The TAME Programming Language

Design and Implementation (Living Document)

Mike Gerwitz

May 2021

TAME is The Algebraic Metalanguage, a programming language and collection of tools designed to aid in the development, understanding, and maintenance of systems performing numerous calculations on a complex graph of dependencies, conditions, and a large number of inputs. TAME has existed for over a decade, and while its initial design was successful and still in active use today, it does suffer from inconsistencies and tradeoffs that introduce certain impediments to users of the language, and compromise future optimizations and language evolution. It also lacks documentation not just of the language itself, but also of the underlying principles and implementation.

This document is an attempt to formally consider certain parts of TAME as it undergoes redesign and reimplementation as part of the TAMER project. It is considered a living document—it is not likely to ever be a finished work.

Contents

0 Notational Conventions

This section provides a fairly terse overview of the foundational mathematical concepts used in this paper. While we try to reason about TAME in terms of algebra, first-order logic; and set theory; notation varies even within those branches. To avoid ambiguity, especially while introducing our own notation, core operators and concepts are explicitly defined below.

This section begins its numbering at 0. This is not only a hint that TAME (and this paper) use 0-indexing, but also because equations; definitions; theorems; corollaries; and the like are all numbered relative to their section. When you see any of these prefixed with "0.", this sets those references aside as foundational mathematical concepts that are not part of the theory and operation of TAME itself.

0.1 Propositional Logic

We reproduce here certain axioms and corollaries of propositional logic for convenience and to clarify our interpretation of certain concepts. The use of the symbols $\wedge$, $\vee$, and $\neg$ are standard. The symbol $\vdash$ means "infer". We use $\implies$ in place of $\rightarrow$ for implication, since the latter is used to denote the mapping of a domain to a codomain in reference to functions. We further use $\equiv$ in place of $\leftrightarrow$ to represent material equivalence.

Definition 0.1 (Logical Conjunction)**.** $p, q \vdash (p \wedge q)$.

Definition 0.2 (Logical Disjunction)**.** $p \vdash (p \vee q)$ and $q \vdash (p \vee q)$.

Definition 0.3 ($\wedge$-Associativity)**.** $(p \wedge (q \wedge r)) \vdash ((p \wedge q) \wedge r)$.

Definition 0.4 ($\vee$-Associativity)**.** $(p \vee (q \vee r)) \vdash ((p \vee q) \vee r)$.

Definition 0.5 ($\wedge$-Commutativity)**.** $(p \wedge q) \vdash (q \wedge p)$.

Definition 0.6 ($\vee$-Commutativity)**.** $(p \vee q) \vdash (q \vee p)$.

Definition 0.7 ($\wedge$-Simplification)**.** $p \wedge q \vdash p$.

Definition 0.8 (Double Negation)**.** $\neg\neg p \vdash p$.

Definition 0.9 (Law of Excluded Middle)**.** $\vdash (p \vee \neg p)$.

Definition 0.10 (Law of Non-Contradiction)**.** $\vdash \neg(p \wedge \neg p)$.

Definition 0.11 (De Morgan's Theorem). $\neg(p \wedge q) \vdash (\neg p \vee \neg q)$ and $\neg(p \vee q) \vdash (\neg p \wedge \neg q)$.

Definition 0.12 (Material Equivalence). $p \equiv q \vdash ((p \wedge q) \vee (\neg p \wedge \neg q))$.

$\equiv$ denotes a logical identity. Consequently, it'll often be used as a definition operator.

Definition 0.13 (Implication). $p \implies q \vdash (\neg p \vee q)$.

Definition 0.14 (Tautologies). $p \equiv (p \wedge p)$ and $p \equiv (p \vee p)$.

Definition 0.15 (Truth Values). $\vdash \top$ and $\vdash \neg\bot$.

Definition 0.16 (Boolean/Integer Equivalency). $\{0, 1\} \in \mathbb{Z}$, $\bot \equiv 0$ and $\top \equiv 1$.

0.2 First-Order Logic and Set Theory

The symbol $\emptyset$ represents the empty set—the set of zero elements. We assume that the axioms of ZFC set theory hold, but define $\in$ here for clarity.

Definition 0.17 (Set Membership). $x \in S \equiv \{x\} \cap S \neq \emptyset$.

TODO: *Introduce set-builder notation,* $\cup$, $\cap$.

$\forall$ denotes first-order universal quantification ("for all"), and $\exists$ first-order existential quantification ("there exists"), over some domain of discourse.

Definition 0.18 (Existential Quantification). $\exists x \in X\,(P(x)) \equiv \top \in \{P(x) \mid x \in X\}$.

Definition 0.19 (Universal Quantification). $\forall x \in X\,(P(x)) \equiv \neg\exists x \in X\,(\neg P(x))$.

Remark 0.1 (Vacuous Truth). By Definition 0.18, $\exists x \in \emptyset\,(P) \equiv \bot$ and by Definition 0.19, $\forall x \in \emptyset\,(P) \equiv \top$. And so we also have the tautologies $\vdash \neg\exists x \in \emptyset\,(P)$ and $\vdash \forall x \in \emptyset\,(P)$. Empty domains lead to undesirable consequences—in particular, we must carefully guard against them in Definition 0.20 and Definition 0.21 to maintain soundness.

We also have this shorthand notation:

$$\forall x, y, z \in S\,(P) \equiv \forall x \in S\,(\forall y \in S\,(\forall z \in S\,(P))), \tag{0.1}$$
$$\exists x, y, z \in S\,(P) \equiv \exists x \in S\,(\exists y \in S\,(\exists z \in S\,(P))). \tag{0.2}$$

Definition 0.20 (Quantifiers Over Connectives). Assuming that x is not free in φ,

$$\varphi \wedge \exists x \in X\,(P(x)) \equiv \exists x \in X\,(\varphi \wedge P(x)),$$
$$\varphi \vee \exists x \in X\,(P(x)) \equiv \exists x \in X\,(\varphi \vee P(x)) \qquad \text{assuming } X \neq \emptyset.$$

Definition 0.21 (Quantifier Elimination). $\exists x \in X\,(\varphi) \equiv \varphi$ assuming $X \neq \emptyset$ and x is not free in φ.

