

Quantitative μ -calculus and CTL based on constraint semirings¹

Alberto Lluch-Lafuente and Ugo Montanari

lafuente@di.unipi.it, ugo@di.unipi.it

Dipartimento di Informatica

Università di Pisa

Abstract

Model checking and temporal logics are boolean. The answer to the model checking question *does a system satisfy a property?* is either *true* or *false*, and properties expressed in temporal logics are defined over boolean propositions. While this classic approach is enough to specify and verify boolean temporal properties, it does not allow to reason about quantitative aspects of systems. Some quantitative extensions of temporal logics has been already proposed, especially in the context of probabilistic systems. They allow to answer questions like *with which probability does a system satisfy a property?*

We present a generalization of two well-known temporal logics: CTL and the μ -calculus. Both extensions are defined over c-semirings, an algebraic structure that captures many problems and that has been proposed as a general framework for soft constraint satisfaction problems (CSP). Basically, a c-semiring consists of a domain, an additive operation and a multiplicative operation, which satisfy some properties. We present the semantics of the extended logics over transition systems, where a formula is interpreted as a mapping from the set of states to the domain of the c-semiring, and show that the usual connection between CTL and μ -calculus does not hold in general. In addition, we reason about the feasibility of computing the logics and illustrate some applications of our framework, including boolean model checking.

Key words: Constraint Semirings, Constraints, Temporal Logics, Quantitative Model Checking

1 Introduction

Model Checking [9] is a formal automated method for system verification consisting of three main steps: modeling the system, modeling the properties of

¹ This work has been supported by the European Research Training Network SEGRAVIS.

the system and checking whether the properties hold in the system. Usually, systems are represented using formalisms such as automata, labeled transition systems and Kripke structures with a finite number of states, where boolean propositions are associated to states and transitions. On the other hand, temporal logics, which combine modal operators with boolean connectives and propositions, are often used for representing system properties. The verification step returns a boolean answer: either *yes* (the system satisfies the specification) or *false* (the system violates the specification).

Motivation.

While this approach is sufficient to reason about *qualitative* aspects of systems, it is clearly not sufficient to reason about *quantitative* aspects. Hence, approaches have been proposed for the analysis and verification of quantitative systems. Among others, we cite durational systems [16], probabilistic systems [13,14] and timed systems [1]. Formalisms to represent systems and properties are equipped with quantitative aspects such as time, probabilities or costs. The verification question can be boolean, but it can be quantitative too. For example, when reasoning about probabilistic system one can ask if a property holds with a probability higher than p , or which is the probability that a property holds.

A similar situation exists in the domain of Constraint Satisfaction Problems. There, classical problems consider boolean constraints only: either a constraint is present or it is not. Some CSP problems, however, cannot be properly formulated using this crisp interpretation of constraints. Several approaches exist to remedy this, which basically associate non-boolean values to constraints, like probabilities, costs and sets. Such constraints are called soft constraints and the corresponding problems are soft CSP, like fuzzy CSP, probabilistic CSP, weighted CSP, among others.

A general framework for soft CSP has been proposed in [3,4,5]. The key of the approach is an algebraic structure called *constraint semiring* (c-semirings for short). A c-semiring consists of a domain and two operations, which are called additive and multiplicative operations. The basic idea is that the domain is used to represent the values associated to constraints (boolean, probabilities, etc.), the additive operation is used to project constraints and the multiplicative operation is used to combine constraints. The framework captures most of the common soft CSP problems, such that the results stated for the framework can be applied to concrete instances. Such results can be often employed effectively on the particular instance or can be used, for example, as hints for the applicability of special solution methods for a concrete instance. More interestingly, c-semiring constraint methods have a unique advantage when problems with multiple criteria or multiple metrics must be tackled. In fact, it turns out that Cartesian products, exponentials and power constructions of c-semirings are c-semirings. Thus the same concepts and algorithms can be applied again and again.

Goals.

The purpose of our research is to define a general framework for quantitative verification, based on c -semirings. In this paper we present our first step. We have extended two well-known temporal logics, μ -calculus and CTL, where boolean propositions and connectives are replaced by c -semiring values and operations, while temporal operators are substituted by operators on sequences of c -semiring values. We present a first semantics over transition systems. While in boolean model checking the interpretation of a formula can be seen as a mapping from the set of states to the boolean domain, which implicitly represents the subset of states satisfying the formula, we interpret formulae as mappings from the set of system states to the domain of the c -semiring.

While we expect our framework to capture many problems, one of our main interests is to use it as a formalism to analyze *Quality of Service* (QoS) properties. QoS are measures of the non functional properties of services. Bandwidth, delay and jitter are typical examples of network QoS properties, while application-level QoS includes, among others, price and access rights. In most cases, the implicit algebraic structure is a c -semiring. As a matter of fact, c -semirings has been used in Kaos [15], a calculus for programmable QoS, as a formalism for representing QoS properties of WAN applications.

Related work.

Our approach is not the first attempt to define a framework for quantitative verification. For instance, the algebraic μ -calculus [2] extends the classical μ -calculus with arithmetic expressions and functions. This approach is very general and embeds a wide range of problems, from graph theoretical problems to matrix operations. Quantitative verification of systems like, for example, probabilistic and durational systems can also be performed using this framework. Our work is also inspired by previous research on quantitative analysis and verification of probabilistic systems. In [13], for instance, a probabilistic extension of the μ -calculus is proposed. The semantics is defined over probabilistic transition systems, where three alternative probabilistic semantics for disjunction and conjunction are presented. Since fixpoint iteration is infeasible, they propose an alternative approach to evaluate a significant fragment of their logic, which basically consists on reducing the problem of fixpoint computation to an (equivalent) optimization problem in linear programming. The implicit algebraic structure they use is however not a c -semiring. On the other hand, the work described in [10] defines two quantitative extensions of the μ -calculus: a probabilistic one, where disjunction and conjunction of probabilities are given a fuzzy (min/max) interpretation, and a discounted one, where events are weighted according to their distance to the present. Their semantics is defined over games, a formalism that generalizes, among others, boolean and probabilistic transition systems. A discounted version of CTL is proposed in [11], where two semantics are given: the path and fixpoints

semantics. While in boolean model checking both semantics are equivalent, allowing the use of fixpoint iterations as algorithms for CTL, they differ when discounted CTL is interpreted over probabilistic systems. Specific algorithms for each semantics are thus proposed.

Structure of the paper.

This paper is structured as follows. Section 2 gives the necessary background on c-semirings and transition systems. The next section describes syntax and semantics of the extended logics, and presents some theoretical results. Section 4 is on the computation of our logics and Section 5 illustrates various applications of our framework. The last section concludes the paper and outlines current and future work. An appendix following the bibliography contains proofs of some lemmas and theorems.

2 Preliminaries

Our logics are defined over the domain of an algebraic structure called c-semiring, where “c” stands for “constraint”, meaning that they are the natural structures to be used when handling constraints. As already explained c-semirings have been proposed as a formalism for soft constraint solving and programming problems [3,4,5].

A c-semiring is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that:

- A is a set;
- $\mathbf{0}$ and $\mathbf{1}$ are elements of A ;
- $+$: $2^A \rightarrow A$ is defined over (possibly infinite) sets of elements of A as follows²: $\sum\{a\} = a$, $\sum \emptyset = \mathbf{0}$, $\sum A = \mathbf{1}$ and $\sum(\bigcup A_i) = \sum\{\sum A_i\}$, for $A_i \subseteq A$, $i \geq 0$;
- \times : $A \times A \rightarrow A$ is a binary associative, commutative operation that distributes over $+$, has $\mathbf{1}$ as its unit element and $\mathbf{0}$ as its absorbing element.

The fact that $+$ is defined over sets of elements, automatically makes such an operation associative, commutative and idempotent. Moreover, one can show that it has $\mathbf{0}$ as unit element and $\mathbf{1}$ as absorbing element [5]. Given a sequence $a_0, a_1 \dots$ of elements of A we write $\sum_{i \geq 0} a_i$ rather than $\sum_{\{a_0, a_1 \dots\}}$. Given a finite sequence a_0, \dots, a_n we abbreviate $a_0 \times \dots \times a_n$ with $\prod_{0 \leq i \leq n} a_i$. In the rest of the paper we assume that \prod is defined over infinite sequences too. Most of the c-semirings used in practice satisfy this.

