics [2]. However, reasoning about their behavior is difficult as different interpretations of the weights lead to different properties of the programs and their semantics, impacting, for instance, whether the equivalence of weighted programs is decidable. In this paper, we propose an algebraic approach to reason about equivalence of weighted programs and devise a decision procedure which applies to a large class of weights.

The starting point of our development is Guarded Kleene Algebra with Tests (GKAT) [30], an algebraic framework for reasoning about equivalence of uninterpreted imperative programs. Its equational axiomatization offers a simple framework for deductive reasoning, and it also has a decision procedure. We propose and axiomatize a GKAT-inspired weighted programming language, wGKAT, which enables reasoning about the equivalence of deterministic, uninterpreted programs with *weighted branching*. We design this language with the goals of offering an equational axiomatization and a polynomial-time decision procedure for the equivalence of expressions, both derived uniformly for a broad class of weights.

One of the primary challenges we address is identifying a suitable class of semirings which is sufficient to prove that bisimulation is a complete proof technique for behavioural

$$
\begin{array}{rrlr}
e, f \in \mathbf{Exp} \coloneqq & & \mathbf{p} \in \mathbf{Act} & \text{do } \mathbf{p} \\
& \mid & b \in \mathbf{BExp} & \text{assert } \mathbf{b} \\
& \mid & e +_b f & \text{if } \mathbf{b} \text{ do } e \text{ else do } f \\
& \mid & e; f & \text{do } e \text{ then do } f \\
& \mid & e^{(b)} & \text{while } \mathbf{b} \text{ do } e \\
& \mid & v \in \mathbf{Out} & \text{return } \mathbf{v} \\
& \mid & e\,{}_r{\oplus}_s\, f & \text{do } e \text{ with weight } \mathbf{r} \text{ and } f \text{ with weight } \mathbf{s}
\end{array}
$$

Figure 1 Syntax of wGKAT

equivalence and, moreover, to define a operational semantics of loops. Using this class of semirings we were able to generalize traditional arguments for soundness and completeness from process algebra to our setting [28]. Another challenge is the decidability of equivalence for weighted programs, which hinges on properties of the semiring as well. We leverage existing work on monoid labeled transition systems to design the operational semantics and establish that equivalence up to bisimilarity is decidable for wGKAT.

Our work was developed concurrently with another weighted variant of Kleene Algebra with Tests, called Kleene Algebra with Weights and Tests (KAWT) [29]. KAWT lacks a decision procedure and admits only a restrictive class of weights. The key difference in our work that enables us to significantly enlarge the class of weights we can consider is that KAWT extends KAT directly whereas our starting point is GKAT, the deterministic fragment of KAT. This is what allows us to offer a decision procedure and a sound and complete axiomatization: by carefully selecting the class of weights we retain the general road map to soundness and completeness of GKAT as well as the efficient decision procedure.

In a nutshell, our contributions are as follows.

1. We propose a weighted version of GKAT and equip it with an operational semantics using a variant of weighted automata (Sections 2 and 3).
2. We identify a class of semirings for which the semantics of weighted loops can be computed when considering bisimulation equivalence (Section 3.2).
3. We axiomatize bisimulation equivalence for wGKAT and provide a proof soundness (Section 4) and a proof of completeness (Section 5).
4. We design a decision procedure for bisimilarity which runs in $O(n^3 \log^2 n)$ time if the number of tests is fixed, where n is the size of the programs considered (Section 6).

We conclude and discuss directions for future work in Section 8.

2 Syntax

In this section, we introduce the syntax of our language and examples on how to use it to model simple quantitative programs. The syntax of wGKAT (Figure 1) is two-sorted consisting of the set **Exp** of expressions containing a sort **BExp** of Boolean expressions also called tests. Intuitively, **Exp** represents the overall syntax of weighted programs, while **BExp** is used to specify assertions, as well as conditional expressions appearing in the scope of `if-then-else` and `while` constructs. Given a finite set T of *primitive tests*, we define **BExp**, the grammar of tests, to be given by the following:

$$b, c \in \mathbf{BExp} \coloneqq \mathbb{0} \mid \mathbb{1} \mid t \in T \mid \bar{b} \mid b + c \mid bc$$

In the above, $\mathbb{0}$ and $\mathbb{1}$ respectively represent false and true, $\bar{\cdot}$ denotes negation, $+$ represents disjunction and juxtaposition is conjunction. We will write $\equiv_{\mathsf{BA}}$ to denote Boolean equivalence in **BExp**. Logical entailment defines a preorder given by $b \leq_{\mathsf{BA}} c \iff b + c \equiv_{\mathsf{BA}} c$. The quotient of **BExp** by the relation $\equiv_{\mathsf{BA}}$, denoted by $\mathbf{BExp}/\equiv_{\mathsf{BA}}$ is in one-to-one correspondence with the Boolean Algebra freely generated by the set T. The entailment $[b]_{\equiv_{\mathsf{BA}}} \leq [c]_{\equiv_{\mathsf{BA}}} \iff b \leq_{\mathsf{BA}} c$ defines a partial order on $\mathbf{BExp}/\equiv_{\mathsf{BA}}$. The minimal nonzero elements in that order are called *atoms* and we will write At for the set of them.

The syntax of wGKAT is parametric on two fixed sets: **Act** of uninterpreted program actions and **Out** of possible return values. The first five constructs in Figure 1, namely uninterpreted program actions, Boolean assertions, conditionals, sequential composition and while loops capture the syntax of GKAT. The *return values* are inherited from ProbGKAT [26] and intuitively correspond to the `return` operation from imperative programming languages.

The new syntactic construct in wGKAT is *weighted choice* denoted by $e \;{}_r{\oplus}_s\; f$. Intuitively, given two programs e and f, $e \;{}_r{\oplus}_s\; f$ represents a program that runs e with the weight of r and f with the weight of s. To lighten up notation, we will follow a convention that sequential composition binds tighter than weighted choice. Moreover, we define a scaling operation as the following syntactic sugar for a program that immediately continues with the weight of r.

▶ **Definition 2.1** (Scaling). $\odot r \coloneqq \mathbb{1} \;{}_r{\oplus}_0\; \mathbb{0}$

As a notational convention, scaling will take the highest precedence out of all wGKAT operators.

Weights in the aforementioned syntactic constructs are drawn from the set R equipped with a structure of a semiring. The usage of semirings for modeling weighted computation has been standard in automata theory, program semantics and verification communities [7]. We start by recalling the appropriate definitions concerning semirings.

▶ **Definition 2.2** ([11]). *A semiring $(R, +, \cdot, 0, 1)$ is a nonempty carrier set R equipped with two monoid structures interacting in the suitable way.*

1. $(R, +, 0)$ *is a commutative monoid with identity* 0
2. $(R, \cdot, 1)$ *is a monoid with identity element* $1 \neq 0$
3. $a \cdot (b + c) = a \cdot b + a \cdot c$ *and* $(b + c) \cdot a = b \cdot a + c \cdot a$ *for all* $a, b, c \in R$
4. $0 \cdot a = 0 = a \cdot 0$ *for all a in R*

We will abuse the notation and write multiplication in $(R, \cdot, 1)$ as juxtaposition; additionally we will write R to denote the whole algebraic structure, when the operations defined on the carrier are clear from the context. Note that additive and multiplicative identities of the semiring (respectively denoted by 0 and 1) are distinct from false and true tests (respectively denoted by $\mathbb{0}$ and $\mathbb{1}$).

Classic examples of semirings include (non-negative) reals/rationals with addition and multiplication and the tropical semiring of extended non-negative natural numbers (denoted by $\mathbb{N}^{+\infty}$), where the addition is given by pairwise minimum and semiring multiplication is a sum of extended natural numbers.

Later, we will consider semirings subject to some mild algebraic constraints, allowing us to meaningfully define an operational model for wGKAT programs, in particular with loops. We will make those constraints precise in further sections. Before we do so, we first give two examples of problems that could be encoded using the syntax that we have just described.

▶ **Example 2.1** (Ski Rental [2]). *Consider the problem of a person going on a ski trip for n days. Each day they either rent skis for a cost of 1, or they buy them for a cost of y and no* *longer have the need to rent. We can encode the situation of making this choice on the trip lasting n days, via the following* wGKAT *expression over the tropical semiring:*

$$(\mathbb{1}\ {}_1\!\oplus_y v)^n; v$$

Here, for positive $n \in \mathbb{N}$, we write e^n the n-fold sequential composition of the expression $e \in \mathbf{Exp}$.

Essentially, we perform at most n choices between paying 1 and immediately terminating upon paying y. The immediate termination is represented using a return value v.

▶ **Example 2.2** (Coin Game). *Consider the problem of playing a game where a coin is flipped and if it is heads the player wins a dollar and the game continues but if it is tails no money is won and the game ends. How much money should a player expect to make? To model expected value we will use weights to represent both values and probabilities. If we let Ⓢ be a return value representing the outcome of winning a dollar, then we can encode this situation via the following* wGKAT *expression over the semiring of extended non-negative rationals:*

$$((\mathbb{1}\ {}_1\!\oplus_1 (\odot 1; Ⓢ))\ {}_{.5}\!\oplus_{.5} (\odot 0; Ⓢ))^{(\mathbb{1})}$$

The while true makes the program continue to execute until some terminating output is reached. We interpret the choice weights as probabilities. The outer weighted choice captures the coin flip, so each branch executes with probability .5. The inner weighted choice is slightly different, it represents an "and". Each branch executes with probability 1. This can be thought of as similar to concurrently executing both branches rather than nondeterministically choosing one. The intuitive difference stems from the fact that addition in the rationals is not idempotent. The weights in the scalings represent the value *of the outcome* Ⓢ, *it is* how many *dollars the player wins.*

3 Operational Semantics

In this section, we present an operational semantics for wGKAT using a variant of weighted automata [7], which are akin to GKAT automata [30], where additionally each transition carries a weight taken from a semiring. Before we formally introduce our operational model, we recall the necessary definitions concerning weighted transitions.

For a set X and a fixed semiring $\mathbb{S}$, define $\mathcal{M}_\omega(X) = \{\varphi \mid \varphi\colon X \to \mathbb{S}, \mathsf{supp}(\varphi) \text{ is finite}\}$. Here, supp refers to the support of a function given by $\mathrm{supp}(\nu) = \{x \in X \mid \nu(x) \neq 0\}$. Elements of the set $\mathcal{M}_\omega(X)$ are functions $\nu : X \to \mathbb{S}$ which we will refer to as weightings on X over $\mathbb{S}$. We can view these as formal sums over X with coefficients from $\mathbb{S}$ or, alternatively, as a generalization of finitely supported distributions obtained by replacing probabilities in an unit interval with weights taken from a semiring and by dropping the normalization requirement. Because of this analogy, given an element $x \in X$, we can define an analog of Dirac delta specialized to weightings, namely $\delta_x : X \to \mathbb{S}$, which maps its argument to the semiring multiplicative identity 1 if it is equal to x, and the semiring additive identity 0 otherwise. Given a set $A \subseteq X$, we will write $\nu[A] \coloneqq \sum_{x \in A} \nu(x)$ for the total weight of the set A under the weighting ν. Moreover, any function $h\colon X \to \mathcal{M}_\omega(Y)$ can be uniquely extended to a function $\mathsf{lin}(h)\colon \mathcal{M}_\omega(X) \to \mathcal{M}_\omega(Y)$, called the *linearization of h*, given by

$$\mathsf{lin}(h)(\varphi)(y) = \sum_{x \in X} \varphi(x) h(x)(y)$$

Crucially, the above satisfies the property that $\mathsf{lin}(h) \circ \delta = h$, where $\delta : X \to \mathcal{M}_\omega(X)$ is defined by $\delta(x) = \delta_x$. A categorically minded reader might observe that weightings are precisely free semimodules over $\mathbb{S}$ and hence form a monad on the category of sets. The linerization $\mathsf{lin}(h)$ is the unique $\mathbb{S}$-semimodule homomorphism of the type $\mathcal{M}_\omega(X) \to \mathcal{M}_\omega(Y)$, that allows to factor h through δ. This is a consequence of a free-forgetful adjunction between the category of sets and the category of $\mathbb{S}$-semimodules.

wGKAT automata. A wGKAT automaton is a pair (X, β) consisting of a set of states X and a transition function $\beta : X \to \mathcal{M}_\omega(2 + \mathbf{Out} + \mathbf{Act} \times X)^{\mathrm{At}}$ that assigns to each state and a Boolean atom (capturing the current state of the variables) a weighting over three kinds of elements, representing different ways a state might perform a transition:

- an element of $2 = \{✓, ✗\}$, representing immediate acceptance or rejection;
- an element of $\mathbf{Out}$, representing a return value;
- a pair consisting of an atomic program action $\mathbf{Act}$ and a next state in X, representing performing a labelled transition to a next state.

To lighten up notation, we will write $\beta(x)_\alpha$ instead of $\beta(x)(\alpha)$.

▶ **Example 3.1.** *Let $X = \{x_1, x_2, x_3\}, \mathbb{S} = (\mathbb{N}^{+\infty}, +, \cdot, 0, 1), \mathrm{At} = \{\alpha, \beta\}, \mathbf{Act} = \{p_1, p_2, p_3\}$, $\mathbf{Out} = \{v\}$. We consider a* wGKAT *automaton (X, τ), whose pictorial representation is below on the left, while the definition of τ is below to the right.*

$$\begin{tikzcd}
✗ & \circ \arrow[l, dashed, "4"'] \arrow[r, dashed, "p_1|3"'] & x_2 \arrow[dr, "\alpha,\beta"'] & ✓ \\
x_1 \arrow[ur, "\alpha"] \arrow[dr, "\beta"'] & v & & \circ \arrow[u, dashed, "15"'] \\
& \circ \arrow[u, dashed, "1"] \arrow[uur, dashed, "p_2|5"] \arrow[r, dashed, "p_3|2"'] & x_3 \arrow[ur, "\alpha,\beta"'] &
\end{tikzcd}$$

$$\tau(x_1)_\alpha = 4\delta_{✗} + 3\delta_{(p_1,x_2)}$$
$$\tau(x_1)_\beta = 1\delta_v + 5\delta_{(p_2,x_2)} + 2\delta_{(p_3,x_3)}$$
$$\tau(x_2)_\alpha = \tau(x_2)_\beta = \tau(x_3)_\alpha = \tau(x_3)_\beta = 15\delta_{✓}$$

3.1 Operational semantics of Expressions

We are now ready to define the operational semantics of wGKAT expressions: we will construct a wGKAT automaton with transition function $\partial : \mathbf{Exp} \to \mathcal{M}_\omega(2 + \mathbf{Out} + \mathbf{Act} \times \mathbf{Exp})^{\mathrm{At}}$, whose states are wGKAT expressions. The semantics of an expression e, will be given by the behavior of the corresponding state in that automaton. This construction is reminiscent of the Brzozowski/Antimirov derivative construction for DFA and NFA respectively. For $\alpha \in \mathrm{At}$, we define $\partial(e)_\alpha$ by structural induction on e.

Tests, actions, output variables. Suppose $b \in \mathbf{BExp}, v \in \mathbf{Out}, p \in \mathbf{Act}$, we let

$$\partial(b)_\alpha = \begin{cases} \delta_{✓} & \alpha \leq_{\mathrm{BA}} b \\ \delta_{✗} & \alpha \leq_{\mathrm{BA}} \bar{b} \end{cases} \qquad \partial(v)_\alpha = \delta_v \qquad \partial(p)_\alpha = \delta_{(p,✓)}$$

A test either accepts or aborts with weight 1 depending on whether the given atom entails the truth of the test. An output variable outputs itself with weight 1 and then terminates. An action outputs the atomic program action and then accepts with weight 1, allowing for further computation after the action has been taken. We note here that when we do sequential composition, we will rewire acceptance into the composed expression. For this reason $\delta_{✓}$ can also be thought of as skipping with weight 1.

Guarded Choice. A guarded choice $+_b$ is an if-then-else statement conditioned on b. To capture this, we let $e +_b f$ have the outgoing edges of e if $\alpha \leq_{\mathsf{BA}} b$ and f otherwise.

$$\partial(e +_b f)_\alpha = \begin{cases} \partial(e)_\alpha & \alpha \leq b \\ \partial(f)_\alpha & \alpha \leq \bar{b} \end{cases}$$

Weighted Choice. The intuition here is that the automaton executes both arguments and scales their output by the given quantity. We define $e\,{}_r{\oplus}_s\, f$ as the automaton with all outgoing edges of e, scaled by r, and all outgoing edges of f, scaled by s. The derivative is:

$$\partial(e\,{}_r{\oplus}_s\, f)_\alpha = r \cdot \partial(e)_\alpha + s \cdot \partial(f)_\alpha$$

where these operations are the addition and scalar multiplication of the weightings.

Sequential Composition. We define the derivative of sequential composition as:

$$\partial(e; f)_\alpha = \partial(e)_\alpha \lhd_\alpha f$$

where given α and f we let $(- \lhd_\alpha f) : \mathcal{M}_\omega(2+\mathbf{Out}+\mathbf{Act}\times\mathbf{Exp}) \to \mathcal{M}_\omega(2+\mathbf{Out}+\mathbf{Act}\times\mathbf{Exp})$ be the linearization of $c_{\alpha,f} : 2 + \mathbf{Out} + \mathbf{Act} \times \mathbf{Exp} \to \mathcal{M}_\omega(2 + \mathbf{Out} + \mathbf{Act} \times \mathbf{Exp})$, given by:

$$c_{\alpha,f}(x) = \begin{cases} \partial(f)_\alpha & x = \checkmark \\ \delta_x & x \in \{\text{✗}\} \cup \mathbf{Out} \\ \delta_{(p,e';f)} & x = (p, e') \end{cases}$$

The intuition here is that if e skips, then the behavior of the composition is the behavior of f. If e aborts or returns a value then the derivative is the behavior of e. Finally, if e executes an action and transitions to another state, then the behavior is the next step of e composed with f along with emitting the given action.

