Compositional Quantitative Reasoning*

Krishnendu Chatterjee
UC Berkeley

Luca de Alfaro
UC Santa Cruz

Marco Faella
Università di Napoli "Federico II"

Thomas A. Henzinger
EPFL and UC Berkeley

Rupak Majumdar
UC Los Angeles

Mariëlle Stoelinga
University of Twente

Abstract. We present a compositional theory of system verification, where specifications assign real-numbered costs to systems. These costs can express a wide variety of quantitative system properties, such as resource consumption, price, or a measure of how well a system satisfies its specification. The theory supports the composition of systems and specifications, and the hiding of variables. Boolean refinement relations are replaced by real-numbered distances between descriptions of a system at different levels of detail. We show that the classical boolean rules for compositional reasoning have quantitative counterparts in our setting.

While our general theory allows costs to be specified by arbitrary cost functions, we also consider a class of linear cost functions, which give rise to an instance of our framework where all operations are computable in polynomial time.

1. Introduction

In formal approaches to system design and verification, a *specification* prescribes a set of desirable behaviors; and a system *implements* the specification if all of its behaviors are among the desirable ones. Hence, the implementation question is phrased as a yes/no, or boolean, question. This setup contrasts with the quantitative point of view taken in other disciplines. For instance, in optimization theory (e.g., optimal control [8, 1] or linear programming [17]), a specification not only describes the correct implementations (the feasible solutions), but also describes the quality of the implementations via rewards, costs, or objective functions.

Recently, much research has been devoted to developing a quantitative approach to formal design and verification. Quantitative temporal logics [6, 10, 4] and calculi [13, 7, 15, 6, 5, 14] have been suggested as specification formalisms, and directed and undirected metrics on states have been proposed as generalized notions of refinement and behavioral equivalence [9, 10, 2]. The proposal in this paper differs from the above quantitative approaches in two main respects. In [9, 10, 2, 6, 5], the quantitative aspect aims at providing a measure of the similarity between systems. In our proposal, on the other hand, the quantitative aspect is used to define a *cost* that differentiates among more or less desirable implementations, in analogy with the setting common in optimization problems. Furthermore, the emphasis in previous work has been on systems that are modeled as indivisible units. The success of the boolean theory, however, crucially depends on its modularity: it supports component-based design and modular reasoning, whereby the properties of a complex system can be derived from the properties of its components [3, 12].

The basic operations of a modular theory of systems are composition and refinement [16]. Composition permits the construction of complex systems from simple building blocks; refinement permits a change in the level of detail when describing a system. In this paper, we study the interaction of composition and refinement in a quantitative setting, and we develop a modular theory of system verification for quantitative specifications. As in formal system design, the emphasis is on stepwise refinement and modular reasoning; as in optimization theory, the notions of specification and implementation are quantitative, rather than boolean. The proposal of this paper can thus be viewed both as a quantitative extension of the classical techniques for modular and hierarchical reasoning, and as an exploration of the issues of modularity and hierarchy in the context of optimization theory.

We develop our theory in both static (combinational) and dynamic (sequential) contexts. We distinguish between systems and specifications. A static *system* is a set of assignments of values to variables, and it describes a set of behaviors (e.g., input/output pairs) that may occur. A dynamic system is a set of infinite sequences of assignments to variables. A *specification* assigns a cost to each possible value assignment (in the static case), or to each possible infinite behavior (in the dynamic case).

The composition of two systems is defined as usual, by taking the intersection of the system behaviors. The composition of two specifications F and G combines the costs that are assigned by F and G to the implementations via a com-

* Supported in part by the NSF grants CCR-0234690, CCR-0208875, and CCR-0225610; by the NSF grant CCR-0132780 and ARP grant SC20051123.

bination operator $\oplus : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$, which is typically either $+$ or $\max$. Thus, the cost assigned by the composition $F \| G$ to a variable assignment s corresponds to the sum (if $\oplus$ is $+$) or to the maximum (if $\oplus$ is $\max$) of the costs assigned by F and G to s. For example, the cost of a variable assignment may express the amount of memory needed by a configuration. When composing two systems in parallel, it is reasonable to add their memory consumptions and choose $\oplus$ to be $+$; when composing the two systems sequentially, one can reuse the memory if it is shared, and may choose $\oplus$ to me $\max$.

We introduce two notions of refinement: one between a system and a specification; the other between two specifications. Both notions are quantitative. The refinement distance $d(I, F)$ between a system I and a specification F corresponds to the maximum cost, according to F, of a behavior of I. The refinement distance $r(F, F')$ between a specification F and a more abstract specification F' measures the maximum *relative* cost increase from F to F' incurred by *any* behavior (in particular, $r(F, F) = 1$). We focus on the relative distance between specifications, rather than on the absolute one, because it leads to a theory that is more robust to disturbances. For instance, if the cost assigned by F can grow unboundedly large for undesirable behaviors, then the absolute distance between F and $1.01 \cdot F$ is infinite, while the relative distance is 1.01. Moreover, the relative distance is less sensitive to an increase in costs that affect already high costs, and that are thus less likely to be paid by actual implementations.

With these definitions, we are able to cast in a quantitative setting the classical theorems relating composition and refinement. In particular, the classical rule for compositionality of boolean refinement, that $I \preceq F$ and $J \preceq G$ implies $I \| J \preceq F \| G$ (where $I \preceq F$ means "I implements F"), corresponds in our quantitative setting to the two inequalities

$$d(I \| J, F \| G) \leq \max\{d(I, F), d(J, G)\}, \tag{1}$$

$$r(F \| G, F' \| G') \leq \max\{r(F, F'), r(G, G')\}, \tag{2}$$

where I and J are systems, and F and F' are specifications. The classical rule allows the decomposition of a refinement problem into two simpler problems: in order to prove that a composite system $I \| J$ implements $F \| G$, it suffices to prove independently that I implements F, and that J implements G. Similarly, the quantitative rules suggest a compositional approach to optimization problems: they provide bounds on the optimality of a global solution in terms of bounds on the optimality of partial solutions for the individual components. Moreover, the transitivity of boolean refinement, that $I \preceq F$ and $F \preceq F'$ implies $I \preceq F'$, corresponds in our setting to the two inequalities

$$d(I, F') \leq d(I, F) \cdot r(F, F'), \tag{3}$$

$$r(F, F'') \leq r(F, F') \cdot r(F', F''), \tag{4}$$

where I is a system and F, F', and F'' are specifications. These rules allow the chaining of multiple refinement steps.

In general, cost functions are infinite, semantic objects. In order to obtain a computational theory, we introduce *symbolic cost algebras*. These are a quantitative generalization of symbolic theories used in classical system verification to represent sets of states. A symbolic cost algebra consists of a set of cost functions that have finite representations, and that are effectively closed under composition, variable hiding, and refinement. Symbolic cost algebras furnish symbolic algorithms for computing refinement distances. To illustrate symbolic cost algebras, we introduce a specific example, the *maxlin* cost algebra, which enables the expression of piecewise-linear concave cost functions (which encode convex utility functions). We show that in this algebra, composition, hiding, and refinement distances can all be computed in polynomial time. As a result, we obtain a compositional and efficiently computable theory for quantitative reasoning about component-based systems.

Example 1 *We illustrate our theory with a simple example. Consider a factory receiving orders from a client that requires just-in-time delivery. At time x, the factory receives an order, and the client expects the delivery at a certain time, say, 7 days after ordering. If the delivery occurs earlier than that, the client has to pay stocking costs, and if the delivery is late, the client needs to delay the production line. Therefore, the factory's contract G with the client includes a penalty, which the factory has to pay if an order placed at time x is delivered at time z. The factory budgets a production cost of $c(x, z) = \alpha + |z - x - 7|$, consisting of a fixed portion of α and a penalty of $|z - x - 7|$. Thus, deviations from the ideal delivery time of 7 days are punished in a linear fashion. A system I in this setting is a set of behaviors specifying a production schedule, indicating when the orders acquired will be delivered. Thus, the refinement distance $d(I, G)$ represents what the factory targets to spend on the schedule I.*

The factory has subcontracted the three production phases —processing, packaging, and shipment— and stipulates three contracts G_1, G_2, and G_3 with the subcontractors. As the combined contract $G_1 \| G_2 \| G_3$ accumulates the costs of the individual contracts, their composition is formed by taking $+$ for the operator $\oplus$. Note that $d(I, G_i)$ is the maximum cost of contract G_i under schedule I, and $d(I, G_1 \| G_2 \| G_3)$ is the maximum total cost of combination of contracts under that schedule. More interestingly, $r(G_1 \| G_2 \| G_3, G)$ measures to what extent the subcontractors together meet the planned production costs. The transitivity rule (3 and 4) shows that for every schedule I, we have $d(I, G) \leq d(I, G_1 \| G_2 \| G_3) \cdot r(G_1 \| G_2 \| G_3, G)$. So, if $r(G_1 \| G_2 \| G_3, G)$ is, say, 1.05, then the factory is guaranteed that the costs realized by the subcontractors will be at most 5% more than the targeted costs.