To enhance readability, operation $+$ is called *additive operation*, while \times is called *multiplicative operation*. Note that we use a boldfaced $+$ and symbol \times to avoid confusion with the additive and multiplicative operations over reals ($+$ and \cdot).

² When $+$ is applied to a set with two elements we use $+$ as binary operator in infix notation, while in all other cases we use symbol \sum in prefix notation.

Typical examples of c-semirings are $\langle \{true, false\}, \vee, \wedge, false, true \rangle$ (logical c-semiring), $\langle \mathbb{R}^+, min, +, +\infty, 0 \rangle$ (optimization c-semiring), $\langle [0, 1], max, \cdot, 0, 1 \rangle$ (probabilistic c-semiring) and $\langle [0, 1], max, min, 0, 1 \rangle$ (fuzzy c-semiring). Note that the intuition behind the operations of the c-semiring is that addition is used to represent selection of values, while multiplication represents combination of values.

The additive operation of the c-semiring induces a partial order as follows: $a \leq_S b$ iff $a + b = b$. For example, in the optimization c-semiring, \leq_S corresponds to arithmetic relation \geq . One can show that \leq_S is indeed a partial order, that $+$, \times are monotone over \leq_S , $\mathbf{0}$ and $\mathbf{1}$ are respectively the minimum and maximum elements of \leq_S , and $\langle A, \leq_S \rangle$ is a complete lattice [5].

In some instances of a c-semiring, the multiplicative operation is idempotent. This implies, among other things, that $+$ distributed over \times and $\langle A, \leq_S \rangle$ is a distributive lattice [5]. The logical and fuzzy c-semirings are an example of this, since logical disjunction is idempotent, i.e. $p \wedge p$ is p . To the contrary, the multiplicative operations of the optimization and probabilistic c-semirings, i.e. addition and multiplication of reals are not idempotent.

It is also common to have a *negation* operator $- : A \rightarrow A$, which is a unary operator such that $-a \in A$ and $--(a) = a$ for all $a \in A$, and $-\bigsqcup\{A'\} = \bigsqcap\{-a \mid a \in A'\}$ for all $A' \subseteq A$, where \bigsqcup and \bigsqcap are the *lowest upper bound* and *greatest lower bound* operators of the lattice $\langle A, \leq_S \rangle$. The negation operator allows us to use the equivalence $-\mathbf{0} = \mathbf{1}$. Note that the duality $-(a+b) = (-a) \times (-b)$ holds exactly when \times is idempotent. Examples where a negation can be defined are the logical c-semiring, where logical negation is a negation operator, and probabilistic and fuzzy c-semirings where $1-$ is a negation operator. On the other hand, it is not possible to define a negation operator for the optimization c-semiring.

As already advanced in the first section, we give an interpretation of our logics over transition systems, defined as tuples $\langle S, T \rangle$, where S is a set of states and $T \subseteq S \times S$ is a set of transitions. We assume the transition system to be *image-finite*, i.e. that for any given $s \in S$ we have that $\{s' \mid (s, s') \in T\}$ is a finite set. We sometimes require T to be total, i.e. for every state s there is at least one outgoing transition $(s, s') \in T$. This avoids end states in the system. Runs of a system are *maximal paths* in the underlying state transition graph, i.e. paths that are either infinite or end in an end state. A path is a sequence $s_0, s_1, s_2 \dots$, such that for all $i \geq 0$ we have $s_i \in S$ and $(s_i, s_{i+1}) \in T$. We denote by $|p|$ the length of a finite path p , by s_i^p the i -th state of path p and by $\gamma(s)$ the set of runs starting at s .

In the rest of the paper, let $M = \langle S, T \rangle$ and $C = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ be the transition system and c-semiring under consideration.

3 Semiring Logics

In this section we first give the syntax and semantics of the μ -calculus extension, which we call *c-semiring μ -calculus* (or *c- L_μ* for short). Later we describe the syntax of the CTL extension, called *c-semiring CTL* (or *c-CTL* for short), and give two semantics: the path and the fixpoint semantics, showing that the equivalence between the two semantics does not hold in general.

As we shall see, we interpret a formula of the extended logics as a mapping $v : S \rightarrow A$, i.e. a mapping from the set of system states S to the domain A of the c-semiring. We call such a mapping, a *valuation*. In the following let $V = A^S$ be the set of all valuations. Let us define an additive operation $\oplus : V \times V \rightarrow V$ such that $(v \oplus v')(s) = v(s) + v'(s)$ for all $s \in S$. Similarly, we define a multiplicative operation $\otimes : V \times V \rightarrow V$ as follows: $(v \otimes v')(s) = v(s) \times v'(s)$ for all $s \in S$. Let $\llbracket \mathbf{0} \rrbracket$ and $\llbracket \mathbf{1} \rrbracket$ be the valuations that respectively assign $\mathbf{0}$ and $\mathbf{1}$ to every state. It can be easily shown that $\langle V, \oplus, \otimes, \llbracket \mathbf{0} \rrbracket, \llbracket \mathbf{1} \rrbracket \rangle$ is a c-semiring. The partial order of this c-semiring is denoted by \leq_V . It is easy to see that for any two valuations $v, v' \in V$, $v \leq_V v'$ iff for all $s \in S$ we have $v(s) \leq_S v'(s)$. Recall that, that $\langle V, \leq_V \rangle$ is a complete lattice [5]. The defined c-semiring inherits various of the properties of the c-semiring C . Hence, in the document we sometimes refer to original properties of $+$ and \times instead of referring to properties of \otimes and \oplus . For example, if \times is idempotent, so is \otimes .

3.1 C-semiring μ -calculus

Let Z be a set of valuation variables, F a set of functions, which range is A and which domain is A^i for some $i > 0$, and $AV \subseteq V$ be a set of *atomic* valuations. Valuation variables are used inside fixpoint formulae as explained below.

Set F is used to represent additional functions over the domain of the concrete c-semiring being considered other than the additive and the multiplicative operations. For example, if the c-semiring under consideration is the boolean one, then one can include boolean negation as a function of F in order to be able to define all possible boolean functions³. In an optimization c-semiring, one might want to include other arithmetic functions like arithmetic multiplication or division. In sum, the choice of F determines the set of all functions that can be represented.

Set AV generalizes the set of atomic propositions of boolean logics. Indeed, an atomic boolean proposition p can be interpreted as a valuation that assigns *true* to states where p holds and *false* otherwise. Let $a \in A$, $v \in AV$, $z \in Z$

³ boolean negation, disjunction and conjunction are sufficient to define all boolean functions.

and $f \in F$. The formulae of the $c\text{-}L_\mu$ are defined as follows:

$$\begin{aligned} \phi &::= a \mid v \mid z \mid f(\phi, \dots, \phi) \mid \phi + \phi \mid \phi \times \phi \mid \kappa \mathbf{X}\phi \mid \mu z. \phi \mid \nu z. \phi \\ \kappa &::= \prod \mid \sum \mid \coprod \end{aligned}$$

Operators μ and ν are used to express least and greatest fixpoints, respectively. Temporal operator \mathbf{X} is used to refer to the evaluation of a formula in a next state. Since a state might have more than one immediate successor we use a quantifier κ preceding \mathbf{X} . While boolean μ -calculus considers only two quantifiers, namely existential and universal quantification, we consider three possible cases: \prod , \sum and \coprod . As we shall see they respectively express the application of the greatest lower bound, additive and multiplicative operations of the lattice $\langle A, \leq_S \rangle$. Note that in c -semirings, \sqcup coincides with \sum [5]. Hence, we do not include \sqcup as a quantifier, since it is redundant. On the other hand, \prod coincides with \coprod if \times is idempotent [5].

Semantics.