Guarded loops A desired feature of the semantics of a program that performs loops is that each loop can be equivalently written as a guarded choice between acceptance and performing the loop body followed again by the loop. However, if the loop body could immediately accept, then the loop itself could non-productively accept any number of times before making a productive transition. Each time the loop immediately accepts the computation is scaled again by the weight of acceptance. One can represent this unbounded accumulation of weights through the notion of a fixpoint. There is a well-studied class of semirings featuring such a construct, namely Conway semirings. We first recall the necessary definitions.

▶ **Definition 3.1** ([8]). *A semiring $\mathbb{S}$ is a Conway Semiring if there exists a function $* : R \to R$ such that $\forall a, b \in R$*

$$(a + b)^* = a^*(ba^*)^* \tag{1}$$
$$(ab)^* = 1 + a(ba)^*b \tag{2}$$

We note that the *fixpoint rule* $aa^* + 1 = a^*$ holds in all Conway Semirings [8, p. 6].

The properties of Conway semirings are enough to guarantee that we can meaningfully define semantics of loops. We will see in Section 4 that these additional identities, Equations (1) and (2) in Definition 3.1, play a crucial role in the proof of soundness.

We now define the semantics of loops by iterating the weight of immediate acceptance:

$$\partial(e^{(b)})_\alpha(x) = \begin{cases} 1 & x = \checkmark \text{ and } \alpha \leq_{\mathsf{BA}} \bar{b} \\ \partial(e)_\alpha(\checkmark)^*\partial(e)_\alpha(x) & x \in \{\text{✗}\} \cup \mathbf{Out} \text{ and } \alpha \leq_{\mathsf{BA}} b \\ \partial(e)_\alpha(\checkmark)^*\partial(e)_\alpha(p, e') & x = (p, (e'; e^{(b)})) \text{ and } \alpha \leq_{\mathsf{BA}} b \\ 0 & \text{otherwise} \end{cases}$$

The loop could fail the Boolean test and then it just skips and does not execute the loop body (case 1). If the loop passes the test the body executes. Recall that the body might skip with some weight. So in this case the loop must either execute an action (case 3) or terminate (case 2) after some number of iterations. If the loop terminates (case 2) then we scale the weight of termination by the accumulated weight and terminate. If the loop takes an action (case 3), then we scale the weight of the action by the accumulated weight, emit the action, and compose the loop with the next step of e to re-enter the loop once computation of e has finished. We identify the behavior of divergent loops (case 4) with the *zero weighting*, since they never take a productive action nor terminate. Such a weighting simply assigns the weight zero to all possible branches of the program. This is distinct from the program that asserts false, which immediately aborts with weight one.

We now show that all expressions have a finite number of derivatives. This will be important for the completeness of our axiomatization and decidability of equivalence. We denote the set of states in the automaton reachable from expression e as $\langle e\rangle_\partial$.

▶ **Lemma 3.1.** *For all $e \in \mathbf{Exp}$, $\langle e\rangle_\partial$ is finite. We bound it by $\#(e) : \mathbf{Exp} \to \mathbb{N}$ where*

$$\#(b) = 1, \quad \#(v) = 1, \quad \#(p) = 2, \quad \#(e\,{}_r\!\oplus_s f) = \#(e) + \#(f),$$
$$\#(e +_b f) = \#(e) + \#(f), \quad \#(e; f) = \#(e) + \#(f), \quad \#(e^{(b)}) = \#(e)$$

3.2 Characterizing Equivalence

We will use bisimilarity as our notion of equivalence. Intuitively, two states of wGKAT automata are bisimilar if at each step of the continuing computation they are indistinguishable to an outside observer. This is a canonical notion of equivalence, that is more strict than language equivalence, and will allow us to present an efficient decision procedure in Section 6. We formalize what it means for states to be bisimilar in terms of automaton homomorphisms.

▶ **Definition 3.2.** *Given two* wGKAT *automata $(X, \beta), (Y, \gamma)$ a function $f : X \to Y$ is a* wGKAT *homomorphism if for all $x \in X$ and $\alpha \in \mathrm{At}$*

1. *For any $o \in 2 + \mathbf{Out}$* $\quad \gamma(f(x))_\alpha(o) = \beta(x)_\alpha(o)$
2. *For any $(p, y) \in \mathbf{Act} \times Y$* $\quad \gamma(f(x))_\alpha(p, y) = \beta(x)_\alpha[\{p\} \times f^{-1}(y)]$

We concretely instantiate the definition of bisimulation [24] for our automata, for the general version see Appendix A.

▶ **Definition 3.3.** *Let (X, β) and (Y, γ) be* wGKAT *automata. A relation $R \subseteq X \times Y$ is a bisimulation if there exists a transition function $\rho : R \to \mathcal{M}_\omega(2 + \mathbf{Out} + \mathbf{Act} \times R)^{At}$ such that the projections $\pi_1 : R \to X, \pi_2 : R \to Y$ are* wGKAT *homomorphisms from (R, ρ) to (X, β) and (Y, γ) respectively. We say the elements $x \in X, y \in Y$ are bisimilar if there exists a bisimulation R such that $(x, y) \in R$.*

To be able to build the combined transition structure over R in Definition 3.3 we require some additional properties that enable us to refine the original weightings on each set to construct more granular weightings over the relation. We achieve this by requiring that the *additive monoid* of the semiring has two properties: *refinement* [32] and *positivity* [14].

▶ **Definition 3.4.** *A monoid M is a refinement monoid if $x + y = z + w$ implies that there exist s, t, u, v such that $s + t = x, s + u = z, u + v = y, t + v = w$. A monoid M with identity e is positive (conical) if for all $x, y \in M$, $x + y = e \implies x = e = y$.*

Refinement extends inductively to sums with any (finite) number of summands [32]. Positivity says that no element other than the additive identity has an additive inverse. We now have a full characterization of the semirings we will consider: *positive, Conway, and refinement.* We offer the following list of semirings as examples for interpretation due to their relevance to weighted programming [2], and assume that all semirings discussed from now on have these three properties.

▶ **Corollary 3.1.** *The following semirings are positive, Conway, and refinement.*

1. *Tropical* $(\mathbb{N}^{+\infty}, \min, +, \infty, 0)$
2. *Arctic* $(\mathbb{N}^{+\infty}_{-\infty}, \max, +, -\infty, 0)$
3. *Bottleneck* $(\mathbb{R}^{+\infty}_{-\infty}, \max, \min, -\infty, \infty)$
4. *Formal languages* $(2^{\Gamma^*}, \cup, \cdot, \emptyset, \{\varepsilon\})$
5. *Extended naturals* $(\mathbb{N}^{+\infty}, +, \cdot, 0, 1)$
6. *Viterbi* $([0, 1], \max, \cdot, 0, 1)$
7. *Boolean* $(\{0, 1\}, \vee, \wedge, 0, 1)$
8. *Extended non-negative rationals* $(\mathbb{Q}^{+\infty}_{\geq 0}, +, \cdot, 0, 1)$

4 Axioms

In this section, we present an axiomatization of bisimulation equivalence for wGKAT expressions (Figure 2). We also derive some facts from the axioms which will be useful for both the examples and the proof of completeness (Section 4.1).

To explain the axioms in Figure 2, we first need to define a property similar to Salomaa's empty word property [27], which will be a familiar side condition for a fixpoint axiom.

▶ **Definition 4.1.** *We define $E\colon \mathbf{Exp} \to \mathrm{At} \to \mathbb{S}$ inductively as follows*

$$E(p)_\alpha = E(v)_\alpha = 0 \qquad E(e +_b f)_\alpha = \begin{cases} E(e)_\alpha & \alpha \leq_{\mathsf{BA}} b \\ E(f)_\alpha & \alpha \leq_{\mathsf{BA}} \bar{b} \end{cases}$$

$$E(b)_\alpha = \begin{cases} 1 & \alpha \leq_{\mathsf{BA}} b \\ 0 & \alpha \leq_{\mathsf{BA}} \bar{b} \end{cases} \qquad E(e; f)_\alpha = E(e)_\alpha E(f)_\alpha$$

$$E(e\, {}_r{\oplus}_s\, f)_\alpha = rE(e)_\alpha + sE(f)_\alpha \qquad E(e^{(b)})_\alpha = E(\bar{b})_\alpha$$

▶ **Lemma 4.1.** *Let $e \in \mathbf{Exp}, \alpha \in \mathrm{At}$. It holds that $E(e)_\alpha = \partial(e)_\alpha(\checkmark)$.*

Axiom Summary We define $\equiv\, \subseteq \mathbf{Exp} \times \mathbf{Exp}$ as the smallest congruence satisfying the axioms in Figure 2. Axioms G1-G4 are inherited from GKAT [30] and describe guarded choice. Axioms D1 and D2 state that weighted choice distributes over guarded choice and another weighted choice, respectively. D3 describes the interaction of tests and weighted choice.

Axioms S1-S7 characterize sequential composition. S4 is perhaps the most interesting; it states that composition right distributes over weighted choice just like guarded choice (S5). Since we equate the programs with respect to bisimilarity, which is a branching-time notion of equivalence, the symmetric rules of left distributivity are not sound. S1 and S2 state that wGKAT expressions have the structure of a monoid with identity $\mathbb{1}$, and with $v \in \mathbf{Out}$ and $\mathbb{0}$ being absorbent elements when composed on the left (S6, S3).

Guarded Choice Axioms

(G1) $e +_b e \equiv e$

(G2) $e +_b f \equiv b; e +_b f$

(G3) $e +_b f \equiv f +_{\bar{b}} e$

(G4) $(e +_b f) +_c g \equiv e +_{bc} (f +_c g)$

Distributivity Axioms

(D1) $e \,{}_r{\oplus}_s\, (f +_b g) \equiv (e \,{}_r{\oplus}_s\, f) +_b (e \,{}_r{\oplus}_s\, g)$

(D2) $e \,{}_r{\oplus}_s\, (f \,{}_t{\oplus}_u\, g) \equiv e \,{}_r{\oplus}_1\, (f \,{}_{st}{\oplus}_{su}\, g)$

(D3) $b; (e \,{}_r{\oplus}_s\, f) \equiv b; (b; e \,{}_r{\oplus}_s\, b; f)$

Sequencing Axioms

(S1) $\mathbb{1}; e \equiv e \equiv e; \mathbb{1}$

(S2) $(e; f); g \equiv e; (f; g)$

(S3) $\mathbb{0}; e \equiv \mathbb{0}$

(S4) $(e \,{}_r{\oplus}_s\, f); g \equiv e; g \,{}_r{\oplus}_s\, f; g$

(S5) $(e +_b f); g \equiv e; g +_b f; g$

(S6) $v; e \equiv v$

(S7) $b; c \equiv bc$

Loop Axioms

(L1) $e^{(b)} \equiv e; e^{(b)} +_b \mathbb{1}$

(L2) $$\frac{e \equiv (f \,{}_r{\oplus}_s\, \mathbb{1}) +_c g}{c; e^{(b)} \equiv c; ((\odot s^* r; f; (e)^{(b)}) +_b \mathbb{1})}$$

Scaling Axioms

(C1) $\odot\, 1 \equiv \mathbb{1}$

(C2) $\odot\, 0; e \equiv \odot 0$

Weighted Choice Axioms

(W1) $e \,{}_r{\oplus}_s\, e \equiv \odot(r + s); e$

(W2) $e \,{}_r{\oplus}_s\, f \equiv f \,{}_s{\oplus}_r\, e$

(W3) $e \,{}_r{\oplus}_s\, (f \,{}_t{\oplus}_u\, g) \equiv (e \,{}_r{\oplus}_{st}\, f) \,{}_1{\oplus}_{su}\, g$

(W4) $e \,{}_{ru}{\oplus}_s\, f \equiv (\odot u; e) \,{}_r{\oplus}_s\, f$

Fixpoint Rule

(F1) $$\frac{g \equiv e; g +_b f \quad \forall \alpha \in \mathrm{At}\ E(e)_\alpha = 0}{g \equiv e^{(b)}; f}$$

Figure 2 The axiomatization of wGKAT bisimulation equivalence. In these statements we let $e, f, g \in \mathbf{Exp}, b, c \in \mathbf{BExp}, v \in \mathbf{Out}, \alpha \in \mathrm{At}, p \in \mathbf{Act},\ r, s, t, u \in \mathbf{S}$. 1 and 0 weights refer to the multiplicative and additive identity in the chosen semiring.

C1-C2 are about scaling. Scaling by the multiplicative identity does nothing, it is the same as skip. Scaling by 0 annihilates any following expressions as it multiplies the weights by the multiplicative annihilator of the semiring. Though note that it is not the same as abort ($\mathbb{0}$), since scaling by 0 assigns weight 0 to ✗, while $\mathbb{0}$ assigns weight 1 to ✗. In this way both expressions annihilate any following expressions, but are not the same.

W1-W4 describe weighted branching. Identical branches can be combined, W1. The order of branches doesn't matter, W2. A repeated branching can be re-nested by re-weighting, W3. Finally, weights can be pushed through the branching into the following expression, W4.

▶ **Remark 4.1.** *We have made a choice to present some of our axioms using the syntactic sugar $\odot$. The axiomatization could be equivalently expressed without it. For example, if we were to express* C1 *without the syntactic sugar it would state $(\mathbb{1} \,{}_1{\oplus}_0\, \mathbb{0}); \mathbb{1} \equiv \mathbb{1}$. Stated explicitly, this axiom offers a way to remove weighted choices from expressions. Note that it is not derivable from other axioms of weighted choice as it is the only axiom with a weighted choice on the left-hand side but not on the right. The use of $\odot$ improves the readability and usability of the axioms. This is because it provides a compact representation of weighted choices with branches scaled by 0 (which are irrelevant for the behavior).*

Finally we discuss the loop axioms. F1 is the fixpoint rule. It states that fixpoints for productive loops are unique. We capture the notion of productivity of expressions, using an auxiliary operator E. L1 is standard loop unrolling. A loop body is executed no times or one or more depending on the test, so it is the same as making a guarded choice between skipping or executing the loop body one or more times. L2 allows unrolling and flattening the first set of iterations of a weighted loop if one of the branches is non-productive. The idea is that the loop can execute the nonproductive branch any number of times, but eventually must take the productive branch, after which it reenters the loop. Since a program action has now been taken, the boolean test c could now fail. Hence when the loop body is re-entered it is of the original form. Intuitively the loop has been unrolled up to the first time it executes the productive branch and this nonproductive unrolled portion has been flattened into one scaling followed by the productive program action and then the original loop.

▶ **Corollary 4.1** (Soundness). *For all $e, f \in \mathbf{Exp}$ $e \equiv f \implies e \sim f$*

4.1 Derivable facts

Some other useful identities are derivable from the axioms.

▶ **Lemma 4.2.** *For all $b, c \in \mathbf{BExp}$, $e, f, g, h \in \mathbf{Exp}$, $r, s, t, u \in \mathbb{S}$*

DF1	$\odot t; (e\ {}_{r}{\oplus}_{s}\ f) \equiv e\ {}_{tr}{\oplus}_{ts}\ f$	DF8	$b; (e +_c f) \equiv b; e +_c b; f$
DF2	$e +_b (f +_c g) \equiv (e +_b f) +_{b+c} g$	DF9	$b; (e +_c f) \equiv b; (b; e +_c f)$
DF3	$e +_b \mathbb{0} \equiv b; e$	DF10	$\odot r; \odot s \equiv \odot (rs)$
DF4	$b; (e +_b f) \equiv b; e$	DF11	$\odot r; (e +_b f) \equiv \odot r; e +_b \odot r; f$
DF5	$(e +_b f)\ {}_{r}{\oplus}_{s}\ g \equiv (e\ {}_{r}{\oplus}_{s}\ g) +_b (f\ {}_{r}{\oplus}_{s}\ g)$	DF12	$g\ {}_{r}{\oplus}_{s}\ \odot 0 \equiv \odot r; g$
DF6	$(e +_b f)\ {}_{r}{\oplus}_{s}\ (g +_b h) \equiv (e\ {}_{r}{\oplus}_{s}\ g) +_b (f\ {}_{r}{\oplus}_{s}\ h)$	DF13	$b; (e\ {}_{r}{\oplus}_{s}\ f) \equiv b; (b; e\ {}_{r}{\oplus}_{s}\ f)$
DF7	$e +_{\mathbb{1}} f \equiv e$	DF14	$g\ {}_{s}{\oplus}_{t}\ (e\ {}_{r}{\oplus}_{u}\ \odot 0) \equiv g\ {}_{s}{\oplus}_{tr}\ e$

With the axioms and derived facts in hand, we now revisit our original examples. In the *ski rental* problem the relevant property is the minimum total cost that could be paid over these sequential days. An appropriate choice of semiring to weight over is the *tropical semiring*, as it allows us to reason about the cheapest branch of computation using sum to produce branches and minimum to compare branches.