The paper is organized as follows. In Section 2, we provide the basic definitions of system, specification, and composition, as well as variable hiding. In Section 3, we define our two notions of quantitative refinement, and we characterize the relationship between composition, hiding, and refinement, giving (among other results) precise statements of (1)–(4). In Section 4, we introduce symbolic cost algebras, and as a particular example, the maxlin algebra. We show that the maxlin algebra allows polynomial-time computation of composition, hiding, and refinement distances. In Section 5, we extend our setting to dynamic systems.

2. Systems and Specifications

Variables and assignments. Let *Vars* be a fixed set of variables; each variable $x \in \mathit{Vars}$ has an associated domain D_x. Given a set $\mathcal{V} \subseteq \mathit{Vars}$ of variables, a *state* over $\mathcal{V}$ is a function $s \in \Pi_{x\in\mathcal{V}} D_x$ that assigns to each variable $x \in \mathcal{V}$ a value $s(x) \in D_x$. We denote by $S[\mathcal{V}]$ the set of all states over $\mathcal{V}$. Given a state $s \in S[\mathcal{V}]$ and $\mathcal{W} \subseteq \mathcal{V}$, we denote by $s|_\mathcal{W} \in S[\mathcal{W}]$ the *restriction* of s to $\mathcal{W}$, defined by $s|_\mathcal{W}(x) = s(x)$ for all $x \in \mathcal{W}$. Given a state $s \in S[\mathcal{V}]$, a variable $x \in \mathit{Vars}$, and a value $d \in D_x$ for x, we denote by $(s \circ [x \mapsto d]) \in S[\mathcal{V} \cup \{x\}]$ the state defined by $(s \circ [x \mapsto d])(y) = s(y)$ for $y \in \mathcal{V} \setminus \{x\}$, and by $(s \circ [x \mapsto d])(x) = d$.

Systems. A *system* $I = \langle \mathcal{V}_I, B_I\rangle$ consists of a set $\mathcal{V}_I$ of variables,[1] and of a subset $B_I \subseteq S[\mathcal{V}_I]$ of states over $\mathcal{V}_I$. We omit the subscript I when clear from the context. We define system composition and hiding as follows.

- **Composition.** Given two systems I and J, their *composition* $I\|J$ is defined by $\mathcal{V}_{I\|J} = \mathcal{V}_I \cup \mathcal{V}_J$, and $B_{I\|J} = \{s \in S[\mathcal{V}_I \cup \mathcal{V}_J] \mid s|_{\mathcal{V}_I} \in B_I \wedge s|_{\mathcal{V}_J} \in B_J\}$.
- **Hiding.** Given a system I and a variable $x \in \mathit{Vars}$, we define the result $\mathit{Hide}_x(I)$ of *hiding* x in I by I itself if $x \notin \mathcal{V}_I$, and otherwise by $J = \langle \mathcal{V}_J, B_J\rangle$, where $\mathcal{V}_J = \mathcal{V}_I \setminus \{x\}$ and $B_J = \{s \in S[\mathcal{V}_J] \mid \exists d \in D_x.(s \circ [x \mapsto d]) \in B_I\}$. For $X = \{x_1, x_2, \dots, x_n\} \subseteq \mathit{Vars}$, we define[2] $\mathit{Hide}_X(I) = \mathit{Hide}_{x_1}(\mathit{Hide}_{x_2}(\cdots \mathit{Hide}_{x_n}(I)))$.

Specifications. Let $\mathbb{R}_{\geq 0}$ be the set of nonnegative real numbers together with ∞. A *specification* $F = \langle \mathcal{V}_F, c_F\rangle$ consists of a set $\mathcal{V}_F$ of variables, and a function $c_F : S[\mathcal{V}_F] \mapsto \mathbb{R}_{\geq 0}$ that assigns to each state $s \in S[\mathcal{V}_F]$ a real number $c_F(s) \in \mathbb{R}_{\geq 0}$. The value $c_F(s)$ can be interpreted as the cost of realizing s. We require that the cost be bounded away from 0: for each specification F, there is a positive constant $b \in \mathbb{R}_{>0}$ such that $c_F(s) \geq b$ for all $s \in S[\mathcal{V}_F]$. We omit the subscript F when clear from the context.

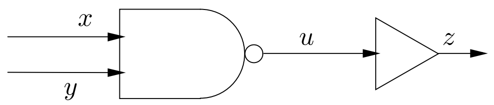

Figure 1.

To define the composition of two specifications, we need an operator to combine costs. A *combination operator* $\oplus : \mathbb{R}^2_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$ is a binary function that satisfies the following properties: (1) $\oplus$ is monotonic, commutative, and associative, and (2) $\oplus$ subdistributes over multiplication: for all $a, b, c \in \mathbb{R}_{\geq 0}$, we have $(a \cdot b) \oplus (a \cdot c) \leq a \cdot (b \oplus c)$. Common choices for $\oplus$ are $+$ and $\max$. For a fixed combination operator $\oplus$, we define hiding and composition of specifications as follows.

- **Composition.** Given two specifications F and G, we define their *composition* $F\|G$ by $\mathcal{V}_{F\|G} = \mathcal{V}_F \cup \mathcal{V}_G$, and for all $s \in S[\mathcal{V}_F \cup \mathcal{V}_G]$, by $c_{F\|G}(s) = c_F(s|_{\mathcal{V}_F}) \oplus c_G(s|_{\mathcal{V}_G})$.
- **Hiding.** Given a specification F and a variable $x \in \mathit{Vars}$, we define the result $\mathit{Hide}_x(F)$ of *hiding* x in F by F itself if $x \notin \mathcal{V}_F$, and otherwise by $G = \langle \mathcal{V}_G, c_G\rangle$, where $\mathcal{V}_G = \mathcal{V}_F \setminus \{x\}$ and, for all $s \in S[\mathcal{V}_G]$, by $c_G(s) = \inf_{d\in D_x} c_F(s \circ [x \mapsto d])$. In other words, $\mathit{Hide}_x(F)$ is equal to the minimum of F over all values of x. Once we hide a variable x in F, the specification F no longer cares about the variable x, and hence the cost is minimal over all values of x. Also note that by defining hiding as above, we obtain the boolean theory as a special case. For $X = \{x_1, x_2, \dots, x_n\} \subseteq \mathit{Vars}$, we define $\mathit{Hide}_X(F) = \mathit{Hide}_{x_1}(\mathit{Hide}_{x_2}(\cdots \mathit{Hide}_{x_n}(F)))$.

With abuse of notation, given a specification F, a state $s \in S[\mathcal{V}]$, and $\mathcal{V}_F \subseteq \mathcal{V}$, we write $c_F(s)$ for $c_F(s|_{\mathcal{V}_F})$. We illustrate these notions giving a simple example.

Example 2 *The circuit in Figure 1 implements an AND gate as the composition of a NAND gate* $I_1 = \langle\{x, y, u\}, B_1\rangle$ *and an inverter* $I_2 = \langle\{u, z\}, B_2\rangle$. *This means that* B_1 *consists of all valuations satisfying* $u = \overline{x \cdot y}$. *and* B_2 *contains all valuations satisfying* $z = \bar{u}$. *As their costs, we consider the worst-case time-to-settle, that is, the maximal time it takes for changes at the input ports to propagate to the output ports. Since the time-to-settle of this system is the sum of the settle times of the components, we interpret* $\oplus$ *as* $+$. *In both gates, raising edges take at most 35ns to settle and falling edges take 33ns. This means that if* $u = 0$ *in the first gate* I_1, *then the worst time-to-settle is 33. Either there has been a change from high voltage to low voltage, or there was no change at all, in which case the settle time is 0. Thus the cost specifications*

1 Often, a system model distinguishes between input and output variables. However, no such distinction is necessary for the development of the theory presented in this paper.