0.3 Functions

The notation $f = x \mapsto x' : A \to B$ represents a function f that maps from x to x', where $x \in A$ (the domain of f) and $x' \in B$ (the co-domain of f).

A function $A \to B$ can be represented as the Cartesian product of its domain and codomain, $A \times B$. For example, $x \mapsto x^2 : \mathbb{Z} \to \mathbb{Z}$ is represented by the set of ordered pairs $\{(x, x^2) \mid x \in \mathbb{Z}\}$, which looks something like

$$\{\ldots, (0,0), (1,1), (2,4), (3,9), \ldots\}.$$

The set of values over which some function f ranges is its *image*, which is a subset of its codomain. In the example above, both the domain and codomain are the set of integers $\mathbb{Z}$, but the image is $\{x^2 \mid x \in \mathbb{Z}\}$, which is clearly a subset of $\mathbb{Z}$.

We therefore have

$$A \to B \subset A \times B, \tag{0.3}$$
$$f : A \to B \vdash f \subset A \times B, \tag{0.4}$$
$$f = \alpha \mapsto \alpha' : A \to B = \{(\alpha, \alpha') \mid \alpha \in A \wedge \alpha' \in B\}, \tag{0.5}$$
$$f[D \subseteq A] = \{f(\alpha) \mid \alpha \in D\} \subset B, \tag{0.6}$$
$$f[] = f[A]. \tag{0.7}$$

An ordered pair (x, y) is also called a *2-tuple*. Generally, an n-*tuple* is used to represent an n-ary function, where by convention we have $(x) = x$. So $f(x, y) = f((x, y)) = x + y$. If we let $t = (x, y)$, then we also have $f(x, y) = ft$, which we'll sometimes write as a subscript f_t where disambiguation is necessary and where parenthesis may add too much noise; this notation is especially well-suited to indexes, as in f_1. Binary functions are often written using *infix* notation; for example, we have $x + y$ rather than $+(x, y)$.

$$f_x = f(x) \in \{b \mid (x, b) \in f\} \tag{0.8}$$

0.3.1 Binary Operations On Functions

Consider two unary functions f and g, and a binary relation R. We introduce a notation R° to denote the composition of a binary function with two unary functions.

$$f : A \to B \tag{0.9}$$
$$g : A \to D \tag{0.10}$$
$$R : B \times D \to F \tag{0.11}$$
$$fR^\circ g = \alpha \mapsto f_\alpha R g_\alpha : A \to F \tag{0.12}$$

Note that f and g must share the same domain A. In that sense, this is the mapping of the operation R over the domain A. This is analogous to unary function composition $f \circ g$.

A scalar value x can be mapped onto some function f using a constant function. For example, consider adding some number x to each element in the image of f:

$$f+^{\circ}(\_ \mapsto x) = \alpha \mapsto f_\alpha + x.$$

The symbol $\_$ is used to denote a variable that matches anything but is never referenced, and is often referred to as a "wildcard" (since it matches anything) or a "hole" (since its value goes nowhere).

Note that we consider the bracket notation for the image of a function $(f : A \to B)[A]$ to itself be a binary function. Given that, we have $f[]^{\circ} = f[A]^{\circ}$ for functions returning functions (such as vectors of vectors in Section 0.5).

0.4 Monoids and Sequences

Definition 0.22 (Monoid)**.** Let S be some set. A *monoid* is a triple $(S, \bullet, e)$ with the axioms

$$\begin{aligned} \bullet : S \times S \to S && \text{(Monoid Binary Closure)} \\ \forall a, b, c \in S\,(a \bullet (b \bullet c) = (a \bullet b) \bullet c)\,, && \text{(Monoid Associativity)} \\ \exists e \in S\,(\forall a \in S\,(e \bullet a = a \bullet e = a))\,. && \text{(Monoid Identity)} \end{aligned}$$

Monoids originate from abstract algebra. A monoid is a semigroup with an added identity element e. Only the identity element must be commutative, but if the binary operation $\bullet$ is *also* commutative, then the monoid is a *commutative monoid*.[1]

Consider some sequence of operations $x_0 \bullet \cdots \bullet x_n \in S$. Intuitively, a monoid tells us how to combine that sequence into a single element of S. When the sequence has one or zero elements, we then use the identity element $e \in S$: as $x_0 \bullet e = x_0$ in the case of one element or $e \bullet e = e$ in the case of zero.

Definition 0.23 (Monoidic Sequence)**.** Generally, given some monoid $(S, \bullet, e)$ and a sequence $\{x_j\}_{j \in J} \in S$ where $n < |J|$, we have $x_0 \bullet x_1 \bullet \cdots \bullet x_{n-1} \bullet x_n$ represent the successive binary operation on all indexed elements of x. When it's clear from context that the index is increasing by a constant of 1, that notation is shortened to $x_0 \bullet \cdots \bullet x_n$ to save space. When $|J| = 1$, then $n = 0$ and we have the sequence x_0. When $|J| = 0$, then $n = -1$, and no such sequence exists, in which case we expand into the identity element e.

[1] A commutative monoid is less frequently referred to as an *abelian monoid*, related to the common term *abelian group*.