We interpret a formula ϕ as a valuation $\llbracket \phi \rrbracket_e$, where $e : Z \rightarrow V$ is an environment.

$$\begin{aligned} \llbracket a \rrbracket_e(s) &= a & \llbracket \phi_1 \times \phi_2 \rrbracket_e &= \llbracket \phi_1 \rrbracket_e \otimes \llbracket \phi_2 \rrbracket_e \\ \llbracket v \rrbracket_e &= v & \llbracket \kappa \mathbf{X}\phi \rrbracket_e(s) &= \kappa_{(s,s') \in T} \llbracket \phi \rrbracket_e(s') \\ \llbracket z \rrbracket_e &= e(z) & \llbracket \nu z. \phi \rrbracket_e &= \text{FIX } \lambda v. \llbracket \phi \rrbracket_{e[v/z]} \\ \llbracket \phi_1 + \phi_2 \rrbracket_e &= \llbracket \phi_1 \rrbracket_e \oplus \llbracket \phi_2 \rrbracket_e & \llbracket \mu z. \phi \rrbracket_e &= \text{fix } \lambda v. \llbracket \phi \rrbracket_{e[v/z]} \\ & & \llbracket f(\phi_1, \dots, \phi_n) \rrbracket_e(s) &= f(\llbracket \phi_1 \rrbracket_e(s), \dots, \llbracket \phi_n \rrbracket_e(s)) \end{aligned}$$

, where $s \in S$, $\kappa \in \{\prod, \sum, \coprod\}$, *FIX* abbreviates *greatest fixpoint* and *fix*, *least fixpoint*, and $e[v/z]$ is the same as e except that $e[v/z](z) = v$. A formula ϕ is closed if every variable is bound by a fixpoint operator. In such cases $\llbracket \phi \rrbracket_e$ does not depend on e and we just write $\llbracket \phi \rrbracket$. Functions $\lambda v. \llbracket \phi \rrbracket_{e[v/z]}$ are operators on V (functions $V \rightarrow V$) that we will denote with τ . In the rest of the paper we shall assume that functions of F are monotone. With this assumption, it is easy to see that every possible operator τ is monotone. Hence, by Knaster-Tarski Theorem [17] the fixpoints are well-defined. More precisely, $\mu z. \tau(z) = \prod_V \{v \mid \tau(v) \leq_V v\}$ (equivalently $\prod_V \{v \mid \tau(v) = v\}$) and $\nu z. \tau(z) = \sqcup_V \{v \mid v \leq_V \tau(v)\}$ (equivalently $\sqcup_V \{v \mid v = \tau(v)\}$)⁴.

In addition, if τ is \sqcup_V -continuous, then $\mu z. \tau(z) = \sqcup_V \{\tau^i(\mathbf{0})\}$ and if it is \prod_V -continuous, then $\nu z. \tau(z) = \prod_V \{\tau^i(\mathbf{1})\}$. In the rest of the paper we consider continuous operators. Note that, because τ is monotone, $\sqcup_V \{\tau^i(\mathbf{0})\} = \tau^\infty(\mathbf{0})$ and $\prod_V \{\tau^i(\mathbf{1})\} = \tau^\infty(\mathbf{1})$. Hence, fixpoint iteration can be applied to

⁴ In this case, \sqcup_V and \prod_V respectively refer to the *lowest upper bound* and *greatest lower bound* operations of the lattice $\langle V, \leq_V \rangle$.

evaluate a formula. Nevertheless this method is not always feasible as we shall see in the next section.

C- L_μ with negation.

The existence of a negation operator as defined in Section 2, enables some equivalences between formulae, for instance $-(\sum \mathbf{X}\phi) = \prod \mathbf{X}\neg\phi$ and $-(\mu z.\phi) = \nu z.-(\phi[-z/z])$. Note that $-(\sum \mathbf{X}\phi)$ is equivalent to $\prod \mathbf{X}\neg\phi$ if \times is idempotent.

As in boolean μ -calculus, one has to impose syntax restrictions on the use of negations in order to guarantee monotony. More precisely, we shall require that each variable appears under an even number of negations, and for each function $f \in F$ there is a (dual) function $\bar{f} \in F$ such that $\neg f(\phi_1, \dots, \phi_n) = \bar{f}(\neg\phi_1, \dots, \neg\phi_n)$. We have also to require the existence of a dual for \otimes . The idea is that with these dualities and the requirement on the use of negations one can push negations such that they appear applied to atomic propositions only.

3.2 C-semiring CTL

Syntax.

The syntax of c-CTL is defined as follows:

$$\begin{aligned} \phi &::= a \mid z \mid v \mid f(\phi, \dots, \phi) \mid \phi + \phi \mid \phi \times \phi \mid \kappa\psi \mid \kappa\mathbf{X}\phi \\ \kappa &::= \prod \mid \sum \mid \Pi \\ \psi &::= \mathbf{F}\phi \mid \mathbf{G}\phi \mid [\phi\mathbf{U}\phi] \mid [\phi\mathbf{R}\phi] \end{aligned}$$

C-CTL formulae are state formulae generated by ϕ . Symbol κ is used to introduce quantifiers over path formulae, which are generated by ψ . Path formulae combine state formulae with temporal operators \mathbf{X} (next time), \mathbf{F} (eventually), \mathbf{G} (globally), \mathbf{U} (until) and \mathbf{R} (release).

Semantics.

Again, we interpret a state formula as a mapping from the set of states S to the set of c-semiring values A . Path formulae, instead, assign c-semiring values to paths. The semantics of a , v , z , $f(\phi_1, \dots, \phi_n)$, $\phi_1 + \phi_2$ and $\phi_1 \times \phi_2$ is the same as in c- L_μ . The rest is defined as follows:

$$\begin{aligned} \llbracket \kappa\mathbf{X}\phi \rrbracket(p) &= \kappa_{(s,s') \in T} \llbracket \phi \rrbracket(s') & \llbracket \mathbf{F}\phi \rrbracket(p) &= \sum_{i \geq 0} \llbracket \phi \rrbracket(s_i^p) \\ \llbracket \kappa\psi \rrbracket(s) &= \kappa_{p \in \gamma(s)} \llbracket \psi \rrbracket(p) & \llbracket \phi_1 \mathbf{U} \phi_2 \rrbracket(p) &= \sum_{i \geq 0} (\llbracket \phi_2 \rrbracket(s_i^p) \times \prod_{0 \leq j < i} \llbracket \phi_1 \rrbracket(s_j^p)) \\ \llbracket \mathbf{G}\phi \rrbracket(p) &= \prod_{i \geq 0} \llbracket \phi \rrbracket(s_i^p) & \llbracket \phi_1 \mathbf{R} \phi_2 \rrbracket(p) &= \prod_{i \geq 0} (\llbracket \phi_2 \rrbracket(s_i^p) + \sum_{0 \leq j < i} \llbracket \phi_1 \rrbracket(s_j^p)) \end{aligned}$$

, where a p is a path. It is easy to see that $\mathbf{F}\phi \equiv [\mathbf{1}\mathbf{U}\phi]$ and $\mathbf{G}\phi \equiv [\mathbf{0}\mathbf{R}\phi]$.

The previous semantics is called path semantics. In model checking, there is an alternative equivalent semantics which interprets CTL formulae as fix-points [9]. For example, CTL formula $\mathbf{EF}\phi$ is interpreted as $\mu z.\phi \vee \mathbf{E}\mathbf{X}z$.