2 One easily shows that $\mathit{Hide}_{x_1}(\mathit{Hide}_{x_2}(I)) = \mathit{Hide}_{x_2}(\mathit{Hide}_{x_1}(I))$.

$F_1 = \langle\{x,y,u\},c_1\rangle$ *for* I_1, *and* $F_2 = \langle\{u,z\},c_2\rangle$ *for* I_2, *are given by*

$$c_1(x,y,u) = \begin{cases} 33 & \textit{if } u = 0, \\ 35 & \textit{if } u = 1, \end{cases} \quad c_2(u,z) = \begin{cases} 33 & \textit{if } z = 0, \\ 35 & \textit{if } z = 1. \end{cases}$$

The composition $I_1 \| I_2$ *is* $\langle\{x,y,z,u\},B\rangle$, *where* B *contains all valuations over* $\{x,y,z,u\}$ *satisfying* $u = \overline{x \cdot y}$ *and* $z = \bar{u}$. *The composition* $F_1 \| F_2 = \langle\{x,y,z,u\},c\rangle$ *is*

$$c(x,y,z) = \begin{cases} 66 & \textit{if } u = z = 0, \\ 70 & \textit{if } u = z = 1, \\ 68 & \textit{otherwise.} \end{cases}$$

Further, if we hide the variable u *in the composition, then* $\textit{Hide}_u(I_1 \| I_2)$ *is* $\langle\{x,y,z\},B_{xyz}\rangle$, *where* B_{xyz} *contains all valuations with* $z = x \cdot y$. *Similarly,* $\textit{Hide}_u(F_1 \| F_2)$ *is given by* $\langle\{x,y,z\},c_{xyz}\rangle$, *with* $c_{xyz}(x,y,z) = 66$ *if* $z = 0$, *and* 68 *if* $z = 1$.

3. Refinement

We introduce two notions of refinement: refinement between a system and a specification, and refinement between specifications. The refinement between a system and a specification measures the cost of a system according to a specification; the refinement between two specifications measures the relative cost difference between the specifications.

Refinement between systems and specifications. Given a system $I = \langle\mathcal{V}_I, B_I\rangle$ and a specification $F = \langle\mathcal{V}_F, c_F\rangle$ with $\mathcal{V}_F \subseteq \mathcal{V}_I$, we define the refinement distance $d(I,F)$ between I and F as follows:

$$d(I,F) = \sup_{s \in B_I} c_F(s).$$

Refinement between specifications. Given two specifications F and G with $\mathcal{V}_G \subseteq \mathcal{V}_F$, we define the relative refinement distance $r(F,G)$ between F and G as follows:

$$r(F,G) = \sup_{s \in S[\mathcal{V}_F]} \frac{c_G(s)}{c_F(s)}.$$

Refinement hierarchies. The following theorem provides a form of triangular-inequality law formulated for relative distances. This law is the quantitative analogue of the transitivity of boolean refinement.

Theorem 1 *Consider three specifications* F, G, *and* H, *and a system* I, *such that* $\mathcal{V}_H \subseteq \mathcal{V}_G \subseteq \mathcal{V}_F \subseteq \mathcal{V}_I$ *and* $\mathcal{V}_H \neq \emptyset$ *and* $B_I \neq \emptyset$. *The following two assertions hold:*

$$r(F,H) \leq r(F,G) \cdot r(G,H),$$
$$d(I,G) \leq d(I,F) \cdot r(F,G).$$

Proof. We have

$$\begin{aligned} r(F,G) \cdot r(G,H) &= \sup_{s \in S[\mathcal{V}_F]} \frac{c_G(s)}{c_F(s)} \cdot \sup_{s \in S[\mathcal{V}_G]} \frac{c_H(s)}{c_G(s)} \\ &\geq \sup_{s \in S[\mathcal{V}_F]} \tfrac{c_H(s)}{c_F(s)} = r(F,H); \end{aligned}$$

the inequality follows from $\mathcal{V}_G \subseteq \mathcal{V}_F$. Moreover,

$$\begin{aligned} d(I,F) \cdot r(F,G) &= \sup_{s \in B_I} c_F(s) \cdot \sup_{s \in S[\mathcal{V}_F]} \frac{c_G(s)}{c_F(s)} \\ &\geq \sup_{s \in B_I} c_G(s) = d(I,G). \end{aligned}$$ ∎

Compositionality of refinement. To prove the compositionality of refinement, we will need the following lemma.

Lemma 1 *Let* $a,b,c,d \in \mathbb{R}_{\geq 0}$ *and* $c,d \neq 0$. *Then* $(a \oplus b)/(c \oplus d) \leq \max\{a/c, b/d\}$.

Proof. Let $a = \alpha \cdot c$ and $b = \beta \cdot d$. Without loss of generality let $\alpha \geq \beta$. Then we have

$$\begin{aligned} \frac{a \oplus b}{c \oplus d} = \frac{\alpha \cdot c \oplus \beta \cdot d}{c \oplus d} \leq \frac{\alpha \cdot c \oplus \alpha \cdot d}{c \oplus d} \leq \frac{\alpha \cdot (c \oplus d)}{c \oplus d} \leq \alpha \\ = \max\{a/c, b/d\}. \end{aligned}$$

The second inequality follows from subdistributivity of $\oplus$ over multiplication. ∎

The next theorem states the compositionality of refinement.

Theorem 2 *Consider four specifications* F, F', G, *and* G' *with* $\mathcal{V}_{F'} \subseteq \mathcal{V}_F$ *and* $\mathcal{V}_{G'} \subseteq \mathcal{V}_G$, *and two systems* I *and* J *with* $\mathcal{V}_F \subseteq \mathcal{V}_I$, $\mathcal{V}_G \subseteq \mathcal{V}_J$, $\mathcal{V}_{F'} \neq \emptyset$, *and* $\mathcal{V}_{G'} \neq \emptyset$. *Then:*

$$r(F\|G, F'\|G') \leq \max\{r(F,F'), r(G,G')\},$$
$$d(I\|J, F\|G) \leq d(I,F) \oplus d(J,G).$$

Proof. The first assertion can be proved as follows:

$$\begin{aligned} r(F\|G, F'\|G') &= \sup_{s \in S[\mathcal{V}_F \cup \mathcal{V}_G]} \frac{c_{F'}(s) \oplus c_{G'}(s)}{c_F(s) \oplus c_G(s)} \\ &\leq \sup_{s \in S[\mathcal{V}_F \cup \mathcal{V}_G]} \max\left\{\frac{c_{F'}(s)}{c_F(s)}, \frac{c_{G'}(s)}{c_G(s)}\right\} \\ &\leq \max\left\{\sup_{s \in S[\mathcal{V}_F]} \frac{c_{F'}(s)}{c_F(s)}, \sup_{s \in S[\mathcal{V}_G]} \frac{c_{G'}(s)}{c_G(s)}\right\} \\ &= \max\{r(F,F'), r(G,G')\}. \end{aligned}$$

The first inequality follows from Lemma 1.

For the second assertion, first note that $d(I\|J,F) \leq d(I,F)$ and $d(I\|J,G) \leq d(J,G)$. This is because $\sup_{s \in B_{I\|J}} c_F(s) \leq \sup_{s \in B_I} c_F(s)$. Hence we have

$$\begin{aligned} d(I\|J, F\|G) &= \sup_{s \in B_{I\|J}} \big(c_F(s) \oplus c_G(s)\big) \\ &\leq \sup_{s \in B_{I\|J}} c_F(s) \oplus \sup_{s \in B_{I\|J}} c_G(s) \\ &= d(I\|J,F) \oplus d(I\|J,G) \\ &\leq d(I,F) \oplus d(J,G). \end{aligned}$$

The first inequality follows from the monotonicity of $\oplus$. ∎

We note that the theorem is not symmetric with respect to r and d: implementation refinements d are composed with $\oplus$, while specification refinements are composed with $\max$. In particular, the theorem does not hold if in the second assertion we replace $\oplus$ with $\max$. It may also be noted that given systems with boolean variables; specifications that assign cost 1 to states with desired values of variables and assign cost 2 otherwise; and with $\max$ as the combination operators we can obtain the boolean theory as a special case of the present quantitative theory.