For example, given the monoid $(\mathbb{Z}, +, 0)$, the sequence $1{+}2{+}\cdots{+}4{+}5$ can be shortened to $1{+}\cdots{+}5$ and represents the arithmetic progression $1{+}2{+}3{+}4{+}5 = 15$. If $x = \{1, 2, 3, 4, 5\}$, $x_0 + \cdots + x_n$ represents the same sequence. If $x = \{1\}$, that sequence evaluates to $1 = 1$. If $x = \{\}$, we have 0.

Lemma 0.1. $(\mathbb{B}, \wedge, \top)$ *is a commutative monoid.*

Proof. $(\mathbb{B}, \wedge, \top)$ is associative by Definition 0.3 and commutative by Definition 0.5. The identity element is $\top \in \mathbb{B}$ by Definition 0.7. □

Lemma 0.2. $(\mathbb{B}, \vee, \bot)$ *is a commutative monoid.*

Proof. $(\mathbb{B}, \vee, \bot)$ is associative by Definition 0.4 and commutative by Definition 0.6. The identity $\bot \in \mathbb{B}$ follows from

$$\begin{aligned}
\bot \vee p &\equiv p \vee \bot && \text{by Definition 0.6} \\
&\equiv \neg(\neg p \wedge \neg \bot) && \text{by Definition 0.11} \\
&\equiv \neg(\neg p) && \text{by Definition 0.7} \\
&\equiv p. && \text{by Definition 0.8}
\end{aligned}$$

□

0.5 Vectors and Index Sets

TAME supports scalar, vector, and matrix values. Unfortunately, its implementation history leaves those concepts a bit tortured.

A vector is a sequence of values, defined as a function of an index. An *index set* is a set that is used to index values from another set; they are usually subscripts of another set. A *family* is a set that is indexed by the same index set. In this paper, we assume that an index set represents a range of integer values from 0 to some number.

Definition 0.24 (Family and Index Set)**.** Let S be a family indexed by index set J. Then,

$$\{S_j\}_{j \in J}, \qquad J = \{0, 1, \ldots, |J| - 1\} \in \mathcal{P}(\mathbb{Z}). \tag{0.13}$$

$\mathcal{P}(S)$ denotes the *power set* of S—the set of all subsets of S including $\emptyset$ and S itself.

Definition 0.25 (Vector)**.** Let $J \subset \mathbb{Z}$ represent an index set. A *vector* $v \in \mathcal{V}^{\mathbb{R}}$ is a totally ordered sequence of elements represented as a function of an element of its index set:

$$v = \langle v_0, \ldots, v_j \rangle_{j \in J}^{\mathbb{R}} = j \mapsto v_j : J \to \mathbb{R}. \tag{0.14}$$

This definition means that $v_j = v(j)$, making the subscript a notational convenience. We may omit the superscript such that $\mathcal{V}^{\mathbb{R}} = \mathcal{V}$ and $\langle \ldots \rangle^{\mathbb{R}} = \langle \ldots \rangle$.

When appropriate, a vector may also be styled in a manner similar to linear algebra, noting that our indexes begin at 0 instead of 1:

$$\langle v_0, \ldots, v_j \rangle_{j \in J}^{\mathbb{R}} = \begin{bmatrix} v_0 \\ \vdots \\ v_j \end{bmatrix}_{j \in J} = \begin{bmatrix} v_0 \\ \vdots \\ v_j \end{bmatrix}. \tag{0.15}$$

Definition 0.26 (Matrix)**.** Let $J \subset \mathbb{Z}$ represent an index set. A *matrix* $M \in \mathcal{V}^{\mathcal{V}^{\mathbb{R}}}$ is a totally ordered sequence of elements represented as a function of an element of its index set:

$$M = \langle M_0, \ldots, M_j \rangle_{j \in J}^{\mathcal{V}^{\mathbb{R}}} = j \mapsto M_j : J \to \mathcal{V}^{\mathbb{R}}. \tag{0.16}$$

The consequences of Definition 0.26—defining a matrix as a vector of independent vectors—are important. This defines a matrix to be more like a multidimensional array, with no requirement that the lengths of the vectors be equal.

Corollary 0.1 (Matrix Row Length Variance)**.** $\vdash \exists M \in \mathcal{V}^{\mathcal{V}^{\mathbb{R}}} \left(\neg \forall j \forall k \left(|M_j| = |M_k| \right) \right)$.

Corollary 0.1 can be read "there exists some matrix M such that not all row lengths of M are equal". In other words—the inner vectors of a matrix can vary in length. However, certain systems (such as that of Axiom 1.1) may place restrictions by specifying the inner index set as a dependent type:

$$\langle M_{jk} \rangle_{\substack{j \in J \\ k \in K_j}} : J \to K_j \to \mathbb{R}, \quad K : J \to \mathcal{P}(\mathbb{Z}). \tag{0.17}$$

This makes K a set of index sets. When $|K[J]| = 1$ (that is—all K_j are the same index set), the matrix is *rectangular*, and can be written in a manner similar to linear algebra, noting that our indexes begin at 0 instead of 1; that we use double-subscripts (since matrices are functions returning functions); and that we use j, k in place of m, n.

$$\begin{bmatrix} M_{0_0} & M_{0_1} & \ldots & M_{0_k} \\ M_{1_0} & M_{0_1} & \ldots & M_{0_k} \\ \vdots & \vdots & \ddots & \vdots \\ M_{j_0} & M_{j_1} & \ldots & M_{j_k} \end{bmatrix}_{\substack{j \in J \\ k \in K_0}} \quad \text{if } |K[J]| = 1. \tag{0.18}$$

We may optionally omit the domains as in the vector notation.