This equivalence shows that boolean μ -calculus contains CTL, and allows to perform CTL model checking formulae by fixpoint iteration. This suggests to define the following fixpoint semantics to c-CTL formulae:

$$\begin{aligned} \llbracket \kappa \mathbf{F} \phi \rrbracket^f &= \llbracket \mu z. \phi + \kappa \mathbf{X} z \rrbracket & \llbracket \kappa [\phi_1 \mathbf{U} \phi_2] \rrbracket^f &= \llbracket \mu z. \phi_2 + (\phi_1 \times \kappa \mathbf{X} z) \rrbracket \\ \llbracket \kappa \mathbf{G} \phi \rrbracket^f &= \llbracket \nu z. \phi \times \kappa \mathbf{X} z \rrbracket & \llbracket \kappa [\phi_1 \mathbf{R} \phi_2] \rrbracket^f &= \llbracket \nu z. \phi_2 \times (\phi_1 + \kappa \mathbf{X} z) \rrbracket \end{aligned}$$

Unfortunately, we will see that the path and fixpoints semantics of c-CTL are not equivalent in general. In order show this we need to define a bounded version of the temporal operators \mathbf{F} , \mathbf{G} , \mathbf{U} and \mathbf{R} . Intuitively, the idea is they take into account the first k states of a path only:

$$\begin{aligned} \llbracket \mathbf{G}^k \phi \rrbracket(p) &= \prod_{0 \leq i < k} \llbracket \phi \rrbracket(s_i^p) & \llbracket [\phi_1 \mathbf{U}^k \phi_2] \rrbracket(p) &= \sum_{0 \leq i < k} (\phi_2(s_i^p) \times \prod_{0 \leq j < i} \phi_1(s_j^p)) \\ \llbracket \mathbf{F}^k \phi \rrbracket(p) &= \sum_{0 \leq i < k} \llbracket \phi \rrbracket(s_i^p) & \llbracket [\phi_1 \mathbf{R}^k \phi_2] \rrbracket(p) &= \prod_{0 \leq i < k} (\phi_2(s_i^p) + \sum_{0 \leq j < i} \phi_1(s_j^p)) \end{aligned}$$

Clearly, $\mathbf{F}^\infty \equiv \mathbf{F}$, $\mathbf{G}^\infty \equiv \mathbf{G}$, $\mathbf{U}^\infty \equiv \mathbf{U}$ and $\mathbf{R}^\infty \equiv \mathbf{R}$

Theorem 3.1 *If T is total, κ is idempotent⁵ and $+$ distributes over κ then $\llbracket \kappa \mathbf{F} \phi \rrbracket = \llbracket \mu z. \phi + \kappa \mathbf{X} z \rrbracket$.*

Proof. We use the bounded version of \mathbf{F} and induction on k to prove that $\llbracket \kappa \mathbf{F}^k \rrbracket = \tau^k(\mathbf{0})$. Recall that $\tau^\infty(\mathbf{0})$ is the least fixpoint of the continuous operator $\tau = \lambda y. \llbracket \phi + \kappa \mathbf{X} y \rrbracket$.

If $k = 1$ then $\llbracket \kappa \mathbf{F}^1 \phi \rrbracket(s)$ is $\kappa_{p \in \gamma(s)} \llbracket \phi \rrbracket(s)$, while $\tau^1(\mathbf{0})$ is $\llbracket \phi \rrbracket(s) + \kappa_{(s, s') \in T} \mathbf{0} = \llbracket \phi \rrbracket(s)$ since we assume that T is total and hence $\{(s, s') \in T\}$ is not empty. If κ is idempotent, $\kappa_{p \in \gamma(s)} \llbracket \phi \rrbracket(s)$ equals $\llbracket \phi \rrbracket(s)$.

Assume for induction that $\llbracket \kappa \mathbf{F}^j \phi \rrbracket = \tau^j(\mathbf{0})$, then applying τ in both sides we get $\tau(\llbracket \kappa \mathbf{F}^j \phi \rrbracket) = \tau(\tau^j(\mathbf{0})) = \tau^{j+1}(\mathbf{0})$. But $\tau(\llbracket \kappa \mathbf{F}^j \phi \rrbracket)(s) =$

$$\begin{aligned} &= \llbracket \phi + \kappa \mathbf{X} \llbracket \kappa \mathbf{F}^j \phi \rrbracket \rrbracket(s) && \text{(By def. of } \tau) \\ &= \llbracket \phi \rrbracket(s) + \llbracket \kappa \mathbf{X} \llbracket \kappa \mathbf{F}^j \phi \rrbracket \rrbracket(s) && \text{(By def. of } \llbracket + \rrbracket) \\ &= \llbracket \phi \rrbracket(s) + \kappa_{(s, s') \in T} (\llbracket \kappa \mathbf{F}^j \phi \rrbracket(s')) && \text{(By def. of } \llbracket \kappa \mathbf{X} \rrbracket) \\ &= \llbracket \phi \rrbracket(s) + \kappa_{(s, s') \in T} (\kappa_{p \in \gamma(s')} \sum_{0 \leq i < j} \llbracket \phi \rrbracket(s_i^p)) && \text{(By def. of } \llbracket \kappa \mathbf{F} \rrbracket) \\ &= \llbracket \phi \rrbracket(s) + \kappa_{p \in \gamma(s)} \sum_{1 \leq i < j+1} \llbracket \phi \rrbracket(s_i^p) \\ &= \kappa_{p \in \gamma(s)} \sum_{0 \leq i < j+1} \llbracket \phi \rrbracket(s_i^p) && \text{(} + \text{ distr. over } \kappa; s = s_0^p) \\ &= \llbracket \kappa \mathbf{F}^{j+1} \phi \rrbracket(s) && \text{(By def. of } \llbracket \kappa \mathbf{F}^k \rrbracket, \llbracket + \rrbracket) \end{aligned}$$

Hence, $\llbracket \kappa \mathbf{F} \phi \rrbracket = \llbracket \kappa \mathbf{F}^\infty \phi \rrbracket = \tau^\infty(\mathbf{0}) = \llbracket \mu z. \phi + \kappa \mathbf{X} z \rrbracket$. \square

With similar proofs one can show the following two theorems.

Theorem 3.2 *If T is total, κ is idempotent and \times distributes over κ then $\llbracket \kappa \mathbf{G} \phi \rrbracket = \llbracket \nu z. \phi \times \kappa \mathbf{X} z \rrbracket$.*

⁵ Properties of quantifiers refer to properties of the corresponding binary functions, e.g. " \sum idempotent" means " $+$ idempotent".

Theorem 3.3 *If T is total, κ is idempotent and $+$ and \times distribute over κ then $\llbracket \kappa[\phi_1 \mathbf{U}\phi_2] \rrbracket = \llbracket \mu z. \phi_2 + (\phi_1 \times \kappa \mathbf{X}z) \rrbracket$.*

Theorem 3.4 *If T is total and \times is idempotent $\llbracket \kappa[\phi_1 \mathbf{R}\phi_2] \rrbracket = \llbracket \nu z. \phi_2 \times (\phi_1 + \kappa \mathbf{X}z) \rrbracket$.*

Note that by definition, \sum and \sqcap are idempotent and both $+$ and \times distribute over \sum , but the rest of the distributions are not true in general. However, if \times is idempotent, then \prod coincides with \sqcap and both $+$ and \times distribute over \prod . Hence, if T is total and \times is idempotent the path and fixpoint semantics are equivalent.

We now present an example that shows how both semantics differ. Assume that we are working with the optimization c-semiring $\langle \mathbb{R}^+, \min, +, +\infty, 0 \rangle$. Clearly, \min does not distribute over $+$. Suppose we have a transition system with three states s_0 , s_1 and s_2 , where neither of s_1 , s_2 has a transition and s_0 has two transitions: one to s_1 and one to s_2 . Let v be a valuation that maps every state to 1. It is easy to see that $\llbracket \prod \mathbf{F}v \rrbracket(s_0) = \min(1, 1) + \min(1, 1) = 2$, while $\llbracket \mu z. \min(v, \prod \mathbf{X}z) \rrbracket(s_0) = \min(1, 1 + 1) = 1$. The semantics of both formulae differ even if we introduce self-transitions to s_1 and s_2 to avoid end states.

C-CTL with Negation.

As in $c-L_\mu$, we can incorporate a negation operator. If we want to have the usual equivalences $-\sum \mathbf{F}\phi \equiv \prod \mathbf{G}\neg\phi$ and $-\sum[\phi_1 \mathbf{U}\phi_2] \equiv \prod[\neg\phi_1 \mathbf{R}\neg\phi_2]$, we need \times to be idempotent.

4 On computing the semantics.

Devising algorithms for our logics is not trivial. First of all, using fixpoint iteration to compute $c-L_\mu$ is not always possible; continuous operators are required. Even when fixpoint iteration is possible, it becomes infeasible for some formulae if the transition system or the c-semiring domain are infinite.

For example, consider the evaluation of $\nu z. a \times \sum \mathbf{X}z$ in a transition system with just one state s and a transition (s, s) , where the c-semiring is the probabilistic one and $0 < a < 1$. The value of the formula in s is clearly 0 but its computation requires infinitely many iterations.

While restricting to finite transition systems is reasonable, we cannot neglect infinite c-semiring domains.

On the other hand, the fact that when \times is not idempotent, the path and fixpoints semantics of c-CTL are not equivalent, avoids the usual bypass of CTL algorithms as fixpoint iterations. Hence, c-CTL requires specific algorithms.