Example 3 *Consider the system defined in Example 2. The cost of I_1 and I_2 are given by $d(I_1, F_1) = d(I_2, F_2) = 35$, indicating the maximum cost for any state. We also see that $d(I_1 \| I_2, F_1 \| F_2) = 68$, because the worst time-to-settle in $I_1 \| I_2$ is obtained by a falling edge and a raising edge. As stated in Theorem 2, we see that $d(I_1 \| I_2, F_1 \| F_2) = 68 \leq d(I_1, F_1) \oplus d(I_2, F_2) = 35 + 35 = 70$. That is, the costs of the components provide a bound on the cost of the composite system.*

Suppose now that we upgrade the gates by faster versions with the same behavior. Let the cost specifications of the new components be $\tilde{F}_1 = \langle \{x, y, u\}, \tilde{c}_1 \rangle$ and $\tilde{F}_2 = \langle \{u, z\}, \tilde{c}_2 \rangle$, with

$$\tilde{c}_1(x, y, u) = \begin{cases} 31 & \text{if } u = 0, \\ 32 & \text{if } u = 1, \end{cases} \quad \tilde{c}_2(u, z) = \begin{cases} 31 & \text{if } z = 0, \\ 32 & \text{if } z = 1. \end{cases}$$

Then, for $i \in \{1, 2\}$, we have $d(I_i, \tilde{F}_i) = 31$ and $r(F_i, \tilde{F}_i) = \max\{0.94, 0.91\} = 0.94$. As formulated in Theorem 1, we obtain $31 = d(I_i, \tilde{F}_i) \leq d(I_i, F_i) \cdot r(F_i, \tilde{F}_i) = 35 \cdot 0.94 = 32.8$. That is, the new worst-case settle times are bounded by the old settle times multiplied by the relative increase of the new gates with respect to the old ones. Moreover, we see that, in accordance with Theorem 2, $r(F_1 \| F_2, \tilde{F}_1 \| \tilde{F}_2) = 0.94 \leq \max\{r(F_1, \tilde{F}_1), r(F_2, \tilde{F}_2)\} = 0.94$, formalizing the intuition that the relative increase in the cost in the composite system with the new components over the old system is bounded by (in this case equal to) the maximum relative increase for any new component over its old version.

Example 4 *As a second example, consider a specification $F_1 = (\{x, y\}, c_1)$ for an inverter with input x and output y. Its cost function $c_1(x, y) = \alpha + |x + y|$ measures the distance from the perfect inverter $y = -x$, with a minimum fixed cost $\alpha > 0$. The specification F_2 for the second inverter is the same, except that it has y as input and z as output. We can implement the specification F_1 by an inverter I_1 that takes inputs from $[-1, 1]$ and realizes the behavior $y = -0.9 \cdot x$. We see that $d(I_1, F_1) = \alpha + 0.1$ measures the cost of the maximal deviation from the ideal inverter $y = -x$, which occurs when $x = 1$ and $y = -0.9$. Similarly, if we implement F_2 by an inverter I_2 taking inputs from $[-1, 1]$ and realizing $z = -1.3 \cdot y$, we obtain $d(I_2, F_2) = \alpha + 0.3$. The combined implementation $I_1 \| I_2$ is given by $y = -0.9 \cdot x$ and $z = -1.3 \cdot y = 1.22 \cdot x$. By taking $\max$ as $\oplus$, the cost function of the combined specification $F_1 \| F_2$ is given by $c(x, y, z) = \alpha + \max\{|x + y|, |y + z|\}$. As Theorem 2 predicts, we have $\alpha + 0.22 = d(I_1 \| I_2, F_1 \| F_2) \leq \max d(I_1, F_1), d(I_2, F_2) = \alpha + 0.3$.*

Hiding and refinement. The following theorem states that hiding variables of an implementation, or of a lower-level specification, does not change the distance with respect to a higher-level specification.

Theorem 3 *Let I be a system, and let F and G be two specifications such that $\mathcal{V}_G \subseteq \mathcal{V}_I$, and $\mathcal{V}_G \subseteq \mathcal{V}_F$, and $\mathcal{V}_G \neq \emptyset$. Consider a variable x such that if $x \in \mathcal{V}_I$, then $x \notin \mathcal{V}_G$; and if $x \in \mathcal{V}_F$, then $x \notin \mathcal{V}_G$. Consider a variable $y \in Vars$. The following assertions hold:*

$$d(\mathit{Hide}_x(I), G) = d(I, G), \tag{1}$$
$$r(\mathit{Hide}_x(F), G) = r(F, G), \tag{2}$$
$$d(I, \mathit{Hide}_y(G)) \leq d(I, G), \tag{3}$$
$$r(F, \mathit{Hide}_y(G)) \leq r(F, G). \tag{4}$$

Parts (1) and (2) of Theorem 3 are basic requirements of a compositional theory. These results require that hiding of a variable x for specifications is defined as the $\inf$ over the values of x; they do not hold if $\sup$ is used in place of $\inf$. The theorem leads to the following corollary, stating that hiding the same variable in a specification and a system, or in two specifications, yields a lesser or equal distance.

Corollary 1 *Consider an implementation I and two specifications F, G with $\mathcal{V}_G \subseteq \mathcal{V}_F$, and $\mathcal{V}_G \subseteq \mathcal{V}_I$, and $\mathcal{V}_G \neq \emptyset$. For all $x \in Vars$, the following two assertions hold:*

$$d(\mathit{Hide}_x(I), \mathit{Hide}_x(G)) \leq d(I, G),$$
$$r(\mathit{Hide}_x(F), \mathit{Hide}_x(G)) \leq r(F, G).$$

Example 5 *For Example 2, we have*

$$\begin{aligned} d(\mathit{Hide}_u(I_1 \| I_2), &\mathit{Hide}_u(F_1 \| F_2)) = 66 \\ &\leq d(I_1 \| I_2, F_1 \| F_2)) = 68. \end{aligned}$$

From Corollary 1 it follows that, given a sequence of specifications $G_0, G_1, \ldots, G_n$, from the most concrete to the most abstract, when approximating the distance $r(G_0, G_n)$ via the product of the stepwise refinement distances, as in $r(G_0, G_n) \leq \prod_{i=0}^{n-1} r(G_i, G_{i+1})$, we obtain a tighter bound by first hiding from $G_0, \ldots, G_{n-1}$ all the variables that do not appear in the most abstract specification G_n. More precisely, for $1 \leq i < n$, let $\mathcal{W}_i = \mathcal{V}_{G_i} \setminus \mathcal{V}_{G_n}$ be the variables present in G_i but not in G_n, and let $G'_i = \mathit{Hide}_{\mathcal{W}_i}(G_i)$. We have $r(G_0, G_n) \leq \prod_{i=0}^{n-1} r(G'_i, G'_{i+1}) \leq \prod_{i=0}^{n-1} r(G_i, G_{i+1})$. A similar result holds for refinement hierarchies in which the most concrete element is a system, rather than a specification.

Example 6 *To obtain an identity gate from the inverters in Example 4, we hide the variable y in both $I_1 \| I_2$ and $F_1 \| F_2$. We write $I = Hide_y(I_1 \| I_2)$ and $F = Hide_y(F_1 \| F_2)$. Since the value of y that minimizes $\alpha + \max\{|x+y|, |y+z|\}$ is $y = -(x+z)/2$, the cost function c_F is $\alpha + |z-x|/2$, which represents the distance from the identity gate $z = x$. Theorem 3 shows that $\alpha + 0.11 = d(I, F) \leq d(I_1 \| I_2, F_1 \| F_2) = \alpha + 0.22$.*

4. Symbolic Cost Algebras

Static systems and specifications are infinite semantic objects: even when variables have finite domains, there is still an uncountable number of possible cost functions. In this section, we introduce *symbolic cost algebras*, which are symbolic representations for restricted classes of systems and cost functions that are closed under composition and hiding. We also present algorithms that operate on such symbolic representations, and compute refinement distances, compositions, and the result of hiding variables.