If a matrix is *not* rectangular, the symbol $\boxtimes$ can be used to explicitly denote that specific scalar values are undefined; this is useful when the matrix representation is desirable when describing the transformation of non-rectangular data *into* rectangular data. For example,

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & \boxtimes \\ 5 & \boxtimes & \boxtimes \end{bmatrix}_{\substack{j \in J \\ k \in K_j}} = \langle \langle 0,1,2 \rangle, \langle 3,4 \rangle, \langle 5 \rangle \rangle, \qquad \begin{aligned} J &= \{0,1,2\}, \\ K &= \{(0,\{0,1,2\}),\ (1,\{0,1\}),\ (2,\{0\})\}. \end{aligned} \tag{0.19}$$

Definition 0.27 (Rank)**.** The *rank* of some variable x is an integer value

$$\|x\| = \begin{cases} 2 & x \in \mathcal{V}^{\mathcal{V}^{\mathbb{R}}}, \\ 1 & x \in \mathcal{V}^{\mathbb{R}}, \\ 0 & x \in \mathbb{R}. \end{cases}$$

Intuitively, the rank represents the number of dimensions of some variable x. A scalar has zero dimensions (a point); a vector has one (a line); and a matrix has two (a plane). In TAME, the rank is referred to as *dimensions* using the attribute `@dim`.

0.6 XML Notation

The grammar of TAME is XML. Equivalence relations will be used to map source expressions to an underlying mathematical expression. For example,

$$\texttt{<foo bar="x" baz="y" />} \equiv x = y$$

defines that pattern of `foo` expression to be materially equivalent to $x = y$—anywhere an equality relation appears, you could equivalently replace it with that XML representation without changing the meaning of the mathematical expression.

Variables may also bind to literals in an XML expression. For example,

$$\texttt{<quux } \alpha \texttt{ />}, \qquad \alpha \in \{\epsilon, \texttt{bar="baz"}\}$$

can represent either `<quux />` or `<quux bar="baz" />`. ϵ represents the empty string. Any text typeset in `typewriter` represents a literal string of characters.

1 Classification System

A *classification* is a user-defined abstraction that describes ("classifies") arbitrary data. Classifications can be used as predicates, generating functions, and can be composed into more complex classifications. Nearly all conditions in TAME are specified using classifications.

All classifications represent *first-order sentences*—that is, they contain no *free variables*. Intuitively, this means that all variables within a classification are *tightly coupled* to the classification itself. This limitation is mitigated through use of the template system.

Axiom 1.1 (Classification Introduction)**.**

TODO: *Symbol in place of $=$ here ($\equiv$ not appropriate).*

$$\begin{aligned}
&\texttt{<classify as="c" yields="}\gamma\texttt{" desc="_" }\alpha\texttt{>}\\
&\quad\begin{matrix}\left\langle M^0_{jk}\right\rangle_{\substack{j\in J\\ k\in K_j}} & \left\langle v^0_j\right\rangle_{j\in J} & s^0\\ \vdots & \vdots & \vdots\\ \left\langle M^l_{jk}\right\rangle_{\substack{j\in J\\ k\in K_j}} & \left\langle v^m_j\right\rangle_{j\in J} & s^n\end{matrix} \qquad (1.1a)\\
&\texttt{</classify>} = \mathop{\mathfrak{C}}^{c}_{\gamma}(\odot, M, v, s)\,,
\end{aligned}$$

where

$$J \subset \mathbb{Z} \neq \emptyset, \tag{1.1b}$$

$$\forall j \in J\Big(K_j \subset \mathbb{Z} \neq \emptyset\Big), \tag{1.1c}$$

$$\forall k\Big(M^k : J \to K_{j\in J} \to \mathbb{B}\Big), \tag{1.1d}$$

$$\forall k\Big(v^k : J \to \mathbb{B}\Big), \tag{1.1e}$$

$$\forall k\Big(s^k \in \mathbb{B}\Big), \tag{1.1f}$$

$$\alpha \in \{\epsilon, \texttt{any="true"}\}, \tag{1.1g}$$

and the monoid $\odot$ is defined as

$$\odot = \begin{cases}(\mathbb{B}, \wedge, \top) & \alpha = \epsilon,\\ (\mathbb{B}, \vee, \bot) & \alpha = \texttt{any="true"}.\end{cases} \tag{1.1h}$$

We use a 4-tuple $\mathfrak{C}(\odot, M, v, s)$ to represent a $\odot_1$-classification (a classification with the binary operation $\wedge$ or $\vee$) consisting of a combination of matrix (M), vector (v), and scalar (s) matches, rendered above in columns.[2] A $\wedge$-classification is pronounced "conjunctive classification", and $\vee$ "disjunctive".[3]

TODO: *Emphasize index sets, both relationships and nonempty.*

[2] The symbol $\odot$ was chosen since the binary operation for a monoid is $\bullet$ (see Section 0.4) and $\odot$ looks vaguely like $(\bullet)$, representing a portion of the monoid triple.

[3] Conjunctive and disjunctive classifications used to be referred to, respectively, as *universal* and *existential*, referring to fact that $\forall\{a_0, \ldots, a_n\}(a) \equiv a_0 \wedge \ldots \wedge a_n$, and similarly for $\exists$. This terminology has changed since all classifications are in fact existential over their matches' index sets, and so the terminology would otherwise lead to confusion.

The variables c and γ are required in Tame but are both optional in our notation $\mathfrak{C}^{c}_{\gamma}$, and can be used to identify the two different data representations of the classification.[4]

α serves as a placeholder for an optional `any="true"`, with ϵ representing the empty string in (1.1g). Note the wildcard variable matching `@desc`—its purpose is only to provide documentation.