In the following lines we show that if we restrict to finite transitions systems and c-semirings where \times is idempotent then computing any c-CTL formula

can be done by fixpoint iteration where each fixpoint requires $|S|$ iterations only.

Let us denote the concatenation of two paths p, q as pq , where we require p to be finite and $(s_{|p|-1}^p, s_0^q) \in T$, i.e. there is a transition from the last state of p to the first state of q .

A cycle is a path $p = qs$, where s is equal to s_0^p , the initial state of p . Let us denote with p^i the cycle that results from repeating i times cycle p . More precisely, $p^0 = s$ and $p^{i+1} = qp^i$.

We now define some helpful lemmas⁶.

Lemma 4.1 $\llbracket \mathbf{F}\phi \rrbracket(pq) = \llbracket \mathbf{F}\phi \rrbracket(p) + \llbracket \mathbf{F}\phi \rrbracket(q)$. If \times is idempotent $\llbracket \mathbf{G}\phi \rrbracket(pq) = \llbracket \mathbf{G}\phi \rrbracket(p) \times \llbracket \mathbf{G}\phi \rrbracket(q)$.

Lemma 4.2 Let p be a cycle qs . Then $\llbracket \mathbf{F}\phi \rrbracket(p^i) = \llbracket \mathbf{F}\phi \rrbracket(p) = \llbracket \mathbf{F}\phi \rrbracket(q)$ for any $i \geq 1$. If \times is idempotent $\llbracket \mathbf{G}\phi \rrbracket(p^i) = \llbracket \mathbf{G}\phi \rrbracket(p) = \llbracket \mathbf{G}\phi \rrbracket(q)$ for any $i \geq 1$.

Lemma 4.3 $\llbracket \sum \mathbf{F}\phi \rrbracket = \llbracket \sum \mathbf{F}^{|S|}\phi \rrbracket$. If \times is idempotent $\llbracket \prod \mathbf{G}\phi \rrbracket = \llbracket \prod \mathbf{G}^{|S|}\phi \rrbracket$.

Lemma 4.4 If \times is idempotent $\llbracket \prod \mathbf{F}\phi \rrbracket = \llbracket \prod \mathbf{F}^{|S|}\phi \rrbracket$. If \times is idempotent $\llbracket \sum \mathbf{G}\phi \rrbracket = \llbracket \sum \mathbf{G}^{|S|}\phi \rrbracket$.

Proof. We will show by induction that $\llbracket \prod \mathbf{F}^{|S|}\phi \rrbracket(s) = \llbracket \prod \mathbf{F}^i\phi \rrbracket(s)$ for any state $s \in S$ and $i \geq |S|$. This holds trivially for $i = |S|$. Assume for induction that the lemma holds for $i = n \geq |S|$.

Let $p \in \gamma(s)$ be a path starting from state s . By definition \mathbf{F}^{n+1} is $\llbracket \phi \rrbracket(s_0^p) + \dots + \llbracket \phi \rrbracket(s_n^p)$. Now observe that, because the $n + 1 > |S|$, there must be at least one state that appears more than once in the prefix s_0^p, \dots, s_n^p of p . There are two cases.

If s_n^p appears twice in the prefix, i.e. $s_j^p = s_n^p$ for at least one $0 \leq j < n$ then $\llbracket \mathbf{F}^{n+1}\phi \rrbracket(p)$ is $\llbracket \phi \rrbracket(s_0^p) + \dots + \llbracket \phi \rrbracket(s_j^p) + \dots + \llbracket \phi \rrbracket(s_{n-1}^p) + \llbracket \phi \rrbracket(s_n^p)$. Applying associativity and commutativity of $+$ we have that $\llbracket \mathbf{F}^{n+1}\phi \rrbracket(p)$ is $\llbracket \phi \rrbracket(s_0^p) + \dots + \llbracket \phi \rrbracket(s_j^p) + \llbracket \phi \rrbracket(s_n^p) + \dots + \llbracket \phi \rrbracket(s_{n-1}^p)$. Now, since $+$ is idempotent and $\llbracket \phi \rrbracket(s_j^p) = \llbracket \phi \rrbracket(s_n^p)$ we obtain $\llbracket \phi \rrbracket(s_0^p) + \dots + \llbracket \phi \rrbracket(s_j^p) + \dots + \llbracket \phi \rrbracket(s_{n-1}^p)$ which is exactly $\llbracket \mathbf{F}^n\phi \rrbracket(p)$.

The other case is when s_n^p appears only once in the prefix, i.e. only at the end. Since $n + 1 > |S|$ there must be a cycle in the prefix before s_n^p . Thus, let us represent the prefix of p by $qcq's_n^p$ where q, q' are finite paths and c is a cycle. Consider the path $p' = qc^\omega$. Clearly, $p \neq p'$ and since $p \in \gamma(s)$ then $p' \in \gamma(s)$. In the rest of the proof we call this path the absorbing path of p .

⁶ Most of the proofs are in Appendix A. Proofs for lemmas concerning the temporal operator \mathbf{G} are omitted since they are similar to the corresponding proofs concerning the temporal operator \mathbf{F} , where the main difference is that in the former ones we require \times to be idempotent.

Now, let us see the value of $\llbracket \mathbf{F}^{n+1}\phi \rrbracket(p) \times \llbracket \mathbf{F}^{n+1}\phi \rrbracket(p') =$

$$\llbracket \mathbf{F}\phi \rrbracket(qcqs_n^p) \times \llbracket \mathbf{F}\phi \rrbracket(qc) \quad (\text{Definition of } \mathbf{F} \text{ and } 4.2)$$

$$(\llbracket \mathbf{F}\phi \rrbracket(qc) + \llbracket \mathbf{F}\phi \rrbracket(q's_n^p)) \times \llbracket \mathbf{F}\phi \rrbracket(qc) \quad (\text{Lemma } 4.1)$$

$$\llbracket \mathbf{F}\phi \rrbracket(qc) \quad (\text{Absorption law})$$

$$\llbracket \mathbf{F}^n\phi \rrbracket(p') \quad (\text{Definition of } \mathbf{F} \text{ and } |qc| \leq n)$$

Absorption holds because \times is idempotent⁷. Applying a similar reasoning one can see that $\llbracket \mathbf{F}^n\phi \rrbracket(p) \times \llbracket \mathbf{F}^n\phi \rrbracket(p')$ is $\llbracket \mathbf{F}^n\phi \rrbracket(p')$ too.

We now apply induction on $\gamma(s)$ to show that $\prod_{p \in \gamma(s)} \llbracket \mathbf{F}^n\phi \rrbracket(p)$ equals $\prod_{p \in \gamma(s)} \llbracket \mathbf{F}^{n+1}\phi \rrbracket(p)$. The case $\gamma(s) = \emptyset$ is trivial. Now assume for induction that the $\gamma(s) = \gamma' \cup \{p'\}$ and the statement to show holds for γ' . Since \times is associative and commutative $\prod_{p \in \gamma(s)} \llbracket \mathbf{F}^{n+1}\phi \rrbracket(p) = (\prod_{p \in \gamma'} \llbracket \mathbf{F}^{n+1}\phi \rrbracket(p)) \times \llbracket \mathbf{F}^{n+1}\phi \rrbracket(p')$. We distinguish the above discussed two cases for p' . In the first one (the $n+1$ -st state of p' appears in a previous state of p') we know that $\llbracket \mathbf{F}^{n+1}\phi \rrbracket(p')$ is $\llbracket \mathbf{F}^n\phi \rrbracket(p')$. Applying induction and, again, associativity and commutativity of \times we obtain the desired result. In the second case (the $n+1$ -st state of p' does not appear in a previous state of p') we apply associativity and get $\prod_{p \in \gamma(s)} \llbracket \mathbf{F}^{n+1}\phi \rrbracket(p) = \prod_{p \in (\gamma' / \{p'\})} \llbracket \mathbf{F}^{n+1}\phi \rrbracket(p) \times \prod_{p'' \in \{p', p''\}} \llbracket \mathbf{F}^{n+1}\phi \rrbracket(p'')$ and $\prod_{p \in \gamma(s)} \llbracket \mathbf{F}^n\phi \rrbracket(p) = \prod_{p \in (\gamma' / \{p'\})} \llbracket \mathbf{F}^n\phi \rrbracket(p) \times \prod_{p'' \in \{p', p''\}} \llbracket \mathbf{F}^n\phi \rrbracket(p'')$, where p'' is the absorbing path of p . Using the above results ($\prod_{p'' \in \{p', p''\}} \llbracket \mathbf{F}^{n+1}\phi \rrbracket(p'') = \llbracket \mathbf{F}^{n+1}\phi \rrbracket(p'')$) and $\prod_{p'' \in \{p', p''\}} \llbracket \mathbf{F}^n\phi \rrbracket(p'') = \llbracket \mathbf{F}^n\phi \rrbracket(p'')$, associativity of \times and the induction hypothesis we obtain $\prod_{p \in \gamma(s)} \llbracket \mathbf{F}^n\phi \rrbracket(p) = \prod_{p \in \gamma(s)} \llbracket \mathbf{F}^{n+1}\phi \rrbracket(p)$.