A *cost algebra* $\langle Vars, \mathcal{B}, \mathcal{F}, \oplus, [\![\cdot]\!]\rangle$ consists of a set *Vars* of typed variables, a countable set of symbolic system representations $\mathcal{B}$, a countable set of symbolic cost representations $\mathcal{F}$, a computable combination operator $\oplus$, and of a semantics $[\![\cdot]\!]$. Each element of $\mathcal{B}$ and $\mathcal{F}$ must have a finite representation. The semantics $[\![\cdot]\!]$ maps symbolic representations to systems and specifications: specifically, for $\mathbf{B} \in \mathcal{B}$, $[\![\mathbf{B}]\!]$ is a system, and for $\mathbf{F} \in \mathcal{F}$, $[\![\mathbf{F}]\!]$ is a specification. We require that the cost algebra is effectively closed with respect to composition and hiding:

- For all $\mathbf{B}, \mathbf{B}' \in \mathcal{B}$, there is $\mathbf{B}'' \in \mathcal{B}$ effectively computable from $\mathbf{B}$ and $\mathbf{B}'$ such that $[\![\mathbf{B}]\!] \| [\![\mathbf{B}']\!] = [\![\mathbf{B}'']\!]$; for all $\mathbf{F}, \mathbf{F}' \in \mathcal{F}$, there is $\mathbf{F}'' \in \mathcal{F}$ effectively computable from $\mathbf{F}$ and $\mathbf{F}'$ such that $[\![\mathbf{F}]\!] \| [\![\mathbf{F}']\!] = [\![\mathbf{F}'']\!]$.
- For all $\mathbf{B} \in \mathcal{B}$ and all $x \in Vars$, there is $\mathbf{B}' \in \mathcal{B}$ effectively computable from $\mathbf{B}$ and x such that $Hide_x([\![\mathbf{B}]\!]) = [\![\mathbf{B}']\!]$; for all $\mathbf{F} \in \mathcal{F}$ and all $x \in Vars$, there is $\mathbf{F}' \in \mathcal{F}$ effectively computable from $\mathbf{F}$ and x such that $Hide_x([\![\mathbf{F}]\!]) = [\![\mathbf{F}']\!]$.

4.1. Maximum of Linear Terms Algebra

Many interesting examples of static cost systems involve numerical variables representing physical measures, and convex utility functions encoded by piecewise linear concave cost functions that assign higher cost to systems whose behaviors are farther away from a set of desired values for the variables. As an example of a symbolic cost algebra which is geared towards such systems, we consider the *maximum of linear terms algebra* (maxlin algebra, in short). In this cost algebra, we fix a countable set of real-valued variables *Vars* so that $D_x = \mathbb{R}$ for all $x \in Vars$. For a finite set $\mathcal{V} \subseteq Vars$, a *linear term* γ over $\mathcal{V}$ is an expression of the form $b + \sum_{x \in \mathcal{V}} x \cdot a_x$, where $b \in \mathbb{Q}$ and $a_x \in \mathbb{Q}$ for $x \in \mathcal{V}$. For $s \in S[\mathcal{V}]$, we write $\gamma(s)$ for the value of the linear term when each $v \in \mathcal{V}$ takes the real value $s(v)$. We denote by $Lin(\mathcal{V})$ the set of all linear terms over $\mathcal{V}$. Given a set Φ of linear terms over $\mathcal{V}$, we write $Poly(\Phi) = \bigcap_{\gamma \in \Phi} \{s \in S[\mathcal{V}] \mid \gamma(s) \geq 0\}$ for the convex polyhedron defined by the inequalities $\gamma \geq 0$.

Maxlin systems and specifications. Let $A \subseteq_{fin} B$ denote that A is a finite subset of B. A *maxlin system* is a pair $\langle \mathcal{V}, \Phi \rangle$ consisting of a finite set $\mathcal{V} \subseteq_{fin} Vars$ of variables and a finite set $\Phi \subseteq_{fin} Lin(\mathcal{V})$ of linear terms over $\mathcal{V}$. The semantics is defined by $[\![\langle \mathcal{V}, \Phi \rangle]\!] = \langle \mathcal{V}, Poly(\Phi) \rangle$.

We say that a set $\Omega \subseteq_{fin} Lin(\mathcal{V})$ is *positive* if for every $s \in S[\mathcal{V}]$, there is $\gamma \in \Omega$ with $\gamma(s) > 0$. A *maxlin specification* is a pair $\langle \mathcal{V}, \Omega \rangle$, where $\mathcal{V} \subseteq_{fin} Vars$, and $\Omega \subseteq_{fin} Lin(\mathcal{V})$, and Ω is positive. We let $Max_\Omega(s) = \max\{\gamma(s) \mid \gamma \in \Omega\}$ for all $s \in S[\mathcal{V}]$, and $[\![\langle \mathcal{V}, \Omega \rangle]\!] = \langle \mathcal{V}, Max_\Omega \rangle$; it is easy to see that Max_Ω is bounded away from 0. As a remark, notice that the factory example from the introduction, the time-to-settle example (Example 2), and the inverter example (Example 4) can be modeled in the maxlin algebra, as their cost functions only involve absolute value and linear functions.

Maxlin systems and specifications can be concretely represented by listing the coefficients of the linear terms that define them. For a maxlin system I and specification F, we write $|I|$ and $|F|$ for the length of such coefficient lists. We assume that arithmetic operations on rationals are performed in constant time, so that our complexity results essentially measure the number of arithmetic operations.

For the maxlin algebra, the combinator $\oplus$ can be either $\max$ or $+$. The algebra is obviously closed under composition when $\oplus$ is $\max$. To see that it is also closed when $\oplus$ is $+$, it suffices to note that, for $\gamma_1, \ldots, \gamma_n, \gamma'_1, \ldots, \gamma'_m \in Lin(\mathcal{V})$, we have

$$\max\{\gamma_1, \ldots, \gamma_n\} + \max\{\gamma'_1, \ldots, \gamma'_m\} = \max\{\gamma_i + \gamma'_j \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq m\}.$$

We next show that the maxlin algebra is also effectively closed under hiding.

Hiding in a maxlin system. To hide the variable x in a maxlin system $I = \langle \mathcal{V} \cup \{x\}, \Phi \rangle$, we proceed as follows [11]. For each $\phi \in \Phi$, we examine the constraint $\phi \geq 0$. Let a_x be the coefficient of x in ϕ. There are three possible cases. Let Φ_0, Φ_-, and Φ_+ be the least sets such that:

- If $a_x = 0$, then Φ_0 contains ϕ.
- If $a_x > 0$, then the constraint can be put in the form $x \geq \delta$, where $\delta = -b/a_x - \sum_{y \in \mathcal{V}} y \cdot a_y / a_x$. In this case, the set Φ_- contains δ.

- If $a_x < 0$, then the constraint can be put in the form $x \leq \delta$, where $\delta = -b/a_x - \sum_{y \in \mathcal{V}} y \cdot a_y/a_x$. In this case, the set Φ_+ contains δ.

Consider the set $\Phi' = \Phi_0 \cup \{\delta_+ - \delta_- \mid \delta_- \in \Phi_-$ and $\delta_+ \in \Phi_+\}$. Then, we can show that

$$Hide_x([\![\langle \mathcal{V} \cup \{x\}, \Phi \rangle]\!]) = [\![\langle \mathcal{V} \setminus \{x\}, \Phi' \rangle]\!].$$

Moreover, computing Φ' requires time quadratic in $|I|$.

Theorem 4 *For every maxlin system $I = \langle \mathcal{V} \cup \{x\}, \Phi \rangle$, a maxlin system J such that $[\![J]\!] = Hide_x([\![I]\!])$ can be computed in time quadratic in $|I|$.*

We write $Hide_x(\langle \mathcal{V} \cup \{x\}, \Phi \rangle)$ for the maxlin system $\langle \mathcal{V} \setminus \{x\}, \Phi' \rangle$.

Hiding in a maxlin specification. For the sake of simplicity, we restrict ourselves to maxlin specifications whose defining constraints have all nonzero coefficients. Let $F = \langle \mathcal{V} \cup \{x\}, \Omega \rangle$ be such a specification, where $x \notin \mathcal{V}$ and $\Omega = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$. To show that the maxlin specifications are effectively closed under hiding, we need some preliminary definitions.

Given a pair $(i, j) \in \{1, \ldots, n\}^2$ of indices, we say that they form a *counter-slope pair* if the x coefficient is negative in γ_i and positive in γ_j. For a counter-slope pair (i, j), and $s \in S[\mathcal{V}]$, let $x_{i,j}(s)$ be the unique value of x that satisfies $\gamma_i(s, x) = \gamma_j(s, x)$. For $s \in S[\mathcal{V}]$, let $x(s)$ denote the value of d that minimizes $Max_\Omega(s \circ [x \mapsto d])$, that is, $x(s) = \arg\min_{d \in \mathbb{Q}} Max_\Omega(s \circ [x \mapsto d])$. Finally, let $y(s) = Max_\Omega(s \circ [x \mapsto x(s)])$.

Lemma 2 *For every state $s \in S[\mathcal{V}]$, there is a counter-slope pair (i, j) such that $x(s) = x_{i,j}(s)$ and $y(s) = \gamma_i(s, x_{i,j}(s))$.*

Intuitively, the lemma states that the point $(x(s), y(s))$ is always located at the intersection of two counter-slope lines. This can be shown by simple arguments regarding minimizing a linear function over a convex polyhedron.