▶ **Example 4.1** (Ski Rental). *We now prove that the optimal cost is buying skis if the trip is longer than n days and renting skis otherwise. Recall the encoding was $(\mathbb{1}\ {}_{1}{\oplus}_{y}\ v)^n; v$ where n is the number days of the trip, y the cost of skis, and v terminates the process.*

We first prove by induction on n the lemma $(\mathbb{1}\ {}_{1}{\oplus}_{y}\ v)^n \equiv (\mathbb{1}\ {}_{n}{\oplus}_{y}\ v)$. The base case where $n = 1$ is trivial. Suppose it is true for $n = k$.

$$\begin{aligned}
(\mathbb{1}\ {}_{1}{\oplus}_{y}\ v)^{k+1} &\equiv (\mathbb{1}\ {}_{1}{\oplus}_{y}\ v)^{k}; (\mathbb{1}\ {}_{1}{\oplus}_{y}\ v) && \text{(Definition of } e^n\text{)}\\
&\equiv (\mathbb{1}\ {}_{k}{\oplus}_{y}\ v); (\mathbb{1}\ {}_{1}{\oplus}_{y}\ v) && \text{(Induction hypothesis)}\\
&\equiv (\mathbb{1}; (\mathbb{1}\ {}_{1}{\oplus}_{y}\ v)\ {}_{k}{\oplus}_{y}\ v; (\mathbb{1}\ {}_{1}{\oplus}_{y}\ v)) && \text{(S4)}\\
&\equiv ((\mathbb{1}\ {}_{1}{\oplus}_{y}\ v)\ {}_{k}{\oplus}_{y}\ v) && \text{(S1, S6)}\\
&\equiv (v\ {}_{y}{\oplus}_{k}\ (v\ {}_{y}{\oplus}_{1}\ \mathbb{1})) && \text{(W2)}\\
&\equiv ((v\ {}_{y}{\oplus}_{ky}\ v)\ {}_{0}{\oplus}_{k1}\ \mathbb{1}) && \text{(W3)}\\
&\equiv ((\odot(y + ky); v)\ {}_{0}{\oplus}_{k1}\ \mathbb{1}) && \text{(W1)}\\
&\equiv ((\odot y; v)\ {}_{0}{\oplus}_{k1}\ \mathbb{1}) && (y + ky = y)\\
&\equiv (v\ {}_{y}{\oplus}_{k1}\ \mathbb{1}) && \text{(W4)}\\
&\equiv (\mathbb{1}\ {}_{k1}{\oplus}_{y}\ v) && \text{(W2)}
\end{aligned}$$

We now use this fact to simplify the expression of the ski rental problem:

$$\begin{aligned}
(\mathbb{1}\ {}_{1}{\oplus}_{y}\ v)^{n}; v &\equiv (\mathbb{1}\ {}_{n}{\oplus}_{y}\ v); v && \\
&\equiv (\mathbb{1}; v\ {}_{n}{\oplus}_{y}\ v; v) && \text{(S4)}\\
&\equiv (v\ {}_{n}{\oplus}_{y}\ v) && \text{(S1, S6)}
\end{aligned}$$

$$\equiv \odot(n+y); v \qquad \text{(W1)}$$

This is termination with weight equal to the minimum of n and y.

▶ **Example 4.2** (Coin Game). *Recall the encoding is $((\mathbb{1}\ {}_1{\oplus}_1\ (\odot 1; \textcircled{\$}))\ {}_{.5}{\oplus}_{.5}\ (\odot 0; \textcircled{\$}))^{(\mathbb{1})}$, where $\textcircled{\$}$ is the outcome of winning a dollar. We are interested in expected value, so we choose the* extended non-negative rational semiring *because this can capture both the probabilities and the values of outcomes. We choose rationals rather than the reals to avoid having to compare infinite precision numbers. We now simplify the encoding:*

$$\begin{aligned}
&((\mathbb{1}\ {}_1{\oplus}_1\ (\odot 1; \textcircled{\$}))\ {}_{.5}{\oplus}_{.5}\ (\odot 0; \textcircled{\$}))^{(\mathbb{1})} \\
&\quad \equiv ((\mathbb{1}\ {}_1{\oplus}_1\ (\odot 1; \textcircled{\$}))\ {}_{.5}{\oplus}_{.5}\ (\odot 0; \mathbb{0}))^{(\mathbb{1})} && \text{(C2)} \\
&\quad \equiv ((\mathbb{1}\ {}_1{\oplus}_1\ (\odot 1; \textcircled{\$}))\ {}_{.5}{\oplus}_{0}\ \mathbb{0})^{(\mathbb{1})} && \text{(W4)} \\
&\quad \equiv (\odot .5(\mathbb{1}\ {}_1{\oplus}_1\ (\odot 1; \textcircled{\$})))^{(\mathbb{1})} && \text{(Definition 2.1)} \\
&\quad \equiv (\odot .5(\mathbb{1}\ {}_1{\oplus}_1\ \textcircled{\$}))^{(\mathbb{1})} && \text{(C1, S1)} \\
&\quad \equiv (\mathbb{1}\ {}_{.5}{\oplus}_{.5}\ \textcircled{\$})^{(\mathbb{1})} && \text{(DF1)} \\
&\quad \equiv (\textcircled{\$}\ {}_{.5}{\oplus}_{.5}\ \mathbb{1})^{(\mathbb{1})} && \text{(W2)} \\
&\quad \equiv (\textcircled{\$}\ {}_{.5}{\oplus}_{.5}\ \mathbb{1} +_{\mathbb{1}} \mathbb{0})^{(\mathbb{1})} && \text{(DF7)} \\
&\quad \equiv \mathbb{1}; \left(\left(\odot(.5^* \cdot .5); \textcircled{\$}; ((\textcircled{\$}\ {}_{.5}{\oplus}_{.5}\ \mathbb{1}) +_{\mathbb{1}} \mathbb{0})^{(\mathbb{1})}\right) +_{\mathbb{1}} \mathbb{1}\right) && \text{(L2)} \\
&\quad \equiv \odot(.5^* \cdot .5); \textcircled{\$}; (\textcircled{\$}\ {}_{.5}{\oplus}_{.5}\ \mathbb{1} +_{\mathbb{1}} \mathbb{0})^{(\mathbb{1})} && \text{(S1, DF7)} \\
&\quad \equiv \odot(.5^* \cdot .5); \textcircled{\$} && \text{(S6)} \\
&\quad \equiv \odot(2 \cdot .5); \textcircled{\$} \\
&\quad \equiv \odot 1; \textcircled{\$}
\end{aligned}$$

Hence, the player expects to win one dollar in this game.

5 Completeness

It is natural to ask if the set of axioms presented in Section 4 is *complete*, i.e., whether bisimilarity of any pair of wGKAT expressions can be established solely through the means of axiomatic manipulation. As we will see later on, the problem of completeness essentially relies on representing wGKAT automata syntactically as certain systems of equations and uniquely solving them. This is similar to proofs of completeness one might encounter in the literature concerning process algebra and Kleene Algebra.

Unfortunately, similarly to GKAT [30] and its probabilistic exension [26], this poses a few technical challenges. Firstly, unlike Kleene Algebra, we do not have a perfect correspondence between the operational model (wGKAT automata) and the expressivity of the syntax. Namely, there exist wGKAT automata that cannot be described by a wGKAT expression. This simply stems from the fact that our syntax is able to only describe programs with well-structured control flow using while loops and if-then-else statements, which is strictly less expressive than using arbitrary non-local flow involving constructs like goto expressions in an uninterpreted setting. Secondly, classic proofs of completeness of Kleene Algebra strictly rely on the left distributivity of sequential composition over branching, which is unsound under bisimulation semantics. This is crucial, as it allows for reduction of the problem of solving an arbitrary system of equations to the unary case that can be handled through the inference rule similar to F1 of our system.

In the presence of these problems, we circumvent those issues by extending our inference system with a powerful axiom schema generalizing F1 to arbitrary systems of equations (hence avoiding the issue of left distributivity) that states that each such system has *at most one solution* (thus solving the problem of insufficient expressivity of our syntax). We will follow the GKAT terminology and refer to this axiom schema as the *Uniqueness axiom* (UA) [30], but the general idea is standard in process algebra and was explored by Bergstra under the name *Recursive Specification Principle* (RSP) [3].

To obtain the completeness result, we will rely on the following roadmap:

1. First, we show that every wGKAT expression possesses a certain normal form that ties it to its operational semantics.
2. Then, we define systems of equations akin to the normal form the step above and define what it means to solve them.
3. Relying on the above two steps, we prove the correspondence between solutions to systems of equations and wGKAT automata homomorphisms.
4. We establish the soundness of the Uniqueness Axiom. As a consequence of the correspondence from the step above, we show that two expressions being related by some bisimulation can be seen as them being solutions to the same system of equations. This combined with the use of UA guarantees completeness.

5.1 Fundamental Theorem

In order to prove completeness we will use an intermediate result which states that each expression can be simplified to a normal form, obtained from the operational semantics of the expression. Following the terminology from the Kleene Algebra community we call this the *fundamental theorem* [25].

▶ **Definition 5.1** ([26]). *Given $\Phi \subseteq \mathrm{At}$ and an indexed collection $\{e_\alpha\}_{\alpha\in\Phi}$ where $\forall\alpha \in \Phi, e_\alpha \in \mathbf{Exp}$ we define a generalized guarded sum inductively as*

$$\mathop{+\!\!\!\!+}_{\alpha\in\Phi} = \mathbb{0} \text{ if } \Phi = \emptyset \qquad \mathop{+\!\!\!\!+}_{\alpha\in\Phi} = e_\gamma +_\gamma \mathop{+\!\!\!\!+}_{\alpha\in\Phi\setminus\gamma} e_\alpha \quad \gamma \in \Phi$$

▶ **Definition 5.2.** *Given a non-empty finite index set I and indexed collections $\{r_i\}_{i\in I}$ and $\{e_i\}_{i\in I}$ such that $e_i \in \mathbf{Exp}$ and $r_i \in \mathbb{S}$ for all $i \in I$ we define a sum expression inductively:*

$$\bigoplus_{i\in I} r_i \cdot e_i = e_j \,{}_{r_j}\!\oplus_1 \left(\bigoplus_{i\in I\setminus\{j\}} r_i \cdot e_i \right)$$

Both Definition 5.1 and Definition 5.2 are well defined up to the order they are unrolled. See Appendix D for details.

▶ **Theorem 5.1** (Fundamental Theorem). *For every $e \in \mathbf{Exp}$ it holds that*

$$e \equiv \mathop{+\!\!\!\!+}_{\alpha\in\mathrm{At}} \left(\bigoplus_{d\in supp(\partial(e)_\alpha)} \partial(e)_\alpha(d) \cdot \exp(d) \right)$$

where exp defines a function $2 + \mathbf{Out} + \mathbf{Act} \times \mathbf{Exp} \to \mathbf{Exp}$ given by

$$\exp(✗) = \mathbb{0} \quad \exp(✓) = \mathbb{1} \quad \exp(v) = v \quad \exp((p, f)) = p; f \quad (v \in \mathbf{Out}, p \in \mathbf{Act}, f \in \mathbf{Exp})$$

5.2 Solutions to Systems of Equations

System of equations We are now ready to characterize systems of equations and their solutions. We first introduce a three-sorted grammar to mimic the way that atoms index weighted choices in the fundamental theorem.

$$\begin{aligned}
\mathbf{Exp}(X) &\coloneqq \mathop{+\!\!\!\!+}_{\alpha\in\mathrm{At}} w_\alpha && (\forall\alpha\in\mathrm{At}\ \ w_\alpha\in\mathbf{WExp}(X))\\
\mathbf{WExp}(X) &\coloneqq \bigoplus_{i\in I} r_i\cdot t_i && (t_i\in\mathbf{TExp}, r_i\in\mathbb{S}, I\text{ finite})\\
\mathbf{TExp}(X) &\coloneqq \mathbf{f}\mid \mathbf{g}x && (\mathbf{f},\ \mathbf{g}\in\mathbf{Exp}, x\in X)
\end{aligned}$$

Note that the above is well defined only if we take a congruence containing our axioms of both guarded and weighted choice, as the generalized version of guarded choice is defined up to skew-associativity and skew-commutativity. That is, an arbitrary order may be picked when selecting an element of $\mathbf{Exp}(X)$, but after substituting wGKAT expressions for indeterminates $x\in X$ it won't matter which order was selected. See Appendix D for details.

▶ **Definition 5.3** ([26]). *A system of equations is a pair $(X,\tau : X\to\mathbf{Exp}(X))$ consisting of a finite set X of indeterminates and a function τ. If for all $x\in X,\alpha\in\mathrm{At}$ in each $\tau(x)$ the $\boldsymbol{g}x$ subterms satisfy $E(g)_\alpha = 0$, the system of equations is called* Salomaa.

We now define the system of equations associated with a given wGKAT automaton.

▶ **Definition 5.4.** *Let (X,β) be a* wGKAT *automaton. A system of equations associated with (X,β) is a Salomaa system (X,τ), with $\tau : X\to\mathbf{Exp}(X)$ defined by*

$$\tau(x) = \mathop{+\!\!\!\!+}_{\alpha\in\mathrm{At}}\left(\bigoplus_{d\in supp(\beta(x))_\alpha}\beta(x)_\alpha(d)\cdot sys(d)\right)$$

where $sys : 2+\mathbf{Out}+\mathbf{Act}\times\mathbf{Exp}\to\mathbf{WExp}(X)$ is given by

$$sys(\text{✗}) = \mathbb{0}\quad sys(\text{✓}) = \mathbb{1}\quad sys(v) = v\quad sys(p,x) = px$$

▶ **Example 5.1.** *The τ (up to $\equiv$) associated with the automaton from Example 3.1 is*

$$\begin{aligned}
x_1 &\mapsto (\mathbb{0}\ {}_4{\oplus}_3\ (p_1x_2)) +_\alpha (v\ {}_1{\oplus}_1\ ((p_2x_2)\ {}_5{\oplus}_2\ (p_3x_3)) +_\beta \mathbb{0})\\
x_2 &\mapsto \mathbb{1}\ {}_{15}{\oplus}_0\ \mathbb{0} +_\alpha (\mathbb{1}\ {}_{15}{\oplus}_0\ \mathbb{0} +_\beta \mathbb{0})\\
x_3 &\mapsto \mathbb{1}\ {}_{15}{\oplus}_0\ \mathbb{0} +_\alpha (\mathbb{1}\ {}_{15}{\oplus}_0\ \mathbb{0} +_\beta \mathbb{0})
\end{aligned}$$

▶ **Definition 5.5** ([26]). *Given a function $h : X\to\mathbf{Exp}$ that assigns a value to each indeterminate in X we derive a* wGKAT *expression $h^\#(e)$ for each $e\in\mathbf{Exp}(X)$ inductively:*

$$h^\#(f) = f \qquad (3) \qquad\qquad h^\#(w_1\ {}_r{\oplus}_s\ w_2) = h^\#(w_1)\ {}_r{\oplus}_s\ h^\#(w_2) \qquad (5)$$

$$h^\#(e +_b f) = h^\#(e) +_b h^\#(f) \qquad (4) \qquad\qquad h^\#(\boldsymbol{g}x) = \boldsymbol{g}; h(x) \qquad (6)$$

We now characterize a solution to the Salomaa system of equations.

▶ **Definition 5.6** ([26]). *Let $R\subseteq\mathbf{Exp}\times\mathbf{Exp}$ be a congruence. A solution up to R of a system (X,τ) is a map $h : X\to\mathbf{Exp}$ such that for all $x\in X$ that $(h(x),h^\#(\tau(x)))\in R$*

▶ **Example 5.2.** *A solution up to $\equiv$ of Example 5.1 would satisfy*

$$\begin{aligned}
h(x_1) &\equiv (\mathbb{0}\,{}_4{\oplus}_3\, p_1; h(x_2)) +_\alpha (v\,{}_1{\oplus}_1\,(p_2; h(x_2)\,{}_5{\oplus}_2\, p_3; h(x_3)) +_\beta \mathbb{0})\\
h(x_2) &\equiv \mathbb{1}\,{}_{15}{\oplus}_0\, \mathbb{0} +_\alpha (\mathbb{1}\,{}_{15}{\oplus}_0\, \mathbb{0} +_\beta \mathbb{0})\\
h(x_3) &\equiv \mathbb{1}\,{}_{15}{\oplus}_0\, \mathbb{0} +_\alpha (\mathbb{1}\,{}_{15}{\oplus}_0\, \mathbb{0} +_\beta \mathbb{0})
\end{aligned}$$

So in this case we can select

$$\begin{aligned}
h(x_2) = h(x_3) &= \odot 15\\
h(x_1) &= (\mathbb{0}\,{}_4{\oplus}_3\,(p_1; \odot 15)) +_\alpha (v\,{}_1{\oplus}_1\,((p_2; \odot 15)\,{}_5{\oplus}_2\,(p_3; \odot 15)) +_\beta \mathbb{0})
\end{aligned}$$

We now can show the connection between solutions to the Salomaa system of equations and automaton homomorphisms using the equivalence classes of states.

▶ **Theorem 5.2.** *Let (X, β) be a finite* wGKAT *automaton. The map $h : X \to \mathbf{Exp}$ is a solution up to $\equiv$ of the system associated with (X, β) if and only if $[-]_\equiv \circ h$ is a* wGKAT *automaton homomorphism from (X, β) to $(\mathbf{Exp}/\equiv, \bar{\partial})$. Where $\bar{\partial}$ is the unique transition function on $\mathbf{Exp}/\equiv$ which makes the quotient map $[-]_\equiv : \mathbf{Exp} \to \mathbf{Exp}/\equiv$ a* wGKAT *automaton homomorphism from $(\mathbf{Exp}, \partial)$ to $(\mathbf{Exp}/\equiv, \bar{\partial})$.*

5.3 Bisimilarity implies Provable Equivalence

Having characterized systems of equations and their solutions we now present the uniqueness axiom and establish completeness. Let $\doteqdot \subseteq \mathbf{Exp} \times \mathbf{Exp}$ be the least congruence that contains $\equiv$, and satisfies the following axiom:

$$\frac{(X,\tau) \text{ is a Salomaa system} \quad f, g : X \to \mathbf{Exp} \text{ are solutions of } (X,\tau) \text{ up to } \doteqdot}{f(x) \doteqdot g(x) \text{ for all } x \in X} \tag{UA}$$

We establish the soundness of the axiom in the same way as before.

▶ **Theorem 5.3** (Soundess of $\doteqdot$)**.** *For all $e, f \in \mathbf{Exp}$, $e \doteqdot f \implies e \sim f$*

Finally, we can prove completeness using the uniqueness axiom.

▶ **Theorem 5.4** (Completeness)**.** *For all $e, f \in \mathbf{Exp}$, $e \sim f \implies e \doteqdot f$*

6 Decidability and Complexity

In practice, one often cares about a mechanistic way to check program equivalence. We show that bisimulation equivalence for wGKAT expressions is indeed decidable, and we offer an efficient procedure to decide it. This sets our language apart from other weighted KAT variants, in particular KAWT, which does not offer a decision procedure for equivalence and is likely not decidable due to using trace equivalence. The key insight is that bisimulation equivalence lets us appeal to known theory of monoid-weighted transition systems and achieve a decision procedure which, remarkably, is polynomial time.