Now applying the first induction hypothesis ($\llbracket \prod \mathbf{F}^n\phi \rrbracket = \llbracket \prod \mathbf{F}^{|\mathcal{S}|\phi} \rrbracket$) we conclude that the lemma holds. \square

Lemma 4.5 *If \times is idempotent $\llbracket \kappa[\phi_1 \mathbf{U}\phi_2] \rrbracket = \llbracket \kappa[\phi_1 \mathbf{U}^{|\mathcal{S}|\phi_2}] \rrbracket$ and $\llbracket \kappa[\phi_1 \mathbf{R}\phi_2] \rrbracket = \llbracket \kappa[\phi_1 \mathbf{R}^{|\mathcal{S}|\phi_2}] \rrbracket$.*

Theorem 4.6 *If \times is idempotent, \mathcal{S} is finite and T total, any c-CTL formula ϕ can be computed by using at most $|\mathcal{S}|$ iterations provided that any subformula of ϕ has been already computed.*

Proof. We give the proof for c-CTL formulae $\kappa \mathbf{T}\phi$. For the rest of the formulae the proof is similar. By Lemma 4.4 we know that $\llbracket \kappa \mathbf{T}^{|\mathcal{S}|\phi} \rrbracket = \llbracket \kappa \mathbf{T}^{|\mathcal{S}|\phi+1} \rrbracket$ and by Theorem 3.1 we know that $\llbracket \kappa \mathbf{F}^k\phi \rrbracket = \tau^k(\mathbf{0})$, where τ is $\lambda y. \llbracket \phi + \kappa \mathbf{X}y \rrbracket$. Hence, $\tau_1^{|\mathcal{S}|\phi}(\mathbf{0}) = \tau_1^{|\mathcal{S}|\phi+1}(\mathbf{0})$ which means that the fixpoint is achieved at least in the $|\mathcal{S}|$ -th iteration. \square

Observe that the fixpoint formula corresponding to any c-CTL formula ϕ , i.e. the one induced by the path semantics of c-CTL, is a formula in which no variable appears free under the scope of more than one fixpoint

⁷ Since \times is idempotent it coincides with \sqcap . The greatest lower and least upper bound operator of a lattice satisfy the absorption law.

operator. In other words, in every such fixpoint formula $\mu z.\phi_1 + \kappa \mathbf{X}z$, $\nu z.\phi_1 \times \kappa \mathbf{X}z$, $\mu z.\phi_2 + (\phi_1 \times \kappa \mathbf{X}z)$ or $\nu z.\phi_2 \times (\phi_1 + \kappa \mathbf{X}z)$ subformulae ϕ_1, ϕ_2 are closed. This implies that we can compute the fixpoints inside a formula starting with the innermost ones and ending with the outermost ones as sketched in the algorithm below.

Algorithm $eval(\phi)$

Input: A $c\text{-}L_\mu$ formula ϕ corresponding to a c-CTL formula.

Output: A valuation.

switch ϕ **do**

case a, v **return** ϕ ;

case $\phi_1 + \phi_2$ **return** $eval(\phi_1) \oplus eval(\phi_2)$;

case $\phi_1 \times \phi_2$ **return** $eval(\phi_1) \otimes eval(\phi_2)$;

case $f(\phi_1, \dots, \phi_n)$

$v_0, \dots, v_n := eval(\phi_1), \dots, eval(\phi_n)$;

foreach $s \in S$ **do** $v(s) := f(v_0(s), \dots, v_n(s))$;

return v ;

case $\kappa \mathbf{X}\phi_1$

$v_1 := eval(\phi_1)$;

foreach $s \in S$ **do** $v'(s) := \kappa_{(s,s') \in T} v_1(s')$;

return v ;

case $\mu z.\phi_1 + \kappa \mathbf{X}z$

$v := \mathbf{0}$; $v_1 := eval(\phi_1)$;

repeat $v' := v$; $v := eval(v_1 + \kappa \mathbf{X}v')$ **until** $v' = v$;

return v ;

case $\nu z.\phi_1 \times \kappa \mathbf{X}z$

 /* similar as above */

case $\mu z.\phi_2 + (\phi_1 \times \kappa \mathbf{X}z)$

$v := \mathbf{0}$; $v_1 := eval(\phi_1)$; $v_2 := eval(\phi_2)$;

repeat $v' := v$; $v := eval(v_1 + (v_2 \times \kappa \mathbf{X}v'))$ **until** $v' = v$;

return v ;

case $\nu z.\phi_2 \times (\phi_1 + \kappa \mathbf{X}z)$

 /* similar as above */

end

Let t_f denote the highest time complexity among all functions $f \in F$ and let t_+, t_\times respectively denote the time complexity of $+$ and \times .

The worst-case c-CTL formula is $\phi = \kappa[f(v_0, \dots, v_k, \phi_1)\mathbf{U}v_{k+1}]$, where k is $O(1)$ and ϕ_1 has the same form or is an atomic valuation. The corresponding fixpoint formula is $\mu.v_{k+1} + (f(v_0, \dots, v_k, \phi_1) \times \kappa \mathbf{X})$.

Observe that such a formula has $O(|\phi|)$ fixpoints, where $|\phi|$ denotes the length of ϕ . The time required for computing ϕ is the time required to compute ϕ_1 plus the time required to compute f , which is $|S| \cdot t_f$ plus $|S|$ times the time required to perform an iteration which is $|S| \cdot (t_+ + t_\times)$ (time to compute $+$

and \times for each state) plus $|S| + |R| \cdot t_\kappa$ (time to compute $\kappa\mathbf{X}$ for each state). The resulting time complexity is $|\phi| \cdot (|S| \cdot t_f + |S| \cdot (|S| \cdot (t_+ + t_\times) + |S| + |R| \cdot t_\kappa))$.

Assuming t_f, t_+ and t_\times to be $O(1)$ the overall complexity of the algorithm is $O(|\phi| \times |S|^3)$. Nevertheless, the complexity of computing f , $+$ and \times depends on the concrete c-semiring. In common c-semirings (optimization, probabilistic, fuzzy, boolean, etc.) $+$ and \times can be computed in constant time. However, in some cases the problem seems unfeasible. Take for example, the power set of a c-semiring which domain is infinite. The result is a c-semiring where elements of the domain are possibly infinite sets. Storing and manipulating such elements might be unfeasible.

5 Applications

Graph Problems.

Although our semantics is specially tailored for transition systems and hence, to reason about maximal paths of the underlying graph, there are still a lot of graph problems that can be expressed using our c-semiring logics. Reachability problems, for example, can be represented by using a set-based c-semiring $\langle 2^N, \cup, \cap, \emptyset, N \rangle$ and interpreting the graph (N, E) as a transition system. Formula equivalently $\sum \mathbf{F}v$ is used to represent the set of nodes that can be reached from each node, where $v \in V$ assigns to each node u the set $\{u\}$.

Path optimization problems, where the *cost* of a path is the sum of the (non-negative) costs associated to the nodes of the path can also be expressed. We show the simplest case: one optimization criteria and one goal. Consider a cost c-semiring $C = \langle \mathbb{R}^+, \min, +, +\infty, 0 \rangle$. Let $v_2 \in AV$ be a valuation, such that $v_2(s)$ is the cost associated with s if s is a target node, and ∞ otherwise. Moreover, let $v_1 \in AV$ simply assign the cost of a node to the corresponding state. The reader should check that $\sum (v_1 \mathbf{U} v_2)$ represents the minimal cost to reach a target node.