Lemma 3 *Let (i, j) and (i', j') be two counter-slope pairs. For every state $s \in S[\mathcal{V}]$, if $\gamma_i(s, x_{i,j}(s)) < \gamma_{i'}(s, x_{i',j'}(s))$, then there is a $k' \in \{i', j'\}$ such that $\gamma_i(s, x_{i,j}(s)) < \gamma_{k'}(s, x_{i,j}(s))$.*

Intuitively, this lemma states that if the intersection point between two counter-slope lines is lower than the intersection point of another pair, the first point lies beneath one of the defining lines. This can be shown by a simple case analysis.

Theorem 5 *For every maxlin specification $F = \langle \mathcal{V} \cup \{x\}, \Omega \rangle$, a maxlin specification G such that $[\![G]\!] = Hide_x([\![F]\!])$ can be computed in time quadratic in $|F|$.*

Proof. We show that $y(s) = \max\{\gamma_i(s, x_{i,j}(s)) \mid (i, j)$ is a counter-slope pair$\}$. Lemma 2 shows that $y(s)$ is one of the elements in the set on the right-hand side. It remains to be shown that it is the maximal element. Assume that $y(s) = \gamma_i(s, x_{i,j}(s))$ and $x(s) = x_{i,j}(s)$. For all $k = 1, \ldots, n$, we know that $y(s) \geq \gamma_k(s, x(s))$. Now, if $\gamma_i(s, x_{i,j}(s))$ is smaller than the maximum, by Lemma 3, there is $k \in \{1, \ldots, n\}$ such that $\gamma_i(s, x_{i,j}(s)) < \gamma_k(s, x_{i,j}(s))$. We then obtain the contradiction: $y(s) = \gamma_i(s, x_{i,j}(s)) < \gamma_k(s, x_{i,j}(s)) = \gamma_k(s, x(s))$. Intuitively, the contradiction is that the point $(x(s), y(s))$ lies beneath the line γ_k.

Thus, to compute the representation of $Hide_x([\![F]\!])$ for a maxlin specification $F = \langle \mathcal{V}, \{\gamma_1, \ldots, \gamma_n\} \rangle$, we check each pair of indices $(i, j) \in \{1, \ldots, n\}^2$. If (i, j) form a counter-slope pair, we compute the coefficients of $\gamma_i(s, x_{i,j}(s))$, and we add the corresponding linear term to the result. This procedure terminates in time quadratic in $|F|$. ∎

4.2. Computing Refinement Distances

Refinement between a maxlin system and a maxlin specification. Consider a maxlin system $I = [\![\langle \mathcal{V}, \Phi \rangle]\!]$ and a maxlin specification $F = [\![\langle \mathcal{V}', \Omega \rangle]\!]$. For simplicity we present algorithms for the case when $\mathcal{V} = \mathcal{V}'$; they can be extended to the general case when $\mathcal{V} \neq \mathcal{V}'$. Let $\Omega = \{\gamma_1, \ldots, \gamma_n\}$; we have $Max_\Omega(s) = \max\{\gamma_i(s) \mid 1 \leq i \leq n\}$. The refinement distance $d(I, F) = \sup_{s \in \Phi} Max_\Omega(s)$ can be computed by solving the following n linear-programming problems. For all $1 \leq i \leq n$, let

$$\begin{array}{l} m_i = \max\ \gamma_i; \\ \text{subject to the constraint set } \Phi. \end{array}$$

Then $d(I, F) = \max\{m_i \mid 1 \leq i \leq n\}$.

Refinement between two maxlin specifications. Consider two maxlin specifications $F = [\![\langle \mathcal{V}, \Omega \rangle]\!]$ and $F' = [\![\langle \mathcal{V}', \Omega' \rangle]\!]$. As before, we can assume without loss of generality that $\mathcal{V} = \mathcal{V}'$. Let $\Omega = \{\gamma_1, \ldots, \gamma_n\}$ with $Max_\Omega(s) = \max\{\gamma_j(s) \mid 1 \leq j \leq n\}$; and $\Omega' = \{\gamma'_1, \ldots, \gamma'_m\}$ with $Max_{\Omega'}(s) = \max\{\gamma'_i(s) \mid 1 \leq i \leq m\}$. To compute $r(F, F')$ we consider the following $n \cdot m$ optimization problems. For all $1 \leq i \leq m$ and $1 \leq j \leq n$, let

$$\begin{array}{lll} m_{ij} = \max\ \dfrac{\gamma'_i}{\gamma_j}; & & \\ \text{subject to} & \gamma'_i \geq \gamma'_k & \text{for all } 1 \leq k \leq m, \\ & \gamma_j \geq \gamma_l & \text{for all } 1 \leq l \leq n. \end{array}$$

Intuitively, if the pair $\gamma'_i \in \Omega'$ and $\gamma_j \in \Omega$ maximizes the value of $r(F, F')$, then m_{ij} calculates the corresponding value. Hence we have $r(F, F') = \max\{m_{ij} \mid 1 \leq i \leq m$ and $1 \leq j \leq n\}$. In the optimization problems the objective function to maximize is a ratio of two linear functions.

Such objective functions can be easily reduced to linear objectives as follows. The optimization problem

$$\max \frac{\vec{c}^T\vec{x}+\vec{d}}{\vec{e}^T\vec{x}+\vec{f}};$$
$$\text{subject to } G\vec{x} \le \vec{h} \text{ and } A\vec{x}=\vec{b}$$

can be reduced to an equivalent linear-programming problem described as:

$$\max \vec{c}^T\vec{y}+\vec{d}\vec{z};$$
$$\text{subject to } \begin{cases} G\vec{y}-\vec{h}\vec{z} \le 0 \text{ and } A\vec{y}-\vec{b}\vec{z}=0, \\ \vec{e}^T\vec{y}+\vec{f}\vec{z}=1 \text{ and } \vec{z}\ge 0. \end{cases}$$

The equivalence of the two optimization problems is easy to establish by letting $\vec{y}=\frac{\vec{x}}{\vec{e}^T\vec{x}+\vec{f}}$ and $\vec{z}=\frac{1}{\vec{e}^T\vec{x}+\vec{f}}$. Since linear programming is solvable in polynomial time [17] we have proved the following result.

Theorem 6 *For a maxlin system I and two maxlin specifications F and G, the refinement distances $d([\![I]\!],[\![F]\!])$ and $r([\![F]\!],[\![G]\!])$ can be computed in time polynomial in the size of the inputs.*

5. Dynamic Systems and Specifications

So far, we have developed the theory for the static context, where a system is an assignment to variables. We now extend our theory to the dynamic context, where behaviors are infinite temporal sequences of valuations, and specifications assign costs to temporal sequences.

Dynamic systems. A *dynamic system* $I=\langle L,\mathcal{V}_I,B_I,\delta_I\rangle$ consists of a set L of locations, a set $\mathcal{V}_I$ of variables, a function $B_I : L \to 2^{S[\mathcal{V}_I]}$ that assigns for every location $l \in L$ a subset $B_I(l) \subseteq S[\mathcal{V}_I]$ of states over $\mathcal{V}_I$, and a transition relation $\delta_I \subseteq L \times L$. We omit the subscript I when clear from the context. We define composition and hiding for dynamic systems as follows.

- **Composition.** Given two dynamic systems I and J, their *composition* $I\|J$ is defined by $\mathcal{V}_{I\|J}=\mathcal{V}_I\cup\mathcal{V}_J$, $L_{I\|J}=L_I\times L_J$, $B_{I\|J}: L_{I\|J}\to 2^{S[\mathcal{V}_{I\|J}]}$ such that $B_{I\|J}(l_1,l_2)=\{s\in S[\mathcal{V}_{I\|J}] \mid s|_{\mathcal{V}_I}\in B_I(l_1)\wedge s|_{\mathcal{V}_J}\in B_J(l_2)\}$, and $\delta_{I\|J}=\{\langle (l_1,l_2),(l_1',l_2')\rangle \mid \langle l_1,l_1'\rangle \in \delta_I \wedge \langle l_2,l_2'\rangle\in\delta_J\}$.
- **Hiding.** Given a system I and a variable $x\in$ *Vars*, we define the result $\mathit{Hide}_x(I)$ of *hiding* x in I by I itself if $x\notin\mathcal{V}_I$, and otherwise by $J=\langle\mathcal{V}_J,B_J\rangle$, where $\mathcal{V}_J=\mathcal{V}_I\setminus\{x\}$ and $B_J(l)=\{s\in S[\mathcal{V}_J]\mid \exists d\in D_x.(s\circ[x\mapsto d])\in B_I(l)\}$. For $X=\{x_1,x_2,\ldots,x_n\}\subseteq$ *Vars*, we define $\mathit{Hide}_X(I)=\mathit{Hide}_{x_1}(\mathit{Hide}_{x_2}(\cdots\mathit{Hide}_{x_n}(I)))$.