Similar to ProbGKAT [26], we rely on a *coalgebraic* generalization [5] of classic partition refinement [18, 22] to minimize the automaton under bisimilarity which essentially computes the largest bisimulation. Coalgebraic partition refinement offers a procedure to minimize any coalgebra that fits a grammar of supported functors and polynomial constructors [5]. As mentioned before, wGKAT can be viewed as coalgebras. So in order to establish the decidability of bisimulation equivalence for wGKAT automata we show how to express the equivalent coalgebra in the grammar which can be decided by coalgebraic partition refinement. However, in order to determine a bound on the runtime of this procedure we must bound some of the internal operations of the coalgebraic partition refinement algorithm.

To bound the runtime of the coalgebraic partition refinement algorithm we encode our coalgebra in a format with a uniform label set, and then bound the comparison of labels, the **init** step, and the **update** step [5]. The **init** step essentially takes the set of labels on edges from a state and computes the total weight of those edges. The **update** step basically uses a state's labels into a set of states S, and the weight of the same state's edges into another set of states B and computes the weight of edges into S and C/S, as well as computing transitions into blocks of related states [5, 6]. Once we bound these operations we can leverage a complexity result [5, Theorem 3.4] of coalgebraic partition refinement to achieve the desired runtime bound. The details of both of these arguments are in Appendix E.

▶ **Corollary 6.1** (Decidability). *If $e, f \in \mathbf{Exp}$, $n = \#e + \#f$, and* At *is fixed, then* wGKAT *equivalence of e and f is decidable in time $O(n^3 \log^2(n))$*

▶ **Remark 6.1.** *The astute reader will know that the equivalence of rational power series in the tropical semiring is undecidable [21] and might reasonably doubt the decidability of equivalence of* wGKAT *expressions. This does not pose an issue for* wGKAT*. Our notion of equivalence is one of bisimilarity, and hence single step equivalence, not trace equivalence. Due to this, deciding equivalence of* wGKAT *terms over the tropical semiring reduces to deciding equivalence of terms over the* additive monoid *of the tropical semiring, not the tropical semiring itself. For this reason equivalence of* wGKAT *automata can be decided without deciding equivalence of rational power series. For this reason equivalence with respect to bisimilarity of* wGKAT *expressions is decidable even over the tropical semiring.*

7 Related Work

Our work comes in a long line of work on Kleene Algebra with Tests (KAT) [20]. This is an algebraic framework for reasoning about programs based on Kleene Algebra, the algebra of regular expressions. The original model was one with the axioms of Kleene Algebra combined with a boolean algebra of tests, giving control flow to a algebraic model. A restriction of this language to the deterministic fragment led to GKAT which allows for decision of equivalence in near linear time [30]. Following this, a deterministic probabilistic language, ProbGKAT, was developed which was also efficiently decidable [26]. Our work is directly inspired by ProbGKAT, although it is orthogonal as an extension of GKAT with semiring weightings. ProbGKAT extended GKAT with convex sums in order to capture probability, in contrast to the general weighted sums we introduce in this paper which do not enforce convexity [26]. As a result the semantics and axiomatizations differ considerably in form but also in terms of the techniques needed to prove results. We took inspiration for our arguments from the general theory of effectful process algebras [28].

Both idempotent and non-idempotent semirings have been examined and used as models for computation [19, 7]. While our work is directly inspired by [20], the work on various non-idempotent *-semirings was invaluable to our characterization of semirings appropriate for weighting over. In particular we considered both iteration semirings [8] and inductive *-semirings [9]. The work on classifying *-semirings is quite thorough and may offer useful classes for researchers working on weighted computation. Semirings in general are well studied and we refer to this theory at various points for our arguments [11].

We would be remiss to not mention the work on Kleene Algebra with Weights and Tests (KAWT) [29]. This is an extension of KAT with weights. It works by weighting the nondeterministic branches of computation in KAT and summing weights. The syntax is quite similar to our own. KAWT is based on a relational semantics which assigns weights to guarded strings in contrast to our coalgebra-based semantics. The syntax of KAWT introduces another class of expression to KAT, the weightings. This class behaves like the Boolean tests in KAT, allowing for both identities, addition and multiplication. These terms are then subterms of programs. For a semiring to be used as the weights in a KAWT it must be idempotent and complete. This allows for a definition of iteration, while keeping each KAT a KAWT. An idempotent semiring is one in which $\forall s \in \mathbb{S},\ \ s + s = s$ and a complete semiring is one in which there is an infinitary sum operator $\sum_I$ for any index set I, and the sum obeys some natural notions of sums and multiplication.

Two key differences between wGKAT and KAWT languages are decidability and the admissible class of semirings. In particular no decidability results exist for KAWT [29] and idempotent and complete semirings are considered. We require the semiring be positive, Conway and refinement. All complete semirings are positive [11, 22.28], and all complete *-semirings are Conway semirings [7, Theorem 3.4]. So roughly speaking, we impose the new restriction of being a refinement monoid, but in exchange we drop the requirement of idempotence and loosen the requirement of completeness. We find that this trade includes several new useful semirings, like the natural numbers with infinity under addition and multiplication, or even the non-negative extended reals under addition and multiplication, while ruling out only pathological examples.

Finally our work on the wGKAT automaton leverages the existing work on the general theory of coalgebra [24, 17, 23, 14] and in particular monoid-weighted transition systems [16]. We found the property of refinement in monoids to particularly useful and the existing literature to be quite thorough [32, 16]. These ideas of mutual refinements are relevant to the semantics of weighted programming. Of course we leverage an existing algorithm for our decision procedure for bisimulation equivalence [5, 33, 6].

8 Conclusion

We presented wGKAT, a weighted language inspired by Kleene Algebra with Tests, equipped with sequential composition, weighted branching over a large class of semirings, conditionals, Boolean tests, primitive actions, and guarded loops. We proposed an operational semantics and gave a sound and complete axiomatization of bisimilarity. Finally, we showed that bisimulation equivalence of wGKAT is efficiently decidable in $O(n^3 \log^2(n))$ time.

One extension of this work would be to axiomatize the coarser notion of trace equivalence, which brings in new axioms. For example, left distributivity over guarded and weighted choice would seem to be sound under trace equivalence, given some conditions on the weights involved. The undecidability of equivalence of power series in the tropical semiring poses issues for a decision procedure for trace equivalence. However characterizing a decision procedure based on properties of the semiring would be an interesting result.

A practical direction for research would be to take these ideas of weightings and apply them to NetKAT, a variant of KAT for verifying network properties. This would allow for examination of quantitative properties which may be of interest to network operators.

Another direction for future research is to prove completeness without the semiring being refinement and positive. These conditions were needed for a bisimulation to exist, which is what made the completeness theorem possible. If it were possible to prove completeness without laying hands on a bisimulation, then behavioral equivalence could still be established without these properties. While we are only aware of pathological examples of semirings which fail to be refinement, allowing for non-positive semirings would include many familiar semirings. Finally, tackling the question of completeness without the uniqueness axiom which remains open for GKAT, ProbGKAT, and now wGKAT.

References

1 Steve Awodey. *Category theory*, volume 52. Oxford University Press, Inc., 2010.

2 Kevin Batz, Adrian Gallus, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Tobias Winkler. Weighted programming, 2022. `arXiv:2202.07577`, `doi:10.48550/arXiv.2202.07577`.

3 Jan A. Bergstra and Jan Willem Klop. Verification of an alternating bit protocol by means of process algebra. In Wolfgang Bibel and Klaus P. Jantke, editors, *Mathematical Methods of Specification and Synthesis of Software Systems '85, Proceedings of the International Spring School, Wendisch-Rietz, GDR, April 22-26, 1985*, volume 215 of *Lecture Notes in Computer Science*, pages 9–23, Berlin, Heidelberg, 1985. Springer. URL: `https://doi.org/10.1007/3-540-16444-8_1`, `doi:10.1007/3-540-16444-8_1`.

4 E.P. de Vink and Jan J. M. M. Rutten. Bisimulation for probabilistic transition systems: a coalgebraic approach. *Theoretical Computer Science*, 221(1):271–293, 1999. `doi:10.1016/S0304-3975(99)00035-3`.

5 Hans-Peter Deifel, Stefan Milius, Lutz Schröder, and Thorsten Wißmann. Generic partition refinement and weighted tree automata. In Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira, editors, *Formal Methods – The Next 30 Years*, pages 280–297, Cham, 2019. Springer International Publishing. `doi:10.1007/978-3-030-30942-8_18`.

6 Ulrich Dorsch, Stefan Milius, Lutz Schröder, and Thorsten Wißmann. Efficient coalgebraic partition refinement, 2017. URL: `https://arxiv.org/abs/1705.08362`, `arXiv:1705.08362`, `doi:10.48550/arXiv.1705.08362`.

7 M Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata.* Springer-Verlag, 2009. `doi:10.1007/978-3-642-01492-5`.

8 Zoltán Ésik. Iteration semirings. In Masami Ito and Masafumi Toyama, editors, *Developments in Language Theory*, pages 1–20, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. `doi:10.1007/978-3-540-85780-8_1`.

9 Zoltán Ésik and Werner Kuich. Inductive star-semirings. *Theor. Comput. Sci.*, 324(1):3–33, 2004. Words, Languages and Combinatorics. `doi:10.1016/j.tcs.2004.03.050`.

10 Jonathan S. Golan. *Semirings and Affine Equations over Them: Theory and Applications.* Springer Netherlands, Dordrecht, 2003. `doi:10.1007/978-94-017-0383-3`.

11 Jonathan S Golan. *Semirings and their Applications.* Springer Science & Business Media, 2013. `doi:10.1007/978-94-015-9333-5`.

12 H. Gumm and Tobias Schröder. Types and coalgebraic structure. *Algebra Universalis*, 53:229–252, 08 2005. `doi:10.1007/s00012-005-1888-2`.

13 H Peter Gumm. Elements of the general theory of coalgebras, 1999.

14 H Peter Gumm. Copower functors. *Theoretical Computer Science*, 410(12-13):1129–1142, 2009. `doi:10.1016/j.tcs.2008.09.057`.

15 H. Peter Gumm and Ralph Freese. Free-lattice functors weakly preserve epi-pullbacks. *Algebra universalis*, 83:1–18, 2021. URL: `https://api.semanticscholar.org/CorpusID:232257632`, `doi:10.1007/s00012-022-00774-5`.

16 H. Peter Gumm and Tobias Schröder. Monoid-labeled transition systems. In Andrea Corradini, Marina Lenisa, and Ugo Montanari, editors, *Coalgebraic Methods in Computer Science, CMCS 2001, a Satellite Event of ETAPS 2001, Genova, Italy, April 6-7, 2001*, volume 44 of *Electronic Notes in Theoretical Computer Science*, pages 185–204. Elsevier, 2001. CMCS 2001, Coalgebraic Methods in Computer Science (a Satellite Event of ETAPS 2001). `doi:10.1016/S1571-0661(04)80908-3`.

17 H. Peter Gumm and Tobias Schröder. Coalgebras of bounded type. *Math. Struct. Comput. Sci.*, 12(5):565–578, 2002. `doi:10.1017/S0960129501003590`.

18 Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. In Robert L. Probert, Nancy A. Lynch, and Nicola Santoro, editors, *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada, August 17-19, 1983*, PODC '83, pages 228–240, New York, NY, USA, 1983. ACM. doi:10.1145/800221.806724.

19 Dexter Kozen. A completeness theorem for kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994. URL: https://www.sciencedirect.com/science/article/pii/S0890540184710376, doi:10.1006/inco.1994.1037.

20 Dexter Kozen. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst.*, 19(3):427–443, may 1997. doi:10.1145/256167.256195.

21 Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. In W. Kuich, editor, *Automata, Languages and Programming*, pages 101–112, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. doi:10.1007/3-540-55719-9_67.

22 Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987. arXiv:https://doi.org/10.1137/0216062, doi:10.1137/0216062.

23 H. Peter Gumm and Tobias Schröder. Coalgebraic structure from weak limit preserving functors. *Electronic Notes in Theoretical Computer Science*, 33:111–131, 2000. CMCS'2000, Coalgebraic Methods in Computer Science. URL: https://www.sciencedirect.com/science/article/pii/S1571066105803469, doi:10.1016/S1571-0661(05)80346-9.

24 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.

25 Jan J. M. M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theor. Comput. Sci.*, 308(1-3):1–53, 2003. doi:10.1016/S0304-3975(02)00895-2.

26 Wojciech Różowski, Tobias Kappé, Dexter Kozen, Todd Schmid, and Alexandra Silva. Probabilistic guarded kat modulo bisimilarity: Completeness and complexity, 2023. arXiv:2305.01755, doi:10.48550/arXiv.2305.01755.

27 Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, jan 1966. doi:10.1145/321312.321326.

28 Todd Schmid. *Coalgebraic Completeness Theorems for Effectful Process Algebras.* Phd thesis 1 (research tu/e / graduation tu/e), 2024.

29 Igor Sedlár. Kleene algebra with tests for weighted programs. *2023 IEEE 53rd International Symposium on Multiple-Valued Logic (ISMVL)*, pages 111–116, 2023. URL: https://api.semanticscholar.org/CorpusID:257255037, doi:10.48550/arXiv.2303.00322.

30 Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded kleene algebra with tests: verification of uninterpreted programs in nearly linear time. *Proceedings of the ACM on Programming Languages*, 4(POPL):1–28, dec 2019. URL: http://dx.doi.org/10.1145/3371129, doi:10.1145/3371129.

31 Gerco van Heerdt, Clemens Kupke, Jurriaan Rot, and Alexandra Silva. Learning weighted automata over principal ideal domains. *CoRR*, abs/1911.04404, 2019. URL: http://arxiv.org/abs/1911.04404, arXiv:1911.04404, doi:10.48550/arXiv.1911.04404.

32 Friedrich Wehrung. *Refinement Monoids, Equidecomposability Types, and Boolean Inverse Semigroups*, volume 2188 of *Lecture Notes in Mathematics.* Springer International Publishing, Cham, 1st edition 2017 edition, 2017. doi:10.1007/978-3-319-61599-8.

33 Thorsten Wißmann, Ulrich Dorsch, Stefan Milius, and Lutz Schröder. Efficient and modular coalgebraic partition refinement. *Logical Methods in Computer Science*, 16, 2020. doi:10.23638/LMCS-16(1:8)2020.

A Coalgebra

While we strove to use the language of automata in the body of this paper, the underlying arguments rely heavily upon the idea of universal coalgebra [24]. This idea allows us to discuss transition systems in a principled and simplified manner. Universal Coalgebra is based on the language of category theory, which we will assume the reader has a brief awareness of. We will recap some relevant category theoretic ideas, but otherwise refer the reader to [1].

▶ **Definition A.1** ([24]). *Given a functor F, an F-coalgebra is a pair (X, β) where X is a set, and $\beta : X \to FX$. The set X is called the carrier, or set of states, of the system and the function β is called the transition function.*

▶ **Definition A.2.** *Let $(X, \beta), (Y, \gamma)$ be F-coalgebras. A function $f : X \to Y$ is an F-coalgebra homomorphism if $F(f) \circ \beta = \gamma \circ f$. That is, if the following diagram commutes:*

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "\beta"] & Y \arrow[d, "\gamma"] \\ FX \arrow[r, "Ff"] & FY \end{tikzcd}$$

Bisimulation is also a coalgebraic notion.

▶ **Definition A.3** ([24]). *Let (X, β) and (Y, γ) be F-coalgebras. A relation $R \subseteq X \times Y$ is a* bisimulation *if there exists a function $\rho : R \to FR$ such that the projections $\pi_1 : R \to X, \pi_2 : R \to Y$ are F-coalgebra homomorphisms from (R, ρ) to (X, β) and (Y, γ) respectively. We say the elements $x \in X, y \in Y$ are* bisimilar *if there exists a bisimulation R such that $(x, y) \in R$.*

Many of the arguments regarding bisimulation made require that the coalgebra functor F preserve weak pullbacks [24]. In particular we will need this property in order to establish the existence of a bisimulation.

▶ **Definition A.4** ([1],[15]). *In a category C, given morphisms f and g with $cod(f) = cod(g)$, the* pullback *of f and g is the unique object P and morphisms p_1, p_2 as in the following diagram, such that $fp_1 = gp_2$.*

$$\begin{tikzcd} P \arrow[r, "p_2"'] \arrow[d, "p_1"] & A \arrow[d, "g"'] \\ B \arrow[r, "f"] & C \end{tikzcd}$$

Furthermore the pullback is universal with this property, that is, given any $z_1 : Z \to A$ and $z_2 : Z \to B$ with $fz_1 = gz_2$ there exists a unique $u : Z \to P$ such that $z_1 = p_1u$ and $z_2 = p_2u$. If the mapping u is not necessarily unique, then it is called a weak *pullback.*

▶ **Definition A.5** ([16]). *A functor F preserves weak pullbacks if it transforms weak pullback diagrams into weak pullback diagrams.*

B Proofs for Section 3 (Operational Semantics)

$\mathcal{M}_\omega$ is a functor [5] that takes sets and functions to finite weightings and morphisms between weightings. Furthermore $\mathcal{M}_\omega(X)$ is an $\mathbb{S}$ *semimodule* when we have pointwise addition and pointwise scalar multiplication [31]. Where a semimodule is defined by:

▶ **Definition B.1** ([11])**.** *Let $\mathbb{S}$ be a semiring. A (left) $\mathbb{S}$ semimodule is a commutative monoid $(M, +)$ with identity 0_M, for which we have a function $\cdot : \mathbb{S} \times M \to M$ called scalar multiplication where for all $s, r \in \mathbb{S}, n, m \in M$*

$$s \cdot 0_M = 0_M \qquad 0 \cdot m = 0_M \qquad 1 \cdot m = m$$
$$s \cdot (m + n) = s \cdot m + s \cdot n \qquad (s + r) \cdot m = s \cdot m + r \cdot m \qquad (sr) \cdot m = s \cdot (r \cdot m)$$

We require that semimodules have commutative addition, and the result that $\mathcal{M}_\omega(X)$ is a semimodule does not [31]. Clearly the commutativity of addition in $\mathbb{S}$ lifts to component-wise addition of our formal sums.