Representing multi-criteria optimization problems is also possible, but requires to use a c-semiring based on the Hoare power domain of the Cartesian product of the various optimization c-semirings as explained in [6].

Boolean Model Checking.

Our approach can be specialized to traditional model checking as follows. As c-semiring we use the boolean c-semiring $\langle \{true, false\}, \vee, \wedge, false, true \rangle$ and as transition system a c-semiring transition system $\langle S, T \rangle$, where S and T are the usual state and transition sets. For representing the set of atomic propositions we use valuations that assign *true* to states where the proposition is defined and *false* to the rest.

Our framework captures other boolean approaches based on multi-valued logics. Multi-valued CTL [8] is defined over quasi-boolean algebra, which a finite distributive lattice with a negation operator. It can be shown that given

a quasi boolean algebra $\langle A, \sqcap, \sqcup, \neg \rangle$, the algebraic structure $\langle A, \sqcup, \sqcap, \perp, \top \rangle$ with \perp and \top denoting the bottom and top elements of the lattice is a c -semiring where the multiplicative operation is idempotent.

Discounted model checking.

Roughly speaking, discounted model checking [10,11] weights events according to their distance to the present state, i.e. along a path s_0, s_1, \dots the evaluation of a formula in s_i is multiplied by c^i , where c is the a discounting factor between 0 and 1. Discounted model checking of transition systems can be performed using c -semiring μ -calculus as follows. The c -semiring is $\langle [0, 1], \max, \min, 0, 1 \rangle$, while a transition system is represented as previously explained for model checking, where for each proposition r , we have a valuation v_r . DCTL is a discounted version of CTL. In addition to \min and \max , it uses additional operators over $[0, 1]$, namely $-$, \cdot , $+$ and $+_c$, which can be represented by functions in F . The temporal (discounted) operators of DCTL are $\diamond_c, \square_c, \Delta_c$, respectively correspond to the application of \min, \max and *average* to the discounted values of the states in a path. The semantics of DCTL for transition systems can be described with c - L_μ by using a direct translation of the fixpoint semantics of DCTL. For instance, $\exists \diamond_c \phi \equiv \mu z. \phi + (\mathbf{0} +_c \sum \mathbf{X}z)$ and $\forall \diamond_c \phi \equiv \mu z. \phi + (\mathbf{0} +_c \prod \mathbf{X}z)$.

Model Checking with Resources.

Reasoning about resource constraints [7] is another potential field of application of our framework. Consider the simple case in which the system consumes a certain amount of energy in every state. We might be interested in knowing whether power consumption does not exceed a certain value. If we use the c -semiring $\langle \mathbb{R}^+, \min, \max, \infty, 0 \rangle$, and let $v(s)$ denote the power consumption at state s , then formula $\sum \mathbf{G}v$ evaluated in a state s_0 represents the maximal power consumption of optimal execution of the system starting at s_0 .

6 Conclusions and Future Work

We have presented extensions of two well known temporal specification formalism: CTL and the μ -calculus. While the original logics are defined over the boolean domain, our extensions are defined over the domain of a c -semiring, an algebraic structure that captures many problems.

We have defined syntax and semantics of our logics over transition systems, showing that the classical connection between CTL and the μ -calculus breaks when the multiplicative operation is not idempotent or there are end states in the system. The main consequences are two: the expressivenesses of our logics may be incomparable and computing each logic requires different algorithms. We have shown that model checking c -CTL when the multiplicative operation is idempotent and restricting to finite-state transitions can be

done by fixpoint iteration, where each fixpoint requires a number of iterations proportional to the number of states.

In current work we are devising solutions for cases for other fragments of our logics. Future avenues also include extending the semantics to more general formalisms, extending modal logics for WAN applications with QoS properties [12] and defining observational equivalences and preorders.

Acknowledgements.

The authors wish to thank the anonymous referees for their fruitful criticism.

References

- [1] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *5th Symp. on Logic in Computer Science (LICS'90)*, 1990.
- [2] C. Baier and E. Clarke. The algebraic mu-calculus and MTBDDs. In *Workshop on Logic, Language, Information and Computation*, 1998.
- [3] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. *CONSTRAINTS*, pages 199–240, 1999.
- [4] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based constraint logic programming: Syntax and semantics. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 23(1):1–29, January 2001.
- [5] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, March 1997.
- [6] S. Bistarelli, U. Montanari, and F. Rossi. Soft constraint logic programming and generalized shortest path problems. *Journal of Heuristics*, 8(1):25–41, January 2002.
- [7] A. Chakrabati, L. de Alfaro, T. A. Henzinger, and M. Stoenlinga. Resource interfaces. In *Third International Conference on Embedded Software (EMSOFT'02)*, Lecture Notes in Computer Science. Springer, 2003.
- [8] M. Chechik, S. Easterbrook, and A. Gurfinkel. Multi-valued symbolic model-checking. *ACM Transactions on Software Engineering and Methodology*, 2003. To appear.
- [9] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.
- [10] L. de Alfaro. Quantitative verification and control via the mu-calculus. In *Proceedings of 14th International Conference on Concurrency Theory*, Lecture Notes in Computer Science. Springer, 2003.

- [11] L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, Lecture Notes in Computer Science. Springer, 2004. To appear.
- [12] R. de Nicola and M. Loreti. A modal logic for mobile agents. *ACM Transactions on Computational Logics*, 2004. To appear.
- [13] M. Huth and M. Z. Kwiatkowska. Quantitative analysis and model checking. In *Logic in Computer Science*, pages 111–122, 1997.
- [14] A. McIver and C. Morgan. Games, probability and the quantitative μ -calculus. In *Logic Programming, Artificial Intelligence, and Reasoning (LPAR'02)*, number 2514 in Lecture Notes in Computer Science. Springer, 2002.
- [15] R. D. Nicola, G. Ferrari, U. Montanari, R. Pugliese, and E. Tuosto. A formal basis for reasoning on programmable QoS. In N. Dershowitz, editor, *International Symposium on Verification (Theory and Practice)*, Lecture Notes in Computer Science. Springer, 2003. To appear.
- [16] H. Seidl. A modal mu-calculus for durational transition systems. In *LICS'96*, 1996.
- [17] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 1955.

A Proofs

Lemma A.1 *If $-$ is a negation operator then $-\mathbf{0} = \mathbf{1}$ and $-\mathbf{1} = \mathbf{0}$.*

Proof. For all $a \in A$ we have $-a = -(\mathbf{0} \sqcup a)$ since $\mathbf{0}$ is the unit element of \sqcup . Applying the definition of $-$ we know that $-(\mathbf{0} \sqcup a) = -\mathbf{0} \sqcap -a$. It is easy to see that for all $a \in A$ we have $-a = -\mathbf{0} \sqcup -a$ exactly when $-\mathbf{0}$ is $\mathbf{1}$.

Using this result $-\mathbf{1}$ is $--\mathbf{0}$, which equals $\mathbf{0}$ by definition of $-$. □

Lemma A.2 *$-\sum_i a_i = \prod_i (-a_i)$ for all sequences a_0, a_1, \dots where $a_i \in A$ exactly when \times is idempotent.*

Proof. Direction \Leftarrow holds trivially, since if \times is idempotent, then \prod coincides with \prod and the duality holds by definition of $-$.