Dynamic specifications. A *dynamic specification* $F=\langle L,\mathcal{V}_F,c_F,\delta_F\rangle$ consists of a set L of locations, a set $\mathcal{V}_F$ of variables, a function $c_F(l): S[\mathcal{V}_F]\mapsto \mathbb{R}_{\ge 0}$ that assigns to every location l and state $s\in S[\mathcal{V}_F]$ a real number $c_F(l)(s)\in\mathbb{R}_{\ge 0}$, and a transition relation $\delta\subseteq L\times L$. The value $c_F(l)(s)$ can be interpreted as the cost of realizing s at location l. We require that the cost be bounded away from 0: for each specification F, there is a constant $b\in\mathbb{R}_{\ge 0}$ such that $c_F(l)(s)\ge b$ for all states $s\in S[\mathcal{V}_F]$ and locations $l\in L$. We omit the subscript F when clear from the context. For a fixed combination operator $\oplus$, we define hiding and composition of specifications as follows.

- **Composition.** Given two specifications F and G, we define their *composition* $F\|G$ by $\mathcal{V}_{F\|G}=\mathcal{V}_F\cup\mathcal{V}_G$, where the location and transition relation is obtained as the usual synchronous product (as in the case of dynamic systems), and for all $s\in S[\mathcal{V}_{F\|G}]$, we have $c_{F\|G}(l_1,l_2)(s)=c_F(l_1)(s|_{\mathcal{V}_F})\oplus c_G(l_2)(s|_{\mathcal{V}_G})$.
- **Hiding.** Given a specification F and a variable $x\in$ *Vars*, we define the result $\mathit{Hide}_x(F)$ of *hiding* x in F by F itself if $x\notin\mathcal{V}_F$, and otherwise by $G=\langle\mathcal{V}_G,c_G\rangle$, where $\mathcal{V}_G=\mathcal{V}_F\setminus\{x\}$ and, for all $s\in S[\mathcal{V}_G]$, by $c_G(l)(s)=\inf_{d\in D_x} c_F(l)(s\circ[x\mapsto d])$. In other words, $\mathit{Hide}_x(F)$ is equal to the minimum of F over all values of x. For $X=\{x_1,x_2,\ldots,x_n\}\subseteq$ *Vars*, we define $\mathit{Hide}_X(F)=\mathit{Hide}_{x_1}(\mathit{Hide}_{x_2}(\cdots\mathit{Hide}_{x_n}(F)))$.

Traces and trace distances. A trace τ_I of a dynamic system I is an infinite sequence $\langle l_0,l_1,l_2,\ldots\rangle$ of locations such that $(l_i,l_{i+1})\in\delta_I$ for all $i\ge 0$. A trace τ_F of a dynamic specification F is defined similarly. Given a dynamic system I and a dynamic specification F, we denote by $\mathcal{T}_I$ and $\mathcal{T}_F$ the set of all traces of I and F, respectively. Given a system I, and two specifications F and G with $\mathcal{V}_G\subseteq\mathcal{V}_F\subseteq\mathcal{V}_I$, let $\tau_I=\langle l_0,l_1,l_2,\ldots\rangle$, $\tau_F=\langle\widehat{l}_0,\widehat{l}_1,\widehat{l}_2,\ldots\rangle$, and $\tau_G=\langle\widetilde{l}_0,\widetilde{l}_1,\widetilde{l}_2,\ldots\rangle$ be traces of I, F, and G, respectively. The trace distances are defined as follows:

$$td(\tau_I,\tau_F)=\sup_{i\in\mathbb{N}}\sup_{s\in B_I(l_i)} c_F(\widehat{l}_i)(s),$$
$$td(\tau_F,\tau_G)=\sup_{i\in\mathbb{N}}\sup_{s\in S[\mathcal{V}_F]}\frac{c_G(\widetilde{l}_i)(s)}{c_F(\widehat{l}_i)(s)}.$$

Note that for each location $l\in L_I$, the tuple $I(l)$ defined by $I(l)=\langle\mathcal{V}_I,B_I(l)\rangle$ is a static system. Similarly, for each location $l\in L_F$, the tuple $F(l)$ defined by $F(l)=\langle\mathcal{V}_F,c_F(l)\rangle$ is a static specification, and we have

$$td(\tau_I,\tau_F)=\sup_{i\in\mathbb{N}} d(I(l_i),F(\widehat{l}_i)),$$
$$td(\tau_F,\tau_G)=\sup_{i\in\mathbb{N}} r(F(\widehat{l}_i),G(\widetilde{l}_i)).$$

We illustrate the theory of dynamic systems and specifications with a simple, theoretical example. To make the cal-

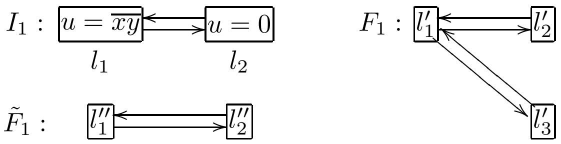

Figure 2. Dynamic system I_1 and specifications F_1 and $\tilde{F}_1$.

culations easier, we reuse some material from previous examples.

Example 7 *Figure 2 displays the dynamic system $I_1 = \langle L_1, \{x, y, u\}, B_1, \delta_1 \rangle$, and the dynamic specification $F_1 = \langle L'_1, \{x, y, u\}, c_1, \delta'_1 \rangle$, where*

$$c_1(l'_1)(x, y, u) = \begin{cases} 33 & \text{if } u = 0, \\ 35 & \text{if } u = 1, \end{cases}$$
$$c_1(l'_2)(x, y, u) = \begin{cases} 31 & \text{if } u = 0, \\ 34 & \text{if } u = 1, \end{cases}$$
$$c_1(l'_3)(x, y, u) = 38.$$

From Example 4, we know that $d(I_1(l_1), F_1(l'_1)) = 35$. Similarly, we have $d(I_1(l_2), F_1(l'_2)) = 31$ and $d(I_1(l_2), F_1(l'_3)) = 38$. Thus, the distance from the trace $\tau = l_1 l_2 l_1 l_2 \ldots$ of I_1 to the trace $\tau_2 = l'_1 l'_2 l'_1 l'_2 \ldots$ of F_1 is

$$td(\tau, \tau_2) = \sup_{i=1,2} d(I_1(l_i), F_1(l'_i)) = \max(35, 31) = 35.$$

Similarly, the distance from τ to the trace $\tau_3 = l'_1 l'_3 l'_1 l'_3 \ldots$ of F_1 is

$$td(\tau, \tau_3) = \sup_{i=1,3} d(I_1(l_i), F_1(l'_i)) = \max(35, 38) = 38.$$

Refinement between dynamic systems and specifications. Given a dynamic system I and a dynamic specification F with $\mathcal{V}_F \subseteq \mathcal{V}_I$, we define the refinement distance $d(I, F)$ between I and F by

$$d(I, F) = \sup_{\tau_I \in \mathcal{T}_I} \inf_{\tau_F \in \mathcal{T}_F} td(\tau_I, \tau_F),$$

i.e., it is the Hausdorff distance between I and F.

Refinement between dynamic specifications. Given two dynamic specifications F and G with $\mathcal{V}_G \subseteq \mathcal{V}_F$, we define the relative refinement distance $r(F, G)$ between F and G by

$$r(F, G) = \sup_{\tau_F \in \mathcal{T}_F} \inf_{\tau_G \in \mathcal{T}_G} td(\tau_F, \tau_G),$$

i.e., it is the Hausdorff distance between F and G.

Example 8 *From Example 7 we obtain that the trace τ_2 is the trace of F_1 that has the minimal distance from τ. The only other trace $\rho = l_2 l_1 l_2 l_1 \ldots$ of I_1 also has a trace of F_1 at (minimal) distance 35, namely, $\rho_2 = l'_2 l'_1 l'_2 l'_1 \ldots$ Thus, $d(I_1, F_1) = \max\{td(\tau, \tau_2), td(\rho, \rho_2)\} = 35$.*

Compositionality of refinement. We start with a technical lemma to generalize the results of compositional reasoning to dynamic systems and specifications. In the following, a and b represent sequences of real numbers, and a_i and b_i their i-th elements.