Corollary 1.1 ($\odot$ Commutative Monoid)**.** *$\odot$ is a commutative monoid in Axiom 1.1.*

Proof. By Axiom 1.1, $\odot$ must be a monoid. Assume $\alpha = \epsilon$. Then, $\odot = (\mathbb{B}, \wedge, \top)$, which is proved by Lemma 0.1. Next, assume $\alpha =$ `any="true"`. Then, $\odot = (\mathbb{B}, \vee, \bot)$, which is proved by Lemma 0.1. □

While Axiom 1.1 seems to imply an ordering to matches, users of the language are free to specify matches in any order and the compiler will rearrange matches as it sees fit. This is due to the commutativity of $\odot$ as proved by Corollary 1.1, and not only affords great ease of use to users of Tame, but also great flexibility to compiler writers.

For notational convenience, we will let

$$\begin{aligned} \mathfrak{C}^{\wedge}(M, \nu, s) &= \mathfrak{C}\left((\mathbb{B}, \wedge, \top), M, \nu, s\right), \\ \mathfrak{C}^{\vee}(M, \nu, s) &= \mathfrak{C}\left((\mathbb{B}, \vee, \top), M, \nu, s\right). \end{aligned} \tag{1.2}$$

Axiom 1.2 (Classification-Predicate Equivalence)**.** Let $\mathfrak{C}^{c}_{\gamma}\left((\mathbb{B}, \bullet, e), M, \nu, s\right)$ be a classification by Axiom 1.1. We then have the first-order sentence

$$c \equiv \exists j \in J\left(\exists k \in K_j\left(M^0_{j_k} \bullet \cdots \bullet M^l_{j_k}\right) \bullet \nu^0_j \bullet \cdots \bullet \nu^m_j\right) \bullet s^0 \bullet \cdots \bullet s^n.$$

Axiom 1.3 (Classification Yield)**.** Let $\mathfrak{C}^{c}_{\gamma}\left((\mathbb{B}, \bullet, e), M, \nu, s\right)$ be a classification by Axiom 1.1. Then,

$$r = \begin{cases} 2 & M \neq \emptyset, \\ 1 & M = \emptyset \wedge \nu \neq \emptyset, \\ 0 & M \cup \nu = \emptyset, \end{cases} \tag{1.3a}$$

$$\exists j \in J\Big(\exists k \in K_j\Big(\Gamma^2_{j_k} = M^0_{j_k} \bullet \cdots \bullet M^l_{j_k} \bullet \nu^0_j \bullet \cdots \bullet \nu^m_j \bullet s^0 \bullet \cdots \bullet s^n\Big)\Big), \tag{1.3b}$$

$$\exists j \in J\Big(\Gamma^1_j = \nu^0_j \bullet \cdots \bullet \nu^m_j \bullet s^0 \bullet \cdots \bullet s^n\Big), \tag{1.3c}$$

$$\Gamma^0 = s^0 \bullet \cdots \bullet s^n. \tag{1.3d}$$

$$\gamma = \Gamma^r. \tag{1.3e}$$

[4] `classify/@yields` is optional in the grammar of Tame, but the compiler will generate one for us if one is not provided. As such, we will for simplicity consider it to be required here.

Theorem 1.1 (Classification Composition). *Classifications may be composed to create more complex classifications using the classification yield γ as in Axiom 1.3. This interpretation is equivalent to Axiom 1.2 by*

$$c \equiv \exists j \in J \left(\exists k \in K_j \left(\Gamma^2_{jk} \right) \bullet \Gamma^1_j \right) \bullet \Gamma^0. \tag{1.4}$$

Proof. Expanding each Γ in Axiom 1.3, we have

$$\begin{aligned}
c &\equiv \exists j \in J \left(\exists k \in K_j \left(\Gamma^2_{jk} \right) \bullet \Gamma^1_j \right) \bullet \Gamma^0 && \text{by Axiom 1.3} \\
&\equiv \exists j \in J \Big(\exists k \in K_j \Big(M^0_{j\,k} \bullet \cdots \bullet M^l_{jk} \bullet \nu^0_j \bullet \cdots \bullet \nu^m_j \bullet s^0 \bullet \cdots \bullet s^n \Big) \\
&\qquad \nu^0_j \bullet \cdots \bullet \nu^m_j \bullet s^0 \bullet \cdots \bullet s^n \Big) \bullet s^0 \bullet \cdots \bullet s^n, \\
&\equiv \exists j \in J \Big(\exists k \in K_j \Big(M^0_{j\,k} \bullet \cdots \bullet M^l_{jk} \Big) \bullet \nu^0_j \bullet \cdots \bullet \nu^m_j \bullet s^0 \bullet \cdots \bullet s^n \\
&\qquad \nu^0_j \bullet \cdots \bullet \nu^m_j \bullet s^0 \bullet \cdots \bullet s^n \Big) \bullet s^0 \bullet \cdots \bullet s^n, && \text{by Definition 0.20} \\
&\equiv \exists j \in J \Big(\exists k \in K_j \Big(M^0_{j\,k} \bullet \cdots \bullet M^l_{jk} \Big) && \text{by Definition 0.14} \\
&\qquad \nu^0_j \bullet \cdots \bullet \nu^m_j \bullet s^0 \bullet \cdots \bullet s^n \Big) \bullet s^0 \bullet \cdots \bullet s^n, \\
&\equiv \exists j \in J \Big(\exists k \in K_j \Big(M^0_{j\,k} \bullet \cdots \bullet M^l_{jk} \Big) && \text{by Definition 0.20} \\
&\qquad \nu^0_j \bullet \cdots \bullet \nu^m_j \Big) \bullet s^0 \bullet \cdots \bullet s^n \bullet s^0 \bullet \cdots \bullet s^n, \\
&\equiv \exists j \in J \Big(\exists k \in K_j \Big(M^0_{j\,k} \bullet \cdots \bullet M^l_{jk} \Big) && \text{by Definition 0.14} \\
&\qquad \nu^0_j \bullet \cdots \bullet \nu^m_j \Big) \bullet s^0 \bullet \cdots \bullet s^n.
\end{aligned}$$