Observe that $-(a+b) = (-a) \times (-b)$ implies $-(a \times b) = (-a) + (-b)$. Now if the lemma holds for every sequence it also holds for sequence a, a . This entails $-(a \times a) = (-a) + (-a)$, which is $-a$, since $+$ is idempotent. Applying $-$ in both sides we get $a \times a = a$ for $a \in A$. In other words, *times* is idempotent. □

Lemma A.3 *$\langle V, \oplus, \otimes, [\mathbf{0}], [\mathbf{1}] \rangle$ is a c-semiring.*

Proof. Observe that a valuation v can be seen as a vector $(v_1, \dots, v_{|S|})$ of $|S|$ elements of A , where $v_i = v(s_i)$ and $S = \{s_0, s_1, \dots\}$. Hence, $\langle V, \oplus, \otimes, [\mathbf{0}], [\mathbf{1}] \rangle$

can be seen as $C^{|S|}$ and we know that the Cartesian product of two c-semirings is a c-semiring [5]. \square

Lemma A.4 *If \times is idempotent then \otimes is idempotent.*

Proof. We need to show that for all $v \in V$, $v \otimes v$ is v . In more detail we need $\forall v \in V, \forall s \in S : (v \otimes v)(s) = v(s)$, but this is clear since $(v \otimes v)(s)$ is $v(s) \times v(s)$ by definition and, since \times is idempotent, this equals $v(s)$. \square

Lemma A.5 *Every operator $\tau : V \rightarrow V$ is monotone if every function $f \in F$ is monotone.*

Proof. We have to show that $v \leq_V v'$ implies $\tau(v) \leq_V \tau(v')$. Recall that τ is an abbreviation of $\lambda v. \llbracket \phi \rrbracket_{e[v/z]}$. We use induction of the subformula of ϕ .

The lemma holds trivially if ϕ is a closed formula. It is also easy to see that if $\phi = z$ the lemma holds since $\tau(v)$ is simply v . If $\phi = z' \neq z$ we know that the value of z' is fixed by e and does not depend on z since we consider closed formulas only.

If $\phi = \phi_1 + \phi_2$, where at least one of ϕ_1, ϕ_2 is not a closed formula we apply induction to obtain $\llbracket \phi_1 \rrbracket_{e[v/z]} \leq_V \llbracket \phi_1 \rrbracket_{e[v'/z]}$ and $\llbracket \phi_2 \rrbracket_{e[v/z]} \leq_V \llbracket \phi_2 \rrbracket_{e[v'/z]}$. Since $+$ is monotone we get $\llbracket \phi_1 \rrbracket_{e[v/z]} + \llbracket \phi_2 \rrbracket_{e[v/z]} \leq_V \llbracket \phi_1 \rrbracket_{e[v'/z]} + \llbracket \phi_2 \rrbracket_{e[v'/z]}$. Hence, the lemma holds in this case. A similar proof can be done for $\phi = \phi_1 \times \phi_2$ or $\phi = f(\phi_1, \dots, \phi_n)$, since \times is also monotone and we assume every $f \in F$ to be monotone.

We now consider the case when $\phi = \mu z'. \phi_1$. Let w_0 be the least fixpoint of $\llbracket \phi_1 \rrbracket_{e[v/z]}$, i.e. $w_0 = \prod_V \{v_0 \mid \llbracket \phi_1 \rrbracket_{e[v/z][v_0/z']} \leq_V v_0\}$ and let w'_0 be the least fixpoint of $\llbracket \phi_1 \rrbracket_{e[v'/z]}$, i.e. $w'_0 = \prod_V \{v_0 \mid \llbracket \phi_1 \rrbracket_{e[v'/z][v_0/z']} \leq_V v_0\}$. Suppose that the lemma does not hold, i.e. $w_0 \not\leq_V w'_0$. Let w''_0 be $w_0 \sqcap_V w'_0$. Now, because $w''_0 \leq_V w_0$ and ϕ_1 is monotone we have that $\llbracket \phi_1 \rrbracket_{e[v/z][w''_0/z']} \leq_V \llbracket \phi_1 \rrbracket_{e[v/z][w_0/z']} \leq_V w_0$. Similarly, since $v \leq_V v'$ and $w''_0 \leq_V w'_0$ we obtain $\llbracket \phi_1 \rrbracket_{e[v/z][w''_0/z']} \leq_V \llbracket \phi_1 \rrbracket_{e[v'/z][w''_0/z']} \leq_V \llbracket \phi_1 \rrbracket_{e[v'/z][w'_0/z']} \leq_V w'_0$. The obtained inequalities imply $\llbracket \phi_1 \rrbracket_{e[v/z][w''_0/z']} \leq_V w_0 \sqcap_V w'_0 = w''_0$. Since we assumed that $w_0 \not\leq_V w'_0$ we know that $w''_0 \leq w_0$ and $w_0 \neq w''_0$. Thus w_0 cannot be the least fixpoint of $\llbracket \phi_1 \rrbracket_{e[v/z]}$. This contradicts our assumption.

The case $\phi = \mu z'. \phi_1$ is similar. \square

Lemma A.6 *If $-$ is a negation operator $-(\sum \mathbf{X}\phi) \equiv \prod \mathbf{X}-\phi$ for any ϕ .*

Proof. By definition $\llbracket -(\sum \mathbf{X}\phi) \rrbracket(s)$ is $-\sum_{(s,s') \in T} \llbracket \phi \rrbracket(s')$. Applying the definition of $-$ we obtain $\prod_{(s,s') \in T} \llbracket -\phi \rrbracket(s')$. \square

Lemma A.7 *If $-$ is a negation operator then $-(\mu z. \phi) \equiv \nu z. -(\phi[-z/z])$ for any ϕ .*

Proof. By definition of μ , $-(\mu z. \tau(z))$ is $-\prod_V \{v \mid v = \tau(v)\}$. Applying the definition of $-$ we obtain $\bigsqcup_V \{-v \mid v = \tau(v)\}$, which is equal to $\bigsqcup_V \{-v \mid -v = -\tau(v)\}$. Now, if we let $v' = -v$ we have $\bigsqcup_V \{v' \mid v' = -\tau(-v')\}$ which by definition of ν is $\nu z. -\tau(-z)$. \square

Lemma A.8 $\mathbf{F}\phi \equiv [\mathbf{1U}\phi]$ and $\mathbf{G}\phi \equiv [\mathbf{0R}\phi]$.

Proof. This is trivial because $\llbracket [\mathbf{1U}\phi] \rrbracket(p)$ is $\sum_{i \geq 0} (\llbracket \phi \rrbracket(s_i^p) \times \prod_{0 \leq j < i} \llbracket \mathbf{1} \rrbracket(s_j^p)) = \sum_{i \geq 0} \llbracket \phi \rrbracket(s_i^p)$, which equals $\llbracket \mathbf{F}\phi \rrbracket(p)$. The proof for the other equivalence is similar. \square

Lemma A.9 *If \times is idempotent then $-(\sum \mathbf{F}\phi) \equiv \prod \mathbf{G}-\phi$, $-(\sum \mathbf{G}\phi) \equiv \prod \mathbf{F}-\phi$, $-(\sum [\phi_1 \mathbf{U}\phi_2]) \equiv \prod [-\phi_1 \mathbf{R}-\phi_2]$ and $-(\prod [\phi_1 \mathbf{U}\phi_2]) \equiv \sum [-\phi_1 \mathbf{R}-\phi_2]$.*

Proof. Recall that, by Lemma A.2, if \times is idempotent then \times is the dual of $+$. We show the proof for dualities $-(\sum \mathbf{F}\phi) \equiv \prod \mathbf{G}-\phi$ and $-(\sum [\phi_1 \mathbf{U}\phi_2]) \equiv \prod [-\phi_1 \mathbf{R}-\phi_2]$. The rest of the proofs are similar.

By definition of \mathbf{F} , \mathbf{G} and $-$: $\llbracket [-(\sum \mathbf{F}\phi)] \rrbracket = -\sum_{p \in \gamma(s)} \sum_{i \geq 0} \llbracket \phi \rrbracket(s_i^p) = \prod_{p \in \gamma(s)} -\sum_{i \geq 0} \llbracket \phi \rrbracket(s_i^p) = \prod_{p \in \gamma(s)} \prod_{i \geq 0} -\llbracket \phi \rrbracket(s_i^p) = \prod_{p \in \gamma(s)} \prod_{i \geq 0} \llbracket -\phi \rrbracket(s_i^p) = \prod \mathbf{G}-\phi$.
 Similarly, $\llbracket [-(\sum [\phi_1 \mathbf{U}\phi_2])] \rrbracket = -\sum_{p \in \gamma(s)} \sum_{i \geq 0} (\llbracket \phi_2 \rrbracket \times \prod_{0 \leq i < j} \llbracket \phi_1 \rrbracket) = \prod_{p \in \gamma(s)} -\sum_{i \geq 0} (\llbracket \