Lemma 4 *Let A, A', B, and B' be sets of sequences of real numbers bounded away from 0. Then:*

$$\sup_{a \in A} \inf_{b \in B} \sup_{i \in \mathbb{N}} \frac{a_i}{b_i} \le \left(\sup_{a \in A} \inf_{b' \in B'} \sup_{i \in \mathbb{N}} \frac{a_i}{b'_i} \right) \cdot \left(\sup_{b' \in B'} \inf_{b \in B} \sup_{i \in \mathbb{N}} \frac{b'_i}{b_i} \right).$$

$$\sup_{a \in A, a' \in A'} \inf_{b \in B, b' \in B'} \sup_{i \in \mathbb{N}} \max \left\{ \frac{a_i}{b_i}, \frac{a'_i}{b'_i} \right\}$$
$$\le \max \left\{ \sup_{a \in A} \inf_{b \in B} \sup_{i \in \mathbb{N}} \frac{a_i}{b_i}, \sup_{a' \in A} \inf_{b' \in B'} \sup_{i \in \mathbb{N}} \frac{a'_i}{b'_i} \right\}.$$

The first part of Lemma 4 and arguments similar to the proof of Theorem 1 yield Theorem 7. This theorem provides a form of triangular-inequality law for relative distances for dynamic systems and specifications.

Theorem 7 *Given three dynamic specifications F, G, and H, and a dynamic system I such that $\mathcal{V}_H \subseteq \mathcal{V}_G \subseteq \mathcal{V}_F \subseteq \mathcal{V}_I$ and $\mathcal{V}_H \neq \emptyset$ and $B_I \neq \emptyset$, the following assertions hold:*

$$r(F, H) \le r(F, G) \cdot r(G, H),$$
$$d(I, G) \le d(I, F) \cdot r(F, G).$$

Part 2 of Lemma 4 and arguments similar to the proof of Theorem 2 yield Theorem 8, which states the compositionality of refinement for dynamic systems and specifications.

Theorem 8 *Given four dynamic specifications F, F', G, and G' with $\mathcal{V}_{F'} \subseteq \mathcal{V}_F$ and $\mathcal{V}_{G'} \subseteq \mathcal{V}_G$, and two dynamic systems I and J with $\mathcal{V}_F \subseteq \mathcal{V}_I$, $\mathcal{V}_G \subseteq \mathcal{V}_J$, $\mathcal{V}_{F'} \neq \emptyset$, and $\mathcal{V}_{G'} \neq \emptyset$, we have:*

$$r(F \| G, F' \| G') \le \max\{r(F, F'), r(G, G')\},$$
$$d(I \| J, F \| G) \le d(I, F) \oplus d(J, G).$$

The analogue Theorem 3 and the corresponding corollary also generalize to dynamic systems and specifications in a straight-forward way. Furthermore, for cost functions specified in the maxlin algebra, refinement distances can be computed through quantifier elimination in the theory of reals with addition and multiplication [11].

Example 9 *Consider the second, dynamic specification $\tilde{F}_1 = \langle \tilde{L}_1, \{x, y, u\}, \tilde{c}_1, \tilde{\delta}_1 \rangle$ from Figure 2, where*

$$\tilde{c_1}(l''_1)(x, y, u) = c_1(l'_1)(x, y, u),$$
$$\tilde{c_1}(l''_2)(x, y, u) = 36.$$

For the static specifications $F_1(l'_1)$, $F_1(l'_2)$, $\tilde{F}_1(l''_1)$, $\tilde{F}_1(l''_2)$, and $\tilde{F}_1(l''_3)$, we have

$$r(F_1(l'_1), \tilde{F}_1(l''_1)) = 1,$$
$$r(F_1(l'_2), \tilde{F}_1(l''_2)) = \max\{\tfrac{36}{31}, \tfrac{36}{34}\} = \tfrac{36}{31},$$
$$r(F_1(l'_3), \tilde{F}_1(l''_2)) = \tfrac{36}{38}.$$

For a trace of F_1 starting in location l'_i, the closest trace of $\tilde{F}_1$ is the one starting at location l''_i. The traces of F_1 containing l'_2 have the largest minimal distance to F_1. For instance, the trace of $\tilde{F}_1$ closest to $l'_1 l'_2 l'_1 l'_2 \ldots$ is $l''_1 l''_2 l''_1 l''_2 \ldots$, and lies at distance $\frac{36}{31}$. Thus, $r(F_1, \tilde{F}_1) = \frac{36}{31}$.

It is easy to see that $d(I_1, \tilde{F}_1) = 36$. Thus, in accordance with Theorem 7, we have $d(I_1, \tilde{F}_1) = 36 \leq d(I_1, F_1) \cdot r(F_1, \tilde{F}_1) = 35 \cdot \frac{36}{31}$.

6. Conclusion

We focused on the basic features of a modular approach to quantitative reasoning, and illustrated them with a simple concrete symbolic cost algebra. We studied both a static setting, where behaviors are sets of variable valuations, and a dynamic setting, where behaviors are sets of sequences of variable valuations. The definitions and results of this paper can be extended to more general settings. For example, we can restrict the domains of cost functions of specifications using constraints. For the maxlin cost algebra and linear constraints, our algorithms from Section 4 generalize in a straight-forward way. Also, while we have studied a relative notion of refinement, similar results may be obtained for an absolute notion, where the distance between two specifications is the difference rather than the ratio of costs. While we have not discussed symbolic cost algebras for the dynamic setting, we can extend the maxlin algebra to use automata labeled by linear constraints and linear cost functions.

In general, we expect that when the framework proposed in this paper is applied to concrete problems, the notion of state, the cost combinator $\oplus$, and the symbolic cost algebra need to be adapted according to the semantics of the given problem. Our purpose was to illustrate the existence of compositional frameworks for quantitative reasoning and optimization, rather than advocate one specific way of assigning, composing, and refining cost functions.

References

[1] D. Bertsekas. *Dynamic Programming and Optimal Control*, vol. I and II. Athena Scientific, 1995.

[2] P. Caspi and A. Benveniste. Toward an approximation theory for computerized control. In *EMSOFT*, vol. 2491 of *Lect. Notes in Comp. Sci.*, pages 294–304. Springer, 2002.

[3] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[4] L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. In *TACAS*, vol. 2988 of *Lect. Notes in Comp. Sci.*, pages 77–92. Springer, 2004.

[5] L. de Alfaro, M. Faella, and M. Stoelinga. Linear and branching metrics for quantitative transition systems. In *ICALP*, vol. 3142 of *Lect. Notes in Comp. Sci.*, pages 97–109. Springer, 2004.

[6] L. de Alfaro, T.A. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *ICALP*, vol. 2719 of *Lect. Notes in Comp. Sci.*, pages 1022–1037. Springer, 2003.

[7] L. de Alfaro and R. Majumdar. Quantitative solution of ω-regular games. In *STOC*, pages 675–683. ACM Press, 2001.

[8] C. Derman. *Finite-State Markovian Decision Processes*. Academic Press, 1970.

[9] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labeled Markov systems. In *CONCUR*, vol. 1664 of *Lect. Notes in Comp. Sci.*, pages 258–273. Springer, 1999.

[10] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Approximating labeled Markov processes. *Info. Comp.*, 184(1):160–200, 2003.

[11] J. Ferrante and C. Rackoff. A decision procedure for the first-order theory of real addition with order. *SIAM J. Comp.*, 4(1):69–76, 1975.

[12] O. Grumberg and D. Long. Model checking and modular verification. *ACM Trans. Prog. Lang. Sys.*, 16(3):843–871, 1994.

[13] M. Huth and M. Kwiatkowska. Quantitative analysis and model checking. In *LICS*, pages 111–122. IEEE Computer Society Press, 1997.

[14] A. Lluch-Lafuente and U. Montanari. Quantitative μ-calculus and CTL defined over constraint semirings. *Theor. Comp. Sci.*, 346(1):135–160, 2005.

[15] A. McIver and C. Morgan. Games, probability, and the quantitative μ-calculus qMμ. In *LPAR*, vol. 2514 of *Lect. Notes in Comp. Sci.*, pages 292–310. Springer, 2002.

[16] R. Milner. An algebraic definition of simulation between programs. In *IJCAI*, pages 481–489. The British Computer Society, 1971.

[17] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1987.