□

Lemma 1.1 (Classification Predicate Vacuity). *Let $\mathfrak{C}^c_\gamma((\mathbb{B}, \bullet, e), \emptyset, \emptyset, \emptyset)$ be a classification by Axiom 1.1. $\odot$ is a monoid by Corollary 1.1. Then $c \equiv \gamma \equiv e$.*

Proof. First consider c.

$$\begin{aligned}
c &\equiv \exists j \in J \left(\exists k \left(e \right) \bullet e \right) \bullet e && \text{by Definition 0.23} \\
&\equiv \exists j \in J \left(e \bullet e \right) \bullet e && \text{by Definition 0.21} \\
&\equiv \exists j \in J \left(e \right) \bullet e && \text{by Monoid Identity} \\
&\equiv e \bullet e && \text{by Definition 0.21} \\
&\equiv e. && \text{by Monoid Identity}
\end{aligned}$$

For γ, we have $r = 0$ by Axiom 1.3, and so by similar steps as c, $\gamma = \Gamma^r = e$. Therefore $c \equiv e$. □

```
<classify as="always" yields="alwaysTrue"
          desc="Always true" />
```
$$= \underset{\texttt{alwaysTrue}}{\overset{\texttt{always}}{\mathfrak{C}^{\bigwedge}}}(\emptyset,\emptyset,\emptyset).$$

```
<classify as="never" yields="neverTrue"
          any="true"
          desc="Never true" />
```
$$= \underset{\texttt{neverTrue}}{\overset{\texttt{never}}{\mathfrak{C}^{\bigvee}}}(\emptyset,\emptyset,\emptyset).$$

Figure 1: `always` and `never` from package `core/base`.

Figure 1 demonstrates Lemma 1.1 in the definitions of the classifications `always` and `never`. These classifications are typically referenced directly for clarity rather than creating other vacuous classifications, encapsulating Lemma 1.1.

Theorem 1.2 (Classification Rank Independence). *Let* $\odot = (\mathbb{B}, \bullet, e)$. *Then,*

$$\underset{\gamma}{\mathfrak{C}}(\odot, M, \nu, s) \equiv \mathfrak{C}\left(\odot, \underset{\gamma'''}{\mathfrak{C}}(\odot, M, \emptyset, \emptyset), \underset{\gamma''}{\mathfrak{C}}(\odot, \emptyset, \nu, \emptyset), \underset{\gamma'}{\mathfrak{C}}(\odot, \emptyset, \emptyset, s)\right). \tag{1.5}$$

Proof. First, by Axiom 1.3, observe these special cases following from Lemma 1.1:

$$\begin{aligned} \Gamma'''^2 &= M^0_{j\,k} \bullet \cdots \bullet M^l_{j\,k}, && \text{assuming } \nu \cup s = \emptyset \\ \Gamma''^1 &= \nu^0_j \bullet \cdots \bullet \nu^m_j, && \text{assuming } M \cup s = \emptyset \\ \Gamma'^0 &= s^0 \bullet \cdots \bullet s^n. && \text{assuming } M \cup \nu = \emptyset \end{aligned} \tag{1.6}$$

By Theorem 1.1, we must prove

$$\exists j \in J\left(\exists k \in K_j\left(M^0_{j\,k} \bullet \cdots \bullet M^l_{j\,k}\right) \bullet \nu^0_j \bullet \cdots \bullet \nu^m_j\right) \bullet s^0 \bullet \cdots \bullet s^n \equiv c \equiv \exists j \in J\left(\exists k \in K_j\left(\gamma'''_{j_k}\right) \bullet \gamma''_j\right) \bullet \gamma'. \tag{1.7}$$

By Axiom 1.3, we have $r''' = 2$, $r'' = 1$, and $r' = 0$, and so $\gamma''' = \Gamma'''^2$, $\gamma'' = \Gamma''^1$, and $\gamma' = \Gamma'^0$. By substituting these values in 1.7, the theorem is proved. □

These definitions may also be used as a form of pattern matching to look up a corresponding variable. For example, if we have $\mathfrak{C}^{\texttt{foo}}$ and want to know its `@yields`, we can write $\mathfrak{C}^{\texttt{foo}}_{\gamma}$ to bind the `@yields` to γ.[5]

[5] This is conceptually like a symbol table lookup in the compiler.

1.1 Matches

A classification consists of a set of binary predicates called *matches*. Matches may reference any values, including the results of other classifications (as in Theorem 1.1 on page 11), allowing for the construction of complex abstractions over the data being classified.

Matches are intended to act intuitively across inputs of different ranks—that is, one can match on any combination of matrix, vector, and scalar values.

Axiom 1.4 (Match Input Translation)**.** Let j and k be free variables intended to be bound in the context of Axiom 1.2. Let J and K be defined by Axiom 1.1. Given some input x,

$$x_\iota = \begin{cases} x_{j_k} & \|x\| = 2; \\ x_j & \|x\| = 1; \\ x & \|x\| = 0, \end{cases} \qquad \begin{array}{l} j \in J, \\ k \in K_j. \end{array}$$

Axiom 1.5 (Match Rank)**.** Let $\sim : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be some binary relation. Then,

$$\|x_\iota \sim y_\iota\| = \begin{cases} \|x\| & \|x\| \geq \|y\|, \\ \|y\| & \text{otherwise.} \end{cases}$$

Axiom 1.6 (Element-Wise Equivalence ($\equiv_\iota$))**.**

$$\begin{aligned} \|x_\iota\| = \|y_\iota\| = 2,\ (x_\iota \equiv_\iota y_\iota) &\vdash \forall j, k\,(x_{j_k} \equiv y_{j_k}), \\ \|x_\iota\| = \|y_\iota\| = 1,\ (x_\iota \equiv_\iota y_\iota) &\vdash \forall j\,(x_j \equiv y_j), \\ \|x_\iota\| = \|y_\iota\| = 0,\ (x_\iota \equiv_\iota y_\iota) &\vdash (x \equiv y). \end{aligned}$$

Matches are represented by `match` nodes in Tame. Since the primitive is rather verbose, `core/match` also defines templates providing a more concise notation (`t:match-`ζ below).