▶ **Definition B.2** ([10])**.** *Let $\mathbb{S}$ be a semiring and let M, M' be $\mathbb{S}$ semimodules with $m, m' \in M, r \in \mathbb{S}$, then a function $\alpha : M \to M'$ is a homomorphism if and only if both of the following are true*

$$\alpha(m + m') = \alpha(m) + \alpha(m')$$
$$\alpha(rm) = r\alpha(m)$$

We will model our wGKAT automata as coalgebras for the functor $\mathcal{H} = \mathcal{M}_\omega(2 + \mathbf{Out} + \mathbf{Act} \times \mathrm{Id})^{\mathrm{At}}$ when we reason about it.

▶ **Definition B.3.** *An $\mathcal{H}$-coalgebra is a tuple (X, β) where X is a set of states, and β is a function $\beta : X \to \mathcal{M}_\omega(2 + \mathbf{Out} + \mathbf{Act} \times X)^{\mathrm{At}}$.*

$\langle e \rangle_\partial$ is the $\mathcal{H}$ subcoalgebra generated by e. This corresponds to the set of states which are reachable from state e in the automaton.

▶ **Lemma 3.1.** *For all $e \in \mathbf{Exp}$, $\langle e \rangle_\partial$ is finite. We bound it by $\#(e) : \mathbf{Exp} \to \mathbb{N}$ where*

$$\#(b) = 1, \quad \#(v) = 1, \quad \#(p) = 2, \quad \#(e\,{}_r\!\oplus_s f) = \#(e) + \#(f),$$
$$\#(e +_b f) = \#(e) + \#(f), \quad \#(e; f) = \#(e) + \#(f), \quad \#(e^{(b)}) = \#(e)$$

Proof. We adapt the analogous proof for ProbGKAT [26]. We set up the problem and show the cases for our new operators, but do not address the untouched cases. For any $e \in \mathbf{Exp}$ let $|\langle e \rangle_\partial|$ be the cardinality of the least subcoalgebra of $\mathbf{Exp}, \partial$ containing e. The proof that for all $e \in \mathbf{Exp}$ $|\langle e \rangle_\partial| \leq \#(e)$ is by induction.

Assume that $|\langle e \rangle_\partial| \leq \#(e), |\langle f \rangle_\partial| \leq \#(f)$:

$\#(e\,{}_r\!\oplus_s f) = \#(e) + \#(f)$ because every derivative of $e\,{}_r\!\oplus_s f$ is a derivative of e or a derivative of f. Therefore $|\langle e\,{}_r\!\oplus_s f \rangle_\partial| \leq |\langle e \rangle_\partial| + |\langle f \rangle_\partial| \leq \#(e) + \#(f)$.

For sequential composition the behavior of $e; f$ is either termination, a derivative of f or a derivative of e follow by f. $|\langle e; f \rangle_\partial| = |\langle e \rangle_\partial \times \{f\}| + |\langle f \rangle_\partial| \leq \langle e \rangle_\partial + \langle f \rangle_\partial$.

For the guarded loop, note that every derivative of $e^{(b)}$ is a derivative of e composed with a derivative of $e^{(b)}$, in the action case in particular. Clearly then the number of derivatives is no greater than the number of derivatives of e. That is $\left|\left\langle e^{(b)} \right\rangle_\partial\right| \leq |\langle e \rangle_\partial| \leq \#(e)$. ◀

The reason that we need the semiring to be refinement and positive (Definition 3.4) is that these two properties correspond precisely with the functor $\mathcal{M}_\omega$ preserving weak pullbacks.

▶ **Lemma B.1.** *$\mathcal{M}_\omega$ preserves weak pullbacks if and only if $\mathbb{S}$ is positive and refinement.*

Proof. TOPROVE 0 ◀

▶ **Lemma B.2.** *the functor $\mathcal{H}$ preserves weak pullbacks if the semiring over which weightings are taken is positive and refinement.*

Proof. TOPROVE 1 ◀

▶ **Lemma B.3.** *The following semirings are positive refinement monoids.*

1. *Tropical* $(\mathbb{N}^{+\infty}, \min, +, \infty, 0)$
2. *Arctic* $(\mathbb{N}^{+\infty}_{-\infty}, \max, +, -\infty, 0)$
3. *Bottleneck* $(\mathbb{R}^{+\infty}_{-\infty}, \max, \min, -\infty, \infty)$
4. *Formal languages* $(2^{\Gamma^*}, \cup, \cdot, \emptyset, \{\varepsilon\})$
5. *Extended naturals* $(\mathbb{N}^{+\infty}, +, \cdot, 0, 1)$
6. *Viterbi* $([0,1], \max, \cdot, 0, 1)$
7. *Boolean* $(\{0,1\}, \vee, \wedge, 0, 1)$
8. *Extended non-negative rationals* $(\mathbb{Q}^{+\infty}_{\geq 0}, +, \cdot, 0, 1)$

Proof. TOPROVE 2 ◀

▶ **Lemma B.4.** *Each of the following semirings is a Conway semiring.*

1. *Tropical* $(\mathbb{N}^{+\infty}, \min, +, \infty, 0)$
2. *Arctic* $(\mathbb{N}^{+\infty}_{-\infty}, \max, +, -\infty, 0)$
3. *Bottleneck* $(\mathbb{R}^{+\infty}_{-\infty}, \max, \min, -\infty, \infty)$
4. *Formal languages* $(2^{\Gamma^*}, \cup, \cdot, \emptyset, \{\varepsilon\})$
5. *Extended naturals* $(\mathbb{N}^{+\infty}, +, \cdot, 0, 1)$
6. *Viterbi* $([0,1], \max, \cdot, 0, 1)$
7. *Boolean* $(\{0,1\}, \vee, \wedge, 0, 1)$
8. *Extended non-negative rationals* $(\mathbb{Q}^{+\infty}_{\geq 0}, +, \cdot, 0, 1)$

Proof. TOPROVE 3 ◀

▶ **Corollary 3.1.** *The following semirings are positive, Conway, and refinement.*

1. *Tropical* $(\mathbb{N}^{+\infty}, \min, +, \infty, 0)$
2. *Arctic* $(\mathbb{N}^{+\infty}_{-\infty}, \max, +, -\infty, 0)$
3. *Bottleneck* $(\mathbb{R}^{+\infty}_{-\infty}, \max, \min, -\infty, \infty)$
4. *Formal languages* $(2^{\Gamma^*}, \cup, \cdot, \emptyset, \{\varepsilon\})$
5. *Extended naturals* $(\mathbb{N}^{+\infty}, +, \cdot, 0, 1)$
6. *Viterbi* $([0,1], \max, \cdot, 0, 1)$
7. *Boolean* $(\{0,1\}, \vee, \wedge, 0, 1)$
8. *Extended non-negative rationals* $(\mathbb{Q}^{+\infty}_{\geq 0}, +, \cdot, 0, 1)$

Proof. This follows from Lemma B.3 and Lemma B.4. ◀

With these three properties, positive, Conway, and refinement, in hand we can characterize $\mathcal{H}$-coalgebras.

▶ **Proposition B.5.** *There exists an $\mathcal{H}$-coalgebra (Z, ζ) which is final in Coalg$_{\mathcal{H}}$. Equivalently, For any $\mathcal{H}$-coalgebra (X, β) there exists a unique homomorphism $!_\beta : X \to Z$*

Proof. TOPROVE 4 ◀

When it is clear from context we will omit the subscript on the map into the final coalgebra.

▶ **Proposition B.6.** *Let (X, β) and (Y, γ) be $\mathcal{H}$-coalgebras. The elements $x \in X, y \in Y$ are bisimilar iff $!_\beta(x) =!_\gamma(y)$.*

Proof. TOPROVE 5 ◀

The condition $!_\beta(x) =!_\gamma(y)$ is called behavioral equivalence.

C Proofs for Section 4 (Axioms)

▶ **Lemma 4.1.** *Let $e \in \mathbf{Exp}, \alpha \in \mathrm{At}$. It holds that $E(e)_\alpha = \partial(e)_\alpha(\checkmark)$.*

Proof. We prove this by induction of the construction of e. The base cases hold immediately.

$$\begin{aligned} E(e\,{}_r\oplus_s f) &= rE(e)_\alpha + sE(f)_\alpha \\ &= r\partial(e)_\alpha(\checkmark) + s\partial(f)_\alpha(\checkmark) && \text{Inductive hypothesis} \\ &= \partial(e\,{}_r\oplus_s f)_\alpha(\checkmark) \end{aligned}$$

$$\begin{aligned} E(e +_b f) &= \begin{cases} E(e)_\alpha & \alpha \leq_{\mathsf{BA}} b \\ E(f)_\alpha & \alpha \leq_{\mathsf{BA}} \bar{b} \end{cases} \\ &= \begin{cases} \partial(e)_\alpha(\checkmark) & \alpha \leq_{\mathsf{BA}} b \\ \partial(f)_\alpha(\checkmark) & \alpha \leq_{\mathsf{BA}} \bar{b} \end{cases} && \text{Inductive hypothesis} \\ &= \partial(e +_b f)_\alpha(\checkmark) \end{aligned}$$

$$\begin{aligned} E(e;f) &= E(e)_\alpha; E(f)_\alpha \\ &= \partial(e)_\alpha(\checkmark)\partial(f)_\alpha(\checkmark) && \text{Inductive hypothesis} \\ &= \partial(e;f)_\alpha(\checkmark) \end{aligned}$$

$$\begin{aligned} E(e^{(b)}) &= \begin{cases} 1 & \alpha \leq_{\mathsf{BA}} \bar{b} \\ 0 & \alpha \leq_{\mathsf{BA}} b \end{cases} \\ &= \begin{cases} \partial(e^{(b)})_\alpha(\checkmark) & \alpha \leq_{\mathsf{BA}} \bar{b} \\ \partial(e^{(b)})_\alpha(\checkmark) & \alpha \leq_{\mathsf{BA}} b \end{cases} \\ &= \partial(e^{(b)})_\alpha(\checkmark) \end{aligned}$$ ◀

We show that $\equiv$ is a bisimulation equivalence by first showing that any pair $(e, f) \in \equiv$ is behaviorally equivalent. Since $\mathcal{H}$ preserves weak pullbacks this also implies that they are bisimilar by Proposition B.6, making $\equiv$ a bisimulation equivalence.

Our proof of soundness via behavioral equivalence is inspired by process algebra [28]. We will first show that the congruence $\equiv$ is the kernel of a morphism, which will then allow us to express the morphisms to the final coalgebra as composite morphisms of the canonical quotient morphism and the morphism from the quotient coalgebra to the final coalgebra.

First we must introduce some notation.

▶ **Definition C.1** ([26]). *Let $A \subseteq \mathbf{Exp}$ and $f \in \mathbf{Exp}$, then $A/f = \{g \in \mathbf{Exp} \mid g; f \in A\}$.*

Furthermore we modify some lemmas from ProbGKAT about this piece of notation which remain sound with our new model. Their proofs are unchanged in the new model, so we omit them.

▶ **Lemma C.1** (Lemma 41, [26]). *If $R \subseteq \mathbf{Exp} \times \mathbf{Exp}$ is a congruence relation with respect to* wGKAT *operators and $(e, f) \in R$ then $R(A/e) \subseteq R(A)/f$.*

▶ **Lemma C.2** (Lemma 42, [26]). *For all $\alpha \in \mathrm{At}$, $r, f \in \mathbf{Exp}$, $p \in \mathbf{Act}$, $A \subseteq \mathbf{Exp}$*

$$\partial(e;f)_\alpha[\{p\} \times A] = \partial(e)_\alpha[\{p\} \times A/f] + \partial(e)_\alpha(\checkmark)\partial(f)_\alpha[\{p\} \times A]$$

▶ **Lemma C.3** (Lemma 45, [26])**.** *Let* $R \subseteq \mathbf{Exp} \times \mathbf{Exp}$ *be a congruence with respect to* wGKAT *operators such that* $(e, f) \in R$*, and let* $Q \in \mathbf{Exp}/R$ *be an equivalence class of* R*. It holds that* $Q/e = Q/f$*.*

▶ **Lemma C.4** (Lemma 46, [26])**.** *Let* $e, f \in \mathbf{Exp}$ *and let* $Q \in \mathbf{Exp}/\equiv$*. Then* $(Q/f)/e = Q/e; f$*.*

▶ **Lemma C.5** (Lemma 48, [26])**.** *Let* $e \in \mathbf{Exp}$ *and* $b \in \mathbf{BExp}$*. If* $a \leq_{BA} b$*, then* $\partial(b; e)_\alpha = \partial(e)_\alpha$*.*

Before proving soundness we recall two relevant properties of Conway semirings.

▶ **Lemma C.6** ([8])**.** *If* $\mathbb{S}$ *is a Conway semiring, the following equation holds in* $\mathbb{S}$

$$a^* = 1 + aa^*$$

▶ **Lemma C.7** ([8])**.** *If* $\mathbb{S}$ *is a Conway semiring, the following equation holds in* $\mathbb{S}$

$$(ab)^*a = a(ba)^*$$

While Conway semirings are not the most general axiomatization of a semiring with a * operator, we need the * operator to obey Equation (1) in Definition 3.1, and Lemma C.7 in addition to the fixpoint rule, Lemma C.6. We note that 2 in Definition 3.1 is easily derivable from Lemma C.7 and the fixpoint rule so this class of semiring is not too strong.

▶ **Lemma C.8.** *If a semiring has a total operator* $^* : R \to R$ *with the properties* $\forall a, b \;\; (ab)^*a = a(ba)^*$ *and* $\forall a \;\; a^* = 1 + aa^*$*, then* $\forall a, b \;\; (ab)^* = 1 + a(ba)^*b$*.*

Proof. TOPROVE 6 ◀

Now we can prove the existence of the necessary homomorphism.

▶ **Lemma C.9.** $\equiv$ *is the kernel of a coalgebra homomorphism.*

Proof. TOPROVE 7 ◀

With this in hand, proving soundness with respect to behavioral equivalence is straightforward.

▶ **Theorem C.10.** $e \equiv f \implies !_\partial e = !_\partial f$

Proof. TOPROVE 8 ◀

▶ **Corollary 4.1** (Soundness)**.** *For all* $e, f \in \mathbf{Exp}$ $e \equiv f \implies e \sim f$

Proof. $e \equiv f \implies !e = !f$ by Theorem C.10. Furthermore, $!e = !f \implies e \sim f$ by Proposition B.6. The result is immediate. ◀

▶ **Lemma 4.2.** *For all* $b, c \in \mathbf{BExp}$*,* $e, f, g, h \in \mathbf{Exp}$*,* $r, s, t, u \in \mathbb{S}$

DF1	$\odot t; (e \,{}_{r}\!\oplus_{s} f) \equiv e \,{}_{tr}\!\oplus_{ts} f$	DF8	$b; (e +_c f) \equiv b; e +_c b; f$
DF2	$e +_b (f +_c g) \equiv (e +_b f) +_{b+c} g$	DF9	$b; (e +_c f) \equiv b; (b; e +_c f)$
DF3	$e +_b \mathbb{0} \equiv b; e$	DF10	$\odot r; \odot s \equiv \odot(rs)$
DF4	$b; (e +_b f) \equiv b; e$	DF11	$\odot r; (e +_b f) \equiv \odot r; e +_b \odot r; f$
DF5	$(e +_b f) \,{}_{r}\!\oplus_{s} g \equiv (e \,{}_{r}\!\oplus_{s} g) +_b (f \,{}_{r}\!\oplus_{s} g)$	DF12	$g \,{}_{r}\!\oplus_{s} \odot 0 \equiv \odot r; g$
DF6	$(e +_b f) \,{}_{r}\!\oplus_{s} (g +_b h) \equiv (e \,{}_{r}\!\oplus_{s} g) +_b (f \,{}_{r}\!\oplus_{s} h)$	DF13	$b; (e \,{}_{r}\!\oplus_{s} f) \equiv b; (b; e \,{}_{r}\!\oplus_{s} f)$
DF7	$e +_{\mathbb{1}} f \equiv e$	DF14	$g \,{}_{s}\!\oplus_{t} (e \,{}_{r}\!\oplus_{u} \odot 0) \equiv g \,{}_{s}\!\oplus_{tr} e$

Proof. We now prove each of these derived facts. DF2-DF4, DF8 retain the same proof as GKAT, so we simply cite GKAT [30] and do not restate the proof.

DF1

$$
\begin{aligned}
\odot t; (e \,{}_{r}{\oplus}_{s}\, f) &\equiv (\mathbb{1} \,{}_{t}{\oplus}_{0}\, \mathbb{0}); (e \,{}_{r}{\oplus}_{s}\, f) && \\
&\equiv \mathbb{1}; (e \,{}_{r}{\oplus}_{s}\, f) \,{}_{t}{\oplus}_{0}\, \mathbb{0}; (e \,{}_{r}{\oplus}_{s}\, f) && \text{S4} \\
&\equiv \mathbb{0}; (e \,{}_{r}{\oplus}_{s}\, f) \,{}_{0}{\oplus}_{t}\, \mathbb{1}; (e \,{}_{r}{\oplus}_{s}\, f) && \text{W2} \\
&\equiv \mathbb{0}; (e \,{}_{r}{\oplus}_{s}\, f) \,{}_{0}{\oplus}_{t}\, (e \,{}_{r}{\oplus}_{s}\, f) && \text{S1} \\
&\equiv \mathbb{0}; (e \,{}_{r}{\oplus}_{s}\, f) \,{}_{0}{\oplus}_{1}\, (e \,{}_{tr}{\oplus}_{ts}\, f) && \text{D2} \\
&\equiv (e \,{}_{tr}{\oplus}_{ts}\, f) \,{}_{1}{\oplus}_{0}\, \mathbb{0}; (e \,{}_{r}{\oplus}_{s}\, f) && \text{W2} \\
&\equiv (e \,{}_{tr}{\oplus}_{ts}\, f) \,{}_{1}{\oplus}_{0}\, \mathbb{0}; (e \,{}_{tr}{\oplus}_{ts}\, f) && \text{S3} \\
&\equiv \mathbb{1}; (e \,{}_{tr}{\oplus}_{ts}\, f) \,{}_{1}{\oplus}_{0}\, \mathbb{0}; (e \,{}_{tr}{\oplus}_{ts}\, f) && \text{S1} \\
&\equiv (\mathbb{1} \,{}_{1}{\oplus}_{0}\, \mathbb{0}); (e \,{}_{tr}{\oplus}_{ts}\, f) && \text{S4} \\
&\equiv \odot 1; (e \,{}_{tr}{\oplus}_{ts}\, f) && \\
&\equiv e \,{}_{tr}{\oplus}_{ts}\, f && \text{C1, S1}
\end{aligned}
$$

DF5

$$
\begin{aligned}
(e +_b f) \,{}_{r}{\oplus}_{s}\, g &\equiv g \,{}_{s}{\oplus}_{r}\, (e +_b f) && \text{W2} \\
&\equiv (g \,{}_{s}{\oplus}_{r}\, e +_b g \,{}_{s}{\oplus}_{r}\, f) && \text{D1} \\
&\equiv (e \,{}_{r}{\oplus}_{s}\, g +_b f \,{}_{r}{\oplus}_{s}\, g) && \text{W2}
\end{aligned}
$$

DF6

$$
\begin{aligned}
(e +_b f) \,{}_{r}{\oplus}_{s}\, (g +_b h) &\equiv ((e +_b f) \,{}_{r}{\oplus}_{s}\, g) +_b ((e +_b f) \,{}_{r}{\oplus}_{s}\, h) && \text{D1} \\
&\equiv ((e \,{}_{r}{\oplus}_{s}\, g) +_b (f \,{}_{r}{\oplus}_{s}\, g)) +_b ((e +_b f) \,{}_{r}{\oplus}_{s}\, h) && \text{DF5} \\
&\equiv b; ((e \,{}_{r}{\oplus}_{s}\, g) +_b (f \,{}_{r}{\oplus}_{s}\, g)) +_b ((e +_b f) \,{}_{r}{\oplus}_{s}\, h) && \text{G2} \\
&\equiv (b; (e \,{}_{r}{\oplus}_{s}\, g)) +_b ((e +_b f) \,{}_{r}{\oplus}_{s}\, h) && \text{DF4} \\
&\equiv (e \,{}_{r}{\oplus}_{s}\, g) +_b ((e +_b f) \,{}_{r}{\oplus}_{s}\, h) && \text{G2} \\
&\equiv ((f +_{\bar{b}} e) \,{}_{r}{\oplus}_{s}\, h) +_{\bar{b}} (e \,{}_{r}{\oplus}_{s}\, g) && \text{G3} \\
&\equiv ((f \,{}_{r}{\oplus}_{s}\, h) +_{\bar{b}} (e \,{}_{r}{\oplus}_{s}\, h)) +_{\bar{b}} (e \,{}_{r}{\oplus}_{s}\, g) && \text{DF5} \\
&\equiv \bar{b}((f \,{}_{r}{\oplus}_{s}\, h) +_{\bar{b}} (e \,{}_{r}{\oplus}_{s}\, h)) +_{\bar{b}} (e \,{}_{r}{\oplus}_{s}\, g) && \text{G2} \\
&\equiv \bar{b}(f \,{}_{r}{\oplus}_{s}\, h) +_{\bar{b}} (e \,{}_{r}{\oplus}_{s}\, g) && \text{DF4} \\
&\equiv (f \,{}_{r}{\oplus}_{s}\, h) +_{\bar{b}} (e \,{}_{r}{\oplus}_{s}\, g) && \text{G2} \\
&\equiv (e \,{}_{r}{\oplus}_{s}\, g) +_b (f \,{}_{r}{\oplus}_{s}\, h) && \text{G3}
\end{aligned}
$$

DF7

$$
\begin{aligned}
e +_{\mathbb{1}} f &\equiv \mathbb{1}; (e +_{\mathbb{1}} f) && \text{S1} \\
&\equiv \mathbb{1}; e && \text{DF4} \\
&\equiv e && \text{S1}
\end{aligned}
$$

DF9

$$
b; (e +_c f) \equiv b; e +_c b; f \qquad \text{DF8}
$$

$$\begin{aligned}
&\equiv bb; e +_c b; f && bb \equiv_{\mathsf{BA}} b\\
&\equiv b; b; e +_c b; f && \mathsf{S7}\\
&\equiv b; (b; e +_c f) && \mathsf{DF8}
\end{aligned}$$

DF10

$$\begin{aligned}
\odot r; \odot s &\equiv \mathbb{1}\,{}_{r}{\oplus}_{0}\,\mathbb{0}; \mathbb{1}\,{}_{s}{\oplus}_{0}\,\mathbb{0}\\
&\equiv \mathbb{1}; (\mathbb{1}\,{}_{s}{\oplus}_{0}\,\mathbb{0})\,{}_{r}{\oplus}_{0}\,\mathbb{0}; (\mathbb{1}\,{}_{s}{\oplus}_{0}\,\mathbb{0}) && \mathsf{S4}\\
&\equiv (\mathbb{1}\,{}_{s}{\oplus}_{0}\,\mathbb{0})\,{}_{r}{\oplus}_{0}\,\mathbb{0} && \mathsf{S1, S3}\\
&\equiv \mathbb{0}\,{}_{0}{\oplus}_{r}\,(\mathbb{1}\,{}_{s}{\oplus}_{0}\,\mathbb{0}) && \mathsf{W2}\\
&\equiv \mathbb{0}\,{}_{0}{\oplus}_{1}\,(\mathbb{1}\,{}_{rs}{\oplus}_{0}\,\mathbb{0}) && \mathsf{D2}\\
&\equiv \mathbb{1}; (\mathbb{1}\,{}_{rs}{\oplus}_{0}\,\mathbb{0})\,{}_{1}{\oplus}_{0}\,\mathbb{0}; (\mathbb{1}\,{}_{rs}{\oplus}_{0}\,\mathbb{0}) && \mathsf{W2, S3, S1}\\
&\equiv \odot 1; (\mathbb{1}\,{}_{rs}{\oplus}_{0}\,\mathbb{0}) && \mathsf{S4}\\
&\equiv \mathbb{1}\,{}_{rs}{\oplus}_{0}\,\mathbb{0} && \mathsf{C1, S1}\\
&\equiv \odot(rs)
\end{aligned}$$

DF11

$$\begin{aligned}
\odot r; (e +_b f) &\equiv (\mathbb{1}\,{}_{r}{\oplus}_{0}\,\mathbb{0}); (e +_b f)\\
&\equiv (\mathbb{1}; (e +_b f)\,{}_{r}{\oplus}_{0}\,\mathbb{0}; (e +_b f)) && \mathsf{S4}\\
&\equiv (e +_b f)\,{}_{r}{\oplus}_{0}\,\mathbb{0} && \mathsf{S1, S3}\\
&\equiv \mathbb{0}\,{}_{0}{\oplus}_{r}\,(e +_b f) && \mathsf{W2}\\
&\equiv (\mathbb{0}\,{}_{0}{\oplus}_{r}\,e) +_b (\mathbb{0}\,{}_{0}{\oplus}_{r}\,f) && \mathsf{D1}\\
&\equiv (e\,{}_{r}{\oplus}_{0}\,\mathbb{0}) +_b (f\,{}_{r}{\oplus}_{0}\,\mathbb{0}) && \mathsf{W2}\\
&\equiv (\mathbb{1}; e\,{}_{r}{\oplus}_{0}\,\mathbb{0}) +_b (\mathbb{1}; f\,{}_{r}{\oplus}_{0}\,\mathbb{0}) && \mathsf{S1}\\
&\equiv (\mathbb{1}; e\,{}_{r}{\oplus}_{0}\,\mathbb{0}; e) +_b (\mathbb{1}; f\,{}_{r}{\oplus}_{0}\,\mathbb{0}; f) && \mathsf{S3}\\
&\equiv (\mathbb{1}\,{}_{r}{\oplus}_{0}\,\mathbb{0}); e +_b (\mathbb{1}\,{}_{r}{\oplus}_{0}\,\mathbb{0}); f && \mathsf{S4}\\
&\equiv \odot r; e +_b \odot r; f
\end{aligned}$$

DF12

$$\begin{aligned}
g\,{}_{r}{\oplus}_{s}\,\odot 0 &\equiv g\,{}_{r}{\oplus}_{s}\,\odot 0; \mathbb{0} && \mathsf{C2}\\
&\equiv g\,{}_{r}{\oplus}_{0}\,\mathbb{0} && \mathsf{W2, W4}\\
&\equiv \mathbb{1}; g\,{}_{r}{\oplus}_{0}\,\mathbb{0}; g && \mathsf{S1, S3}\\
&\equiv (\mathbb{1}\,{}_{r}{\oplus}_{0}\,\mathbb{0})\,; g && \mathsf{S4}\\
&\equiv \odot r; g
\end{aligned}$$

DF13

$$\begin{aligned}
b; (e\,{}_{r}{\oplus}_{s}\,f) &\equiv b; (b; e\,{}_{r}{\oplus}_{s}\,b; f) && \mathsf{D3}\\
&\equiv b; ((bb); e\,{}_{r}{\oplus}_{s}\,b; f) && \mathrm{BA}\\
&\equiv b; (b; b; e\,{}_{r}{\oplus}_{s}\,b; f) && \mathsf{S7}\\
&\equiv b; (b; e\,{}_{r}{\oplus}_{s}\,f) && \mathsf{D3}
\end{aligned}$$

DF14

$$\begin{aligned}
g\,{}_{s}{\oplus}_{t}\,(e\,{}_{r}{\oplus}_{u}\,\odot 0) &\equiv g\,{}_{s}{\oplus}_{t}\,(\odot r; e) && \mathsf{DF12}
\end{aligned}$$

$$\equiv (\odot r; e)\ {}_{t}\oplus_{s}\ g \qquad \text{W2}$$
$$\equiv e\ {}_{tr}\oplus_{s}\ g \qquad \text{W4} \quad ◀$$

D Proofs for Section 5 (Completeness)

▶ **Lemma D.1** ([26], 52). *Generalized guarded sums are well-defined up to* $\equiv$

▶ **Lemma D.2** ([26], 53). *Let $b, c \in$ **BExp** and let $\{e_\alpha\}_{\alpha\in\mathrm{At}}$ be an indexed collection such that $\forall\alpha\ \ e_\alpha \in$ **Exp**. Then:*

$$c; \mathop{+}_{\alpha\leq_{BA}b} e_\alpha \equiv \mathop{+}_{\alpha\leq_{BA}bc}$$

Where we take $\alpha \leq_{\mathsf{BA}} b$ in this case to actually be the set $\{\alpha \mid \alpha \leq_{\mathsf{BA}} b\}$ similar to [30, 26]

▶ **Lemma D.3** ([26], 54). *Let $\Phi \subseteq \mathrm{At}$ and $\{e_\alpha\}_{\alpha\in\mathrm{At}}$ be an indexed collection such that $\forall\alpha\ \ e_\alpha \in$ **Exp**. Then:*

$$\mathop{+}_{\alpha\in\Phi} e_\alpha \equiv \mathop{+}_{\alpha\in\Phi} \alpha; e_\alpha$$

▶ **Lemma D.4** ([26], 55). *Let $\Phi \subseteq \mathrm{At}$ and $\{e_\alpha\}_{\alpha\in\mathrm{At}}$ be an indexed collection such that $\forall\alpha\ \ e_\alpha \in$ **Exp** and let $f \in$ **Exp**. Then:*

$$\left(\mathop{+}_{\alpha\in\Phi} e_\alpha\right); f \equiv \mathop{+}_{\alpha\in\Phi} e_\alpha; f$$

▶ **Lemma D.5.** *Let $\Phi \subseteq \mathrm{At}$ and let $\{e_\alpha\}_{\alpha\in\Phi}$ and $\{f_\alpha\}_{\alpha\in\Phi}$ be indexed collections such that $e_\alpha, f_\alpha \in$ **Exp** for each $\alpha \in \Phi$ and let $r, s \in \mathbb{S}$. Then:*

$$\left(\mathop{+}_{\alpha\in\Phi} e_\alpha\right)\ {}_{r}\oplus_{s} \left(\mathop{+}_{\alpha\in\Phi} f_\alpha\right) \equiv \mathop{+}_{\alpha\in\Phi} (e_\alpha\ {}_{r}\oplus_{s}\ f_\alpha)$$

Proof. TOPROVE 9 ◀

We note that by DF14 we can omit the final branch of the generalized weighted sum which has an empty index set and thus is the zero weighting. We sometimes opt to use DF12 and express the sum with the scaling notation $\odot r; e$ rather than $e\ {}_{r}\oplus_{1}\ \odot 0$.

▶ **Lemma D.6.** *Generalized weighted sums are well defined up to* $\equiv$

Proof. TOPROVE 10 ◀

▶ **Remark D.1.** *We note here that we will sometimes write*

$$\bigoplus_{i\in I} r_i \cdot e_i \oplus \bigoplus_{j\in J} s_j \cdot f_j$$

for the generalized weighted sum

$$\bigoplus_{i\in I\cup J} t_k \cdot g_k \quad t_k = \begin{cases} r_k & k \in I \\ s_k & k \in J \end{cases}, \quad g_k = \begin{cases} e_k & k \in I \\ f_k & k \in J \end{cases}$$

where I and J are finite and nonempty and $I \cap J = \emptyset$. This is useful to separate parts of a generalized weighted sum.

▶ **Lemma D.7.** *Let I be a non-empty finite index set, $\{r_i\}_{i\in I}$ and $\{e_i\}_{i\in I}$ indexed collections such that $r_i \in \mathbb{S}$ and $e_i \in \mathbf{Exp}$ for all i and $f \in \mathbf{Exp}$ then:*

$$\left(\bigoplus_{i\in I} r_i \cdot e_i\right); f \equiv \left(\bigoplus_{i\in I} r_i \cdot e_i; f\right)$$

Proof. TOPROVE 11 ◀

▶ **Lemma D.8.** *Let I be a non-empty finite index set, $\{r_i\}_{i\in I}$ and $\{e_i\}_{i\in I}$ indexed collections such that $r_i \in \mathbb{S}$ and $e_i \in \mathbf{Exp}$ for all $i \in I$. Let $E = \bigcup_{i\in I}\{e_i\}$. Then:*

$$\bigoplus_{i\in I} r_i \cdot e_i \equiv \bigoplus_{e\in E} \left(\sum_{e_i = e} r_i\right) \cdot e$$

Proof. TOPROVE 12 ◀

▶ **Lemma D.9.** *Let I be a non-empty finite index set, $\{r_i\}_{i\in I}$ and $\{e_i\}_{i\in I}$ indexed collections such that $r_i \in S$ and $e_i \in \mathbf{Exp}$ for all $i \in I$. Let $E = \bigcup_{i\in I}\{e_i\}$ then:*

$$\bigoplus_{i\in I} r_i \cdot e_i \equiv \bigoplus_{[e]_\equiv \in E/\equiv} \left(\sum_{e_i \equiv e} r_i\right) \cdot e$$

Proof. TOPROVE 13 ◀

▶ **Theorem 5.1** (Fundamental Theorem)**.** *For every $e \in \mathbf{Exp}$ it holds that*

$$e \equiv \bigplus_{\alpha\in \mathrm{At}} \left(\bigoplus_{d\in supp(\partial(e)_\alpha)} \partial(e)_\alpha(d) \cdot \exp(d)\right)$$

where exp defines a function $2 + \mathbf{Out} + \mathbf{Act} \times \mathbf{Exp} \to \mathbf{Exp}$ given by

$$\exp(✗) = \mathbb{0} \quad \exp(✓) = \mathbb{1} \quad \exp(v) = v \quad \exp((p, f)) = p; f \quad (v \in \mathbf{Out}, p \in \mathbf{Act}, f \in \mathbf{Exp})$$

Proof. The proof is by induction on the construction of e. The base cases are trivial, we illustrate one of them. We also omit the case of guarded choice as it is unchanged from ProbGKAT [26].