Axiom 1.7 (Match Introduction)**.**

```
<t:match-ζ on="x"
           value="y" /> ≡ι  <match on="x">
                              <c:ζ>
                                <c:value-of name="y">
                              </c:ζ>
                            </match>
```

$$\equiv_\iota x_\iota \sim y_\iota, \qquad \sim\ = \begin{cases} = & \zeta = \texttt{eq}, \\ < & \zeta = \texttt{lt}, \\ > & \zeta = \texttt{gt}, \\ \leq & \zeta = \texttt{leq}, \\ \geq & \zeta = \texttt{geq}. \end{cases}$$

Axiom 1.8 (Match Equality Short Form)**.**

```
<match on="x" /> ≡ι <match on="x" value="TRUE" />.
```

TODO: *Define types and typedef.*

Axiom 1.9 (Match Membership). When T is a type defined with `typedef`,

$$\texttt{<match on="x" anyOf="T" />} \equiv_\iota x_\iota \in \mathsf{T}.$$

Theorem 1.3 (Classification Match Element-Wise Binary Relation). *Within the context of Axiom 1.2, all match forms represent binary relations* $\mathbb{R} \times \mathbb{R} \to \mathbb{B}$ *ranging over individual elements of all index sets* J *and* $K_j \in K$.

Proof. First, observe that each of $=$, $<$, $>$, $\leq$, $\geq$, and $\in$ have type $\mathbb{R} \times \mathbb{R} \to \mathbb{B}$. We must then prove that x_ι and y_ι are able to be interpreted as $\mathbb{R}$ within the context of Axiom 1.2.

When $x, y \in \mathbb{R}$, we have the trivial case $x_\iota = x \in \mathbb{R}$ and $y_\iota = y \in \mathbb{R}$ by Axiom 1.4. Otherwise, variables j and k are free.

Consider $\|x_\iota \sim y_\iota\| = 2$; then $\|x_\iota \sim y_\iota\| \in \mathcal{V}^{\mathcal{V}^{\mathbb{R}}}$ by Definition 0.27, and so by Theorem 1.2 we have

$$\forall j \in J \left(\forall k \in K_j \left(M^0_{j\,k} \bullet \cdots \bullet M^1_{j\,k} \right) \right) \equiv \forall j \in J \left(\forall k \in K_j \left(x_\iota \sim y_\iota \right) \right), \tag{1.8}$$

which binds j and k to the variables of their respective quantifiers. Proceed similarly for $\|x_\iota \sim y_\iota\| = 1$ and observe that j becomes bound.

Assume $x \in \mathcal{V}^{\mathcal{V}^{\mathbb{R}}}$; then $x_{j_k} \in \mathbb{R}$ by Definition 0.26. Assume $y \in \mathcal{V}^{\mathbb{R}}$; then $y_j \in \mathbb{R}$ by Definition 0.25. Finally, observe that j ranges over J in 1.8, and k over K_j. □

Theorem 1.3 is responsible for proving that matches range over each individual index. More subtly, it also shows that matches work with any combination of rank. Figure 2 demonstrates a complete translation of source Tame XML using all ranks.

```
<classify as="fullrank" desc="Example of all ranks">
  <match on="A" value="u" />
  <match on="A" value="t" />
  <match on="u" value="t" />
  <match on="t" />
</classify>
```

$$\equiv_\iota A_\iota = u_\iota$$
$$\equiv_\iota A_\iota = t_\iota$$
$$\equiv_\iota u_\iota = t_\iota$$
$$\equiv_\iota t_\iota = \top$$

by Axiom 1.7 (1.9)

$$= \overset{\texttt{fullrank}}{\mathfrak{C}^{\wedge}} \left(\left(\left(A_{j_k} = u_j \right), \left(A_{j_k} = t \right) \right), \left(u_j = t \right), t \right)$$

by Axiom 1.1 (1.10)

$$\equiv \exists j \in J \left(\exists k \in K_j \left(\left(A_{j_k} = u_j \right) \wedge \left(A_{j_k} = t \right) \right) \wedge u_j = t \right) \wedge t.$$

by Theorem 1.3. (1.11)

Figure 2: Example demonstrating Theorem 1.3 using all ranks.

Visually, the one-dimensional construction of Axiom 1.2 does not lend itself well to how intuitive the behavior of the system actually is. We therefore establish a relationship to the notation of linear algebra to emphasize the relationship between each of the inputs.

Axiom 1.10 (Classification Matrix Notation). Let Γ^2 be defined by Axiom 1.3. Then,

$$\Gamma^2 = \begin{bmatrix} M^0_{0_0} & \dots & M^0_{0_k} \\ \vdots & \ddots & \vdots \\ M^0_{j_0} & \dots & M^0_{j_k} \end{bmatrix} \bullet \cdots \bullet \begin{bmatrix} M^l_{0_0} & \dots & M^l_{0_k} \\ \vdots & \ddots & \vdots \\ M^l_{j_0} & \dots & M^l_{j_k} \end{bmatrix} \bullet \begin{bmatrix} v^0_0 \\ \vdots \\ v^0_j \end{bmatrix} \bullet \cdots \bullet \begin{bmatrix} v^m_0 \\ \vdots \\ v^m_j \end{bmatrix} \bullet s^0 \bullet \cdots \bullet s^n,$$

from which Γ^1, Γ^0, and γ can be derived.