Case of: $e \equiv p$

$$\begin{aligned} p &\equiv \bigplus_{\alpha\in\mathrm{At}} p && \text{G1} \\ &\equiv \bigplus_{\alpha\in\mathrm{At}} p; \mathbb{1} && \text{S1} \\ &\equiv \bigplus_{\alpha\in\mathrm{At}} \left(\bigoplus_{d\in \mathrm{supp}(\partial(e)_\alpha)} \partial(e)_\alpha(d) \cdot \exp(d)\right) \end{aligned}$$

Case of: $e \equiv f \,{}_r{\oplus}_s\, g$

$$\begin{aligned}
f \,{}_r{\oplus}_s\, g &\equiv \mathop{\Large +}_{\alpha\in\mathrm{At}} \left(\bigoplus_{d\in\mathrm{supp}(\partial(f)_\alpha)} \partial(f)_\alpha(d)\cdot \exp(d) \right) \\
&\quad {}_r{\oplus}_s \mathop{\Large +}_{\alpha\in\mathrm{At}} \left(\bigoplus_{d\in\mathrm{supp}(\partial(g)_\alpha)} \partial(g)_\alpha(d)\cdot \exp(d) \right) && \text{Ind. hypothesis} \\
&\equiv \mathop{\Large +}_{\alpha\in\mathrm{At}} \left(\bigoplus_{d\in\mathrm{supp}(\partial(f)_\alpha)} \partial(f)_\alpha(d)\cdot \exp(d) \right. \\
&\quad \left. {}_r{\oplus}_s \bigoplus_{d\in\mathrm{supp}(\partial(g)_\alpha)} \partial(g)_\alpha(d)\cdot \exp(d) \right) && \textit{Lemma D.5}
\end{aligned}$$

For each α we can rearrange this inner expression using W3 into a single generalized sum. Furthermore, let $I_\alpha \coloneqq \mathrm{supp}(\partial(f)_\alpha) + \mathrm{supp}(\partial(g)_\alpha)$, where $+$ is the coproduct. Then let

$$r_i = \begin{cases} r\partial(f)_\alpha(i) & i\in\mathrm{supp}(\partial(f)_\alpha) \\ s\partial(g)_\alpha(i) & i\in\mathrm{supp}(\partial(s)_\alpha) \end{cases}$$

Hence we can define

$$f \,{}_r{\oplus}_s\, g = \mathop{\Large +}_{\alpha\in\mathrm{At}} \left(\bigoplus_{i\in I_\alpha} r_i\cdot\exp(i) \right)$$

So by Lemma D.8 we can combine terms which gives us

$$\mathop{\Large +}_{\alpha\in\mathrm{At}} \left(\bigoplus_{d\in\mathrm{supp}(\partial(f)_\alpha)\cup\mathrm{supp}(\partial(g)_\alpha)} (r\partial(f)_\alpha(d) + s\partial(g)_\alpha(d))\cdot\exp(\mathrm{d}) \right)$$

Which is the same as

$$\mathop{\Large +}_{\alpha\in\mathrm{At}} \left(\bigoplus_{d\in\mathrm{supp}(\partial(f \,{}_r{\oplus}_s\, g)_\alpha)} \partial(f \,{}_r{\oplus}_s\, g)_\alpha(d)\cdot\exp(\mathrm{d}) \right)$$

Case of: $e = f; g$

$$\begin{aligned}
f; g &\equiv \left(\mathop{\Large +}_{\alpha\in\mathrm{At}} \left(\bigoplus_{d\in\mathrm{supp}(\partial(f)_\alpha)} \partial(f)_\alpha(d)\cdot\exp(d) \right) \right); g && \text{Inductive hypothesis} \\
&\equiv \left(\mathop{\Large +}_{\alpha\in\mathrm{At}} \left(\bigoplus_{d\in\mathrm{supp}(\partial(f)_\alpha)} \partial(f)_\alpha(d)\cdot\exp(d) \right); g \right) && \textit{Lemma D.4} \\
&\equiv \left(\mathop{\Large +}_{\alpha\in\mathrm{At}} \left(\bigoplus_{d\in\mathrm{supp}(\partial(f)_\alpha)} \partial(f)_\alpha(d)\cdot\exp(d); g \right) \right) && \textit{Lemma D.7}
\end{aligned}$$

We split this term to get

$$\mathop{\Large +}_{\alpha\in\mathrm{At}} \left(\partial(f)_\alpha(\checkmark)\cdot\mathbb{1}; g \oplus \bigoplus_{d\in\{\text{✗}\}\cup\mathbf{Out}} \partial(f)_\alpha(d)\cdot\exp(d); g \right.$$

$$\oplus \bigoplus_{d \in \mathrm{supp}(\partial(f)_\alpha) \cap \mathbf{Act} \times \mathbf{Exp}} \partial(f)_\alpha(d) \cdot \exp(d); g \Bigg)$$

$$\equiv \mathop{\Large +}_{\alpha \in \mathrm{At}} \Bigg(\partial(f)_\alpha(✓) \cdot g \oplus \bigoplus_{d \in \{✗\} \cup \mathbf{Out}} \partial(f)_\alpha(d) \cdot \exp(d); g$$

$$\oplus \bigoplus_{d \in \mathrm{supp}(\partial(f)_\alpha) \cap \mathbf{Act} \times \mathbf{Exp}} \partial(f)_\alpha(d) \cdot \exp(d); g \Bigg) \qquad \mathsf{S1}$$

$$\equiv \mathop{\Large +}_{\alpha \in \mathrm{At}} \Bigg(\partial(f)_\alpha(✓) \cdot g \oplus \bigoplus_{d \in \{✗\} \cup \mathbf{Out}} \partial(f)_\alpha(d) \cdot \exp(d)$$

$$\oplus \bigoplus_{d \in \mathrm{supp}(\partial(f)_\alpha) \cap \mathbf{Act} \times \mathbf{Exp}} \partial(f)_\alpha(d) \cdot \exp(d); g \Bigg) \qquad \mathsf{S3, S6}$$

$$\equiv \mathop{\Large +}_{\alpha \in \mathrm{At}} \alpha; \Bigg(\partial(f)_\alpha(✓) \cdot g \oplus \bigoplus_{d \in \{✗\} \cup \mathbf{Out}} \partial(f)_\alpha(d) \cdot \exp(d)$$

$$\oplus \bigoplus_{d \in \mathrm{supp}(\partial(f)_\alpha) \cap \mathbf{Act} \times \mathbf{Exp}} \partial(f)_\alpha(d) \cdot \exp(d); g \Bigg) \qquad \mathsf{G2}$$

$$\equiv \mathop{\Large +}_{\alpha \in \mathrm{At}} \alpha; \Bigg(\partial(f)_\alpha(✓) \cdot \alpha; g \oplus \bigoplus_{d \in \{✗\} \cup \mathbf{Out}} \partial(f)_\alpha(d) \cdot \exp(d)$$

$$\oplus \bigoplus_{d \in \mathrm{supp}(\partial(f)_\alpha) \cap \mathbf{Act} \times \mathbf{Exp}} \partial(f)_\alpha(d) \cdot \exp(d); g \Bigg) \qquad \mathsf{DF13}$$

$$\equiv \mathop{\Large +}_{\alpha \in \mathrm{At}} \Bigg(\partial(f)_\alpha(✓) \cdot \alpha; g \oplus \bigoplus_{d \in \{✗\} \cup \mathbf{Out}} \partial(f)_\alpha(d) \cdot \exp(d)$$

$$\oplus \bigoplus_{d \in \mathrm{supp}(\partial(f)_\alpha) \cap \mathbf{Act} \times \mathbf{Exp}} \partial(f)_\alpha(d) \cdot \exp(d); g \Bigg) \qquad \mathsf{G2}$$

We can apply the inductive hypothesis and DF4 to say

$$\alpha; g = \alpha; \bigoplus_{d \in \mathrm{supp}(\partial(g)_\alpha)} \partial(g)_\alpha(d) \cdot \exp(d)$$

The α will be removed by G2 as soon as we substitute this back into the above expression. We substitute and can express this as one sum. Let

$$I_\alpha = \mathrm{supp}(\partial(g)_\alpha) + (\mathrm{supp}(\partial(f)_\alpha) \cap (\{✗\} \cup \mathbf{Out}))$$
$$+ \{(a, f'; g) \mid (a, f') \in \mathrm{supp}(\partial(f)_\alpha) \cap \mathbf{Act} \times \mathbf{Exp}\}$$

For each $\alpha \in \mathrm{At}$ let $\{r_i\}_{i \in I_\alpha}$ such that

$$r_i = \begin{cases} \partial(f)_\alpha(✓)\partial(g)_\alpha(i) & i \in \mathrm{supp}(\partial(g)_\alpha) \\ \partial(f)_\alpha(i) & i \in \mathrm{supp}(\partial(f)_\alpha \cap (\{✗\} \cup \mathbf{Out})) \\ \partial(f)_\alpha(a, f') & i = (a, f'; g) \text{ and } (a, f') \in \mathrm{supp}(\partial(f)_\alpha \cap \mathbf{Act} \times \mathbf{Exp}) \end{cases}$$

Hence we have

$$f; g \equiv \mathop{\Large +}_{\alpha \in \mathrm{At}} \left(\bigoplus_{i \in I_\alpha} r_i \cdot \exp(i) \right)$$

Finally we apply Lemma D.8 to arrive at

$$f; g \equiv \mathop{\Large +}_{\alpha \in \mathrm{At}} \left(\bigoplus_{d \in \mathrm{supp}(\partial(f;g)_\alpha)} \partial(f;g)_\alpha(d) \cdot \exp(d) \right)$$

Case of: $e \equiv f^{(b)}$

We want to show that

$$f^{(b)} \equiv \mathop{\Large +}_{\alpha \in \mathrm{At}} \left(\bigoplus_{d \in \mathrm{supp}(\partial(f^{(b)})_\alpha)} \partial \left(f^{(b)} \right)_\alpha (d) \cdot \exp(d) \right)$$

Note that due to Lemma D.3 this is equivalent to proving

$$\mathop{\Large +}_{\alpha \in \mathrm{At}} \alpha; f^{(b)} \equiv \mathop{\Large +}_{\alpha \in \mathrm{At}} \alpha; \left(\bigoplus_{d \in \mathrm{supp}(\partial(f^{(b)})_\alpha)} \partial \left(f^{(b)} \right)_\alpha (d) \cdot \exp(d) \right)$$

Which is a consequence of the statement $\forall \alpha \in \mathrm{At}$

$$\alpha; f^{(b)} \equiv \alpha; \left(\bigoplus_{d \in \mathrm{supp}(\partial(f^{(b)})_\alpha)} \partial \left(f^{(b)} \right)_\alpha (d) \cdot \exp(d) \right)$$

We first consider an atom $\gamma \leq_{\mathsf{BA}} \bar{b}$. It holds that

$$\begin{aligned}
\gamma; f^{(b)} &\equiv \gamma; (f; f^{(b)} +_b \mathbb{1}) && \mathsf{L1} \\
&\equiv (f; f^{(b)} +_b \mathbb{1}) +_\gamma \mathbb{0} && \mathsf{DF3} \\
&\equiv f; f^{(b)} +_{b\gamma} (\mathbb{1} +_\gamma \mathbb{0}) && \mathsf{G4} \\
&\equiv f; f^{(b)} +_{\mathbb{0}} (\mathbb{1} +_\gamma \mathbb{0}) && \gamma \leq_{\mathsf{BA}} \bar{b} \\
&\equiv (\mathbb{1} +_\gamma \mathbb{0}) +_{\mathbb{1}} f; f^{(b)} && \mathsf{G4} \\
&\equiv (\mathbb{1} +_\gamma \mathbb{0}) && \mathsf{DF7} \\
&\equiv \gamma; \mathbb{1} && \mathsf{DF3} \\
&\equiv \gamma; \left(\bigoplus_{d \in \mathrm{supp}(\partial(f^{(b)})_\gamma)} \partial \left(f^{(b)} \right)_\gamma (d) \cdot \exp(d) \right)
\end{aligned}$$

Consider an atom $\gamma \leq_{\mathsf{BA}} b$.

$$\begin{aligned}
f &\equiv \mathop{\Large +}_{\alpha \in \mathrm{At}} \left(\bigoplus_{d \in \mathrm{supp}(\partial(f)_\alpha)} \partial(f)_\alpha (d) \cdot \exp(d) \right) && \text{Inductive hypothesis} \\
&\equiv \bigoplus_{d \in \mathrm{supp}(\partial(f)_\gamma)} \partial(f)_\gamma (d) \cdot \exp(d) \\
&\quad +_\gamma \mathop{\Large +}_{\alpha \in \mathrm{At} \setminus \gamma} \left(\bigoplus_{d \in \mathrm{supp}(\partial(f)_\alpha)} \partial(f)_\alpha (d) \cdot \exp(d) \right)
\end{aligned}$$

$$
\begin{aligned}
&\equiv \left(\bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d)\; {}_{1}\!\oplus_{\partial(f)_\gamma(\checkmark)} \mathbb{1} \right) \\
&\quad +_\gamma \mathop{\Large +}_{\alpha\in\mathrm{At}\setminus\gamma} \left(\bigoplus_{d\in\mathrm{supp}(\partial(f)_\alpha)} \partial\left(f\right)_\alpha\left(d\right)\cdot\exp(d) \right)
\end{aligned}
$$

We continue with L2 to get

$$
\begin{aligned}
\gamma; f^{(b)} &\equiv \gamma; \left(\left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d) \right); f^{(b)} +_b \mathbb{1} \right) \\
&\equiv \gamma; \left(\left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d); f^{(b)} \right) +_b \mathbb{1} \right) && \textit{Lemma D.7} \\
&\equiv \left(\left(\left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d); f^{(b)} \right) +_b \mathbb{1} \right) +_\gamma \mathbb{0} \right) && \mathsf{DF3} \\
&\equiv \left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d); f^{(b)} \right) +_{\gamma b} (\mathbb{1} +_\gamma \mathbb{0}) && \mathsf{G4} \\
&\equiv \left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d); f^{(b)} \right) +_{\gamma} (\mathbb{1} +_\gamma \mathbb{0}) && \gamma \leq_{\mathsf{BA}} b \\
&\equiv (\mathbb{0} +_{\bar\gamma} \mathbb{1}) +_{\bar\gamma} \left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d); f^{(b)} \right) && \mathsf{G3} \\
&\equiv (\bar\gamma; (\mathbb{0} +_{\bar\gamma} \mathbb{1})) +_{\bar\gamma} \left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d); f^{(b)} \right) && \mathsf{G2} \\
&\equiv (\bar\gamma; \mathbb{0}) +_{\bar\gamma} \left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d); f^{(b)} \right) && \mathsf{DF4} \\
&\equiv (\mathbb{0} +_{\bar\gamma} \mathbb{0}) +_{\bar\gamma} \left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d); f^{(b)} \right) && \mathsf{DF3} \\
&\equiv \mathbb{0} +_{\bar\gamma} \left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d); f^{(b)} \right) && \mathsf{G1} \\
&\equiv \left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d); f^{(b)} \right) +_\gamma \mathbb{0} && \mathsf{G3} \\
&\equiv \gamma; \left(\odot\partial(f)_\gamma(\checkmark)^*; \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\setminus\checkmark} \partial\left(f\right)_\gamma\left(d\right)\cdot\exp(d); f^{(b)} \right) && \mathsf{DF3}
\end{aligned}
$$

Separating the support gives

$$
\gamma; f^{(b)} \equiv \gamma; \Bigg(\odot\partial(f)_\gamma(\checkmark)^*; \Bigg(\bigoplus_{d\in\{\text{✗}\}\cup\mathbf{Out}} \partial(f)_\gamma(d)\cdot\exp(d); f^{(b)}
$$

$$\begin{aligned}
&\oplus \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\cap\mathbf{Act}\times\mathbf{Exp}} \partial(f)_\gamma(d)\cdot \exp(d); f^{(b)}\Big)\Big)\\
&\equiv \gamma;\Big(\bigoplus_{d\in\{\text{✗}\}\cup\mathbf{Out}} \partial(f)_\gamma(\checkmark)^*\partial\left(f\right)_\gamma(d)\cdot\exp(d); f^{(b)}\\
&\oplus \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\cap\mathbf{Act}\times\mathbf{Exp}} \partial(f)_\gamma(\checkmark)^*\partial\left(f\right)_\gamma(d)\cdot\exp(d); f^{(b)}\Big) && \mathsf{DF1}\\
&\equiv \gamma;\Big(\bigoplus_{d\in\{\text{✗}\}\cup\mathbf{Out}} \partial(f)_\gamma(\checkmark)^*\partial(f)_\gamma(d)\cdot\exp(d)\\
&\oplus \bigoplus_{d\in\mathrm{supp}(\partial(f)_\gamma)\cap\mathbf{Act}\times\mathbf{Exp}} \partial(f)_\gamma(\checkmark)^*\partial(f)_\gamma(d)\cdot\exp(d); f^{(b)}\Big) && \mathsf{S6, S3}\\
&\equiv \gamma;\left(\bigoplus_{d\in\mathrm{supp}(\partial(f^{(b)})_\gamma)} \partial(f^{(b)})_\gamma(d)\cdot\exp(d)\right) \quad \blacktriangleleft
\end{aligned}$$

▶ **Lemma D.10** ([24, 5.8]). *Let (X,β) be an $\mathcal{H}$-coalgebra, $R\subseteq X\times X$ a bisimulation equivalence on (X,β) and $[-]_R : X\to X/R$ the canonical quotient map. Then, there exists a unique transition map $\bar\beta : X/R\times \mathrm{At}\to\mathcal{M}_\omega(2+\mathbf{Out}+\mathbf{Act}\times X/R)$ which makes $[-]_R$ into an $\mathcal{H}$-coalgebra homomorphism from (X,β) to $(X/R,\bar\beta)$.*

▶ **Theorem 5.2.** *Let (X,β) be a finite* wGKAT *automaton. The map $h : X\to\mathbf{Exp}$ is a solution up to $\equiv$ of the system associated with (X,β) if and only if $[-]_\equiv\circ h$ is a* wGKAT *automaton homomorphism from (X,β) to $(\mathbf{Exp}/\equiv,\bar\partial)$. Where $\bar\partial$ is the unique transition function on $\mathbf{Exp}/\equiv$ which makes the quotient map $[-]_\equiv : \mathbf{Exp}\to\mathbf{Exp}/\equiv$ a* wGKAT *automaton homomorphism from $(\mathbf{Exp},\partial)$ to $(\mathbf{Exp}/\equiv,\bar\partial)$.*

Proof. This theorem and its proof are akin to [26, Theorem 21]. The theorem follows from Lemma D.10. Like Lemma C.9 we appeal to products over all $\alpha\in\mathrm{At}$ to construct $\mathcal{H}$, since At is finite.

Let $\bar\partial : \mathbf{Exp}/\equiv\to\mathcal{H}(\mathbf{Exp}/\equiv)$ be the unique $\mathcal{H}$-coalgebra structure map from Lemma D.10 which makes the quotient map $[-]_\equiv : \mathbf{Exp}\to\mathbf{Exp}/\equiv$ into an $\mathcal{H}$-coalgebra homomorphism. Then given a function $h : X\to\mathbf{Exp}$, $[-]_\equiv\circ h$ is a $\mathcal{H}$-coalgebra homomorphism from (X,β) to $(\mathbf{Exp}/\equiv,\bar\partial)$ if and only if the following diagram commutes.

$$\begin{tikzcd}
X \arrow[r, "{[-]_\equiv\circ h}"] \arrow[d, "\beta"] & \mathbf{Exp}/\equiv \arrow[d, "\bar\partial"] \\
\mathcal{H}X \arrow[r, dashed, "{\mathcal{H}([-]_\equiv\circ h)}"] & \mathcal{H}(\mathbf{Exp}/\equiv)
\end{tikzcd}$$

That is: $\bar\partial\circ[-]_\equiv\circ h = \mathcal{H}([-]_\equiv\circ h)\circ\beta$. Since $[-]_\equiv$ is a coalgebra homomorphism $\mathcal{H}[-]_\equiv\circ\partial = \bar\partial\circ[-]_\equiv$ and the previous statement holds if and only if

$$\mathcal{H}[-]_\equiv\circ\partial\circ h = \mathcal{H}([-]_\equiv\circ h)\circ\beta$$

That is if for all $x\in X, \alpha\in\mathrm{At}, o\in 2+\mathbf{Out}, (a,[e'])\in\mathbf{Act}\times\mathbf{Exp}/\equiv$

$$\beta(x)_\alpha(o) = \partial(h(x))_\alpha(o) \tag{7}$$

$$\sum_{h(x')\equiv e'}\beta(x)_\alpha(a,x') = \sum_{f\equiv e'}\partial(h(x))_\alpha(a,f) \tag{8}$$

We start by proving the converse. Assume that $[-]_\equiv \circ h$ is an $\mathcal{H}$-coalgebra homomorphism. Let (X, τ) be the Salomaa system associated with (X, β). To show h is a solution we will show that $h(x) \equiv (h^\# \circ \tau)(x)$ for all $x \in X$. By Theorem 5.1 it holds that for all $x \in X$ that

$$h(x) \equiv \mathop{+\!\!\!\!+}_{\alpha \in \mathrm{At}} \left(\bigoplus_{d \in \mathrm{supp}(\partial(h(x))_\alpha)} \partial(h(x))_\alpha(d) \cdot \exp(d) \right)$$

We can unroll this and use Lemma D.9 to obtain for all $x \in X$

$$h(x) = \mathop{+\!\!\!\!+}_{\alpha \in \mathrm{At}} \left(\bigoplus_{d \in 2+\mathbf{Out}} \partial(h(x))_\alpha(d) \cdot d \oplus \bigoplus_{(a,[e']_\equiv) \in E_\alpha} \left(\sum_{f \equiv e'} \partial(h(x))_\alpha(a, f) \right) \cdot a; e \right) \tag{9}$$

Where for each $\alpha \in \mathrm{At}$

$$E_\alpha = \{(a, Q) \in \mathbf{Act} \times \mathbf{Exp}/\equiv \mid \exists e' \;\; e' \in Q, \partial(h(x))_\alpha(a, e') \neq 0\}$$

We can write the other side of the equation, for all $x \in X$

$$(h^\# \circ \tau)(x) \equiv \mathop{+\!\!\!\!+}_{\alpha \in \mathrm{At}} \left(\bigoplus_{d \in 2+\mathbf{Out}} \beta(x)_\alpha(d) \cdot d \oplus \bigoplus_{(a,f) \in \mathrm{supp}(\beta(x)_\alpha)} \beta(x)_\alpha(a, f) \cdot a; h(f) \right)$$

Applying Lemma D.9 we get

$$(h^\# \circ \tau)(x) \equiv \mathop{+\!\!\!\!+}_{\alpha \in \mathrm{At}} \left(\bigoplus_{d \in 2+\mathbf{Out}} \beta(x)_\alpha(d) \cdot d \oplus \bigoplus_{[e]_\equiv \in F_\alpha} \left(\sum_{h(x') \equiv e'} \beta(x)_\alpha(a, x') \right) \cdot a; e' \right) \tag{10}$$

Where for all $\alpha \in \mathrm{At}$

$$F_\alpha = \{(a, Q) \in \mathbf{Act} \times \mathbf{Exp}/\equiv \mid \exists x' \;\; h(x') \in Q, \beta(x)_\alpha(a, x') \neq 0\}$$

We argue that Equation (9) and Equation (10) are the same. We do this by showing that the indexed collections that form each sum are equivalent for any fixed $x \in X$ and $\alpha \in \mathrm{At}$. For any $\alpha \in \mathrm{At}$ given $d \in 2 + \mathbf{Out}$ if $d \in \mathrm{supp}(\partial(h(x)))_\alpha$, then because of Equation (7) $d \in \mathrm{supp}(\beta(x)_\alpha)$. The converse holds by symmetry. Furthermore, for all such d, again by condition Equation (7) $\beta(x)_\alpha(d) = \partial(h(x))_\alpha(d)$. Consider $(a, Q) \in \mathbf{Act} \times \mathbf{Exp}/\equiv$. If $(a, Q) \in E_\alpha$ then there is some $e' \in \mathbf{Exp}$ such that $\partial(h(x))_\alpha(a, e') \neq 0$. Because of Equation (8) there exists some $x' \in X$ satisfying $h(x') \equiv e'$ and $\beta(x)_\alpha(a, x') \neq 0$. The converse holds by symmetry. Hence $F_\alpha = E_\alpha$. Similarly also by Equation (8) the weights associated with each $(a, Q) \in \mathbf{Act} \times \mathbf{Exp}/\equiv$ are the same. Therefore $h(x) = (h^\# \circ \tau)(x)$ for all $x \in X$ To prove the other direction we assume h is a solution up to $\equiv$. First we show that $(\partial \circ h^{(\#)} \circ \tau)(x) = (\mathcal{H}(h) \circ \beta)(x)$ for all $x \in X$. For all $x \in X$

$$(h^\# \circ \tau)(x) \equiv \mathop{+\!\!\!\!+}_{\alpha \in \mathrm{At}} \left(\bigoplus_{d \in 2+\mathbf{Out}} \beta(x)_\alpha(d) \cdot d \oplus \bigoplus_{(a,f) \in \mathrm{supp}(\beta(x)_\alpha)} \beta(x)_\alpha(a, f) \cdot a; h(f) \right)$$

When we apply the transition map, we can split into two cases for all $\alpha \in \mathrm{At}$.

1. For $o \in 2 + \mathbf{Out}$, $(\partial_\alpha \circ h^\# \circ \tau)(x)(o) = \beta(x)_\alpha(o) = (\mathcal{H}(h) \circ \beta)(x)_\alpha(o)$
2. For all $(a, e') \in \mathbf{Act} \times \mathbf{Exp}$,

$$(\partial_\alpha \circ h^\# \circ \tau)(x)(a, e') = \sum_{h(x') = e'} \beta(x)_\alpha(a, x') = (\mathcal{H}(h) \circ \beta)(x)_\alpha(a, e')$$

Hence $(\partial \circ h^{\#} \circ \tau) = (\mathcal{H}(h) \circ \beta)$. Composing with $\mathcal{H}[-]_{\equiv}$ gives

$$\begin{aligned}
\mathcal{H}[-]_{\equiv} \circ \partial \circ h^{\#} \circ \tau &= \mathcal{H}[-]_{\equiv} \circ \mathcal{H}(h) \circ \beta \\
\mathcal{H}[-]_{\equiv} \circ \partial \circ h^{\#} \circ \tau &= \mathcal{H}([-]_{\equiv} \circ h) \circ \beta && \text{Functor Associativity} \\
\bar{\partial} \circ [-]_{\equiv} \circ h^{\#} \circ \tau &= \mathcal{H}([-]_{\equiv} \circ h) \circ \beta && [-]_{\equiv} \text{ is a homomorphism} \\
\bar{\partial} \circ [-]_{\equiv} \circ h &= \mathcal{H}([-]_{\equiv} \circ h) \circ \beta && h^{\#} \circ \tau \equiv h
\end{aligned}$$

Therefore $[-]_{\equiv} \circ h$ is an $\mathcal{H}$-coalgebra homomorphism from (X, β) to $(\mathbf{Exp}/\equiv \bar{\partial})$. ◀

▶ **Lemma D.11.** *The congruence $\doteqdot$ is the kernel of a coalgebra homomorphism*

Proof. TOPROVE 14 ◀

▶ **Corollary D.1.** *For $e, f \in \mathbf{Exp}$, $e \doteqdot f \implies !_{\partial} e = !_{\partial} f$*

Proof. TOPROVE 15 ◀

▶ **Theorem 5.3** (Soundess of $\doteqdot$). *For all $e, f \in \mathbf{Exp}$, $e \doteqdot f \implies e \sim f$*

Proof. Immediate from Corollary D.1 and Proposition B.6 ◀

▶ **Theorem 5.4** (Completeness). *For all $e, f \in \mathbf{Exp}$, $e \sim f \implies e \doteqdot f$*

Proof. This fact now follows for the same reason as ProbGKAT [26]. There exists some bisimulation (R, δ) on $\langle e \rangle_{\partial} \times \langle f \rangle_{\partial}$ which witnesses their equivalence. Clearly R is finite as $\langle e \rangle_{\partial}$ and $\langle f \rangle_{\partial}$ are both finite due to 3.1. Let π_1, π_2 be the projection mappings from R to $(\langle e \rangle_{\partial}, \partial)$ and $(\langle f \rangle_{\partial}, \partial)$. Let j, k be the inclusion maps from $(\langle e \rangle_{\partial}, \partial)$ and $(\langle f \rangle_{\partial}, \partial)$ into $(\mathbf{Exp}, \partial)$. By Theorem 5.2 $j \circ \pi_1$, and $k \circ \pi_2$ are solutions of the Salomaa system associated with (R, δ) up to $\equiv$, since they may be composed with $[-]_{\equiv}$. They are also solutions up to $\doteqdot$ as $\equiv \subseteq \doteqdot$. Hence by UA $\forall (g, h) \in R \;\; (j \circ \pi_1)(g, h) \doteqdot (k \circ \pi_2)(g, h)$. So it follows that

$$e \doteqdot j(e) \doteqdot (j \circ \pi_1)(e, f) \doteqdot (k \circ \pi_2)(e, f) \doteqdot k(f) \doteqdot f$$ ◀

E Proofs for Section 6 (Decidability and Complexity)

As with GKAT and ProbGKAT we fix the set of atoms beforehand to avoid being able to encode boolean unsatisfiability.

▶ **Theorem E.1** (Decidability). *$e \doteqdot f$ is decidable*

Proof. TOPROVE 16 ◀

In order to characterize the complexity of deciding bisimilarity, we examine some of the internals of the partition refinement algorithm by defining some mappings. The runtime of coalgebraic partition refinement on a given coalgebra is determined by the runtime of two mappings **init** and **update**, as well as how much time it takes to compare weights. In order to characterize the required mappings and weights we need to determine a refinement interface for $\mathcal{H}$. This is done by encoding the functor and its associated coalgebra [33, Section 8].

▶ **Definition E.1** ([5]). *The encoding of the functor F is $(A, \flat)$, where A is a set of labels and $\flat$ is is a family of maps $\flat : FX \to \mathcal{B}_{\omega}(A \times X)$, one for every set X.*

$\flat$ can be interpreted as creating the labeled edges from the codomain of the transition function. For each X let ! be the unique map $X \to 1$, we let 1 be a fixed singleton set $\{*\}$ [5]. Furthermore:

▶ **Definition E.2** ([5]). *The encoding of the F-coalgebra (X, ψ) is given by $\langle F!, \flat\rangle \cdot \psi : X \to F1 \times \mathcal{B}_\omega(A \times X)$.*

We say that the coalgebra has $|X|$ states, and $\sum_{x \in X} |\flat(c(x))|$ edges [5]. The refinement interface will be defined in reference to the following mappings which essentially sort elements of the coalgebra according to their relationship to the subsets S and B.

▶ **Definition E.3** ([5]). *If $S \subseteq B \subseteq X$ then let $\chi_S^B : X \to 3$ such that*

$$\chi_S^B(x \in S) = 2 \quad \chi_S^B(x \in B \setminus S) = 1 \quad \chi_S^B(x \in X \setminus B) = 0$$

▶ **Definition E.4** (Refinement Interface [5]). *Given an encoding $(A, \flat)$ of the set functor F, a refinement interface for F is the triple $(W, \textbf{\textit{init}}, \textbf{\textit{update}})$. Where W is a set of weights, $\textbf{\textit{init}} : F1 \times \mathcal{B}_\omega A \to W$, $\textbf{\textit{update}} : \mathcal{B}_\omega A \times W \to W \times F3 \times W$ such that there exists a family of weight maps $w : \mathcal{P}X \to (FX \to W)$ such that for all $t \in FX$, and $S \subseteq B \subseteq X$*

$$w(X)(t) = \textbf{\textit{init}}(F!(t), \mathcal{B}_\omega \pi_1(\flat(t)))$$
$$(w(S)(t), F\chi_S^B(t), w(B \setminus S)(t)) = \textbf{\textit{update}}(\{a \mid (a, x) \in \flat(t), x \in S\}, w(B)(t))$$

The maps w can be thought of as tracking the weight into a set of states from the chosen element of the coalgebra. For example $w(B)(t)$ for $B \subseteq X, t \in FX$ is the weight into B from t.

Since $\mathcal{H}$ is a composite functor we can obtain the refinement interface by flattening and desorting $\mathcal{H}$ [33, Section 8] to obtain a set functor which is a coproduct of basic functors in the provided grammar. By doing this, the refinement interface can be obtained by piece-wise combining the refinement interfaces of the basic functors.

In our case $\mathcal{H}$ flattens into three polynomial functors and a monoid weighting functor. Consider the $\mathcal{H}$-coalgebra (Q, τ). We first flatten $\mathcal{H}$ to the multisorted Set^4 functor

$$\bar{H}(X_1, X_2, X_3, X_4) = (X_2^{\mathrm{At}}, \mathcal{M}_\omega^{X_3}, 2 + \mathbf{Out} + X_4, \mathbf{Act} \times X_1)$$

We desort this multisorted coalgebra into a coalgebra for $\bar{X} = X_1 + X_2 + X_3 + X_4$ and coproduct the resulting refinement interfaces to obtain a refinement interface for the desorted coalgebra [33, Section 8]. Each of the basic subfunctors has an existing refinement interface [33, 5]. So we simply count states and transitions. There are $|\mathrm{At}||Q|, |\mathrm{At}||Q|, |\mathbf{Act}||Q|, |Q|$ states in X_2, X_3, X_4 and X_1 respectively. There are $|Q||\mathrm{At}|, |Q||\mathrm{At}|, k, |\mathbf{Act}||Q|$ edges from $X_1 \to X_2, X_2 \to X_3, X_3 \to X_4$, and $X_4 \to X_1$ respectively. Where

$$k = \sum_{q \in Q, \alpha \in \mathrm{At}, o \in 2 + \mathbf{Out} + \mathbf{Act} \times Q} [\tau(q)_\alpha(o) \neq 0]$$

In order to know the complexity we must combine the refinement interfaces as per [33, 8.18] which allows us to bound the runtime of **init** and **update**, as well as comparisons of terms of type W and then apply [5, Theorem 3.4]. The labels and weights are the coproducts of the labels and weights of each subfunctor [33, 8.18]. Furthermore, the maps $\flat$ and w are component-wise functions build from the original $\flat_i, w_i$ maps from the subfunctors [33, 8.18]. The same is true for **init** and **update** [33, 8.18]. So we simply bound the runtimes of **init** and **update** by bounding the worst-case refinement interface of the underlying functions.

▶ **Theorem E.2.** *If* At *is fixed, then the bisimilarity of states in a* wGKAT *automaton* (Q, τ) *is decidable in time* $O((|Q| \cdot |\mathbf{Out}| + |\mathbf{Act}| \cdot |Q|^2)(\log^2(|\mathbf{Out}| + |\mathbf{Act}| \cdot |Q|)))$

Proof. TOPROVE 17 ◀

▶ **Corollary 6.1** (Decidability). *If* $e, f \in \mathbf{Exp}$*,* $n = \#e + \#f$*, and* At *is fixed, then* wGKAT *equivalence of* e *and* f *is decidable in time* $O(n^3 \log^2(n))$

Proof. Unreachable states need not be included in a bisimulation, therefore, if it exists, we can find the required bisimulation in the smallest subcoalgebra of $(\mathbf{Exp}, \partial)$ containing the two expressions we are deciding equivalence of.

By Lemma 3.1, $\langle e, f\rangle_\partial$ has no more than $\#e + \#f$ reachable states. Furthermore, the number of distinct actions and outputs from $\langle e, f\rangle_\partial$ are each no more than n also per Lemma 3.1. Hence $|\mathbf{Out}|, |\mathbf{Act}|, |Q| \leq n$, and hence $|Q|^2 \cdot |\mathbf{Act}| \leq n^3$. We assume that At is fixed. Hence, from Theorem E.2.

$$\begin{aligned}
O((|Q| \cdot |\mathbf{Out}| + |\mathbf{Act}| \cdot |Q|^2) \cdot \log^2(|\mathbf{Out}| + |\mathbf{Act}| \cdot |Q|))& \\
&\leq O((n^2 + n^3) \cdot \log^2(n + n^2)) \\
&= O(n^3 \log^2(n + n^2)) \\
&\leq O(n^3 \log^2(2n^2)) \\
&= O(n^3 (\log(2n^2))^2) \\
&= O(n^3 (\log(n^2) + \log(2))^2) \\
&= O(n^3 (\log^2(n^2) + 2\log(n^2)\log(2) + \log^2(2))) \\
&= O(n^3 (\log^2(n^2))) \\
&= O(n^3 4 \log^2(n)) \\
&= O(n^3 \log^2(n)) \quad ◀
\end{aligned}$$