Remark 1.1 (Logical Connectives With Matrix Notation). From the definition of Axiom 1.10, it should be clear that the logical connective $\bullet$ necessarily acts like a Hadamard product[1] with respect to how individual elements are combined.

Axiom 1.10 makes it easy to visualize classification operations simply by drawing horizontal boxes across the predicates, as demonstrated by Figure 3. This visualization helps to show intuitively how the classification system is intended to function, with matrices serving as higher-resolution vectors.[6]

$$\begin{array}{l}
\left(M^0_{0_0} \bullet \cdots \bullet M^l_{0_0}\right) \bullet v^0_0 \bullet \cdots \bullet v^m_0 \bullet s^0 \bullet \cdots \bullet s^n \\
\quad \vdots \qquad\qquad \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Gamma^2_0 \\
/\!/ \left(M^0_{0_k} \bullet \cdots \bullet M^l_{0_k}\right) \bullet v^0_0 \bullet \cdots \bullet v^m_0 \bullet s^0 \bullet \cdots \bullet s^n \\
\boxed{\begin{bmatrix} M^0_{0_0} & \dots & M^0_{0_k} \\ \vdots & \ddots & \vdots \\ M^0_{j_0} & \dots & M^0_{j_k} \end{bmatrix} \bullet \cdots \bullet \begin{bmatrix} M^l_{0_0} & \dots & M^l_{0_k} \\ \vdots & \ddots & \vdots \\ M^l_{j_0} & \dots & M^l_{j_k} \end{bmatrix} \bullet \begin{bmatrix} v^0_0 \\ \vdots \\ v^0_j \end{bmatrix} \bullet \cdots \bullet \begin{bmatrix} v^m_0 \\ \vdots \\ v^m_j \end{bmatrix} \bullet s^0 \bullet \cdots \bullet s^n} \quad \vdots \\
/\!/\!/ \left(M^0_{j_0} \bullet \cdots \bullet M^l_{j_0}\right) \bullet v^0_j \bullet \cdots \bullet v^m_j \bullet s^0 \bullet \cdots \bullet s^n \\
\quad \vdots \qquad\qquad \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Gamma^2_j \\
\left(M^0_{j_k} \bullet \cdots \bullet M^l_{j_k}\right) \bullet v^0_j \bullet \cdots \bullet v^m_j \bullet s^0 \bullet \cdots \bullet s^n
\end{array}$$

Figure 3: Visual interpretation of classification by Axiom 1.10. For each boxed row of the matrix notation there is an equivalence to the first-order logic of Theorem 1.1.

Lemma 1.2 (Match As Proposition). *Matches can be represented using propositional logic provided that binary operators of Axiom 1.7 are restricted to* $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$.

[6] For example, with insurance, one may have a vector of data by risk location, and a matrix of chosen class codes by location. Consequently, we expect M_j to be the set of class codes associated with location j so that it can be easily matched against location-level data v_j.

Proof.

$$\begin{aligned}
x = \top &\equiv x, && = : \mathbb{B} \times \mathbb{B} \to \mathbb{B};\\
x = \bot &\equiv \neg x, && = : \mathbb{B} \times \mathbb{B} \to \mathbb{B};\\
x < y &\equiv \neg x \wedge y, && < : \mathbb{B} \times \mathbb{B} \to \mathbb{B};\\
x > y &\equiv x \wedge \neg y, && > : \mathbb{B} \times \mathbb{B} \to \mathbb{B};\\
x \leq y &\equiv \neg x \vee y, && \leq : \mathbb{B} \times \mathbb{B} \to \mathbb{B};\\
x \geq y &\equiv x \vee \neg y, && \geq : \mathbb{B} \times \mathbb{B} \to \mathbb{B};\\
x \in \mathbb{B} &\equiv \top, && \in : \mathbb{B} \times \mathbb{B} \to \mathbb{B}.
\end{aligned}$$

□

Theorem 1.4 (Classification As Proposition). *Classifications with either $M \cup \nu = \emptyset$ or with constant index sets can be represented by propositional logic provided that the domains of the binary operators of Axiom 1.7 are restricted to $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$.*

Proof. Propositional logic does not include quantifiers or relations. Matches of the domain $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$ are proved to be propositions by Lemma 1.2. Having eliminated relations, we must now eliminate quantifiers.

Assume $M \cup \nu = \emptyset$. By Theorem 1.2,

$$c \equiv s^0 \bullet \cdots \bullet s^n,$$

which is a propositional formula.

Similarly, if we define our index set J to be constant, we are then able to eliminate existential quantification over J as follows:

$$\begin{aligned}
c &\equiv \exists j \in J \left(\nu_j^0 \bullet \cdots \bullet \nu_j^m \right),\\
&\equiv \left(\nu_0^0 \bullet \cdots \bullet \nu_0^m \right) \vee \cdots \vee \left(\nu_{|J|-1}^0 \bullet \cdots \bullet \nu_{|J|-1}^m \right),
\end{aligned} \tag{1.12}$$

which is a propositional formula. Similarly, for matrices,

$$\begin{aligned}
c &\equiv \exists j \in J \left(\exists k \in K_j \left(M_{j\,k}^0 \bullet \cdots \bullet M_{j\,k}^l \right) \right),\\
&\equiv \exists j \in J \left(\left(M_{j\,0}^0 \bullet \cdots \bullet M_{j\,|K_j|-1}^0 \right) \vee \cdots \vee \left(M_{j\,0}^l \bullet \cdots \bullet M_{j\,|K_j|-1}^l \right) \right),
\end{aligned}$$

and then proceed as in 1.12. □

References

[1] *Hadamard product (matrices)*. Wikipedia. May 26, 2020. URL: https://en.wikipedia.org/wiki/Hadamard_product_(matrices).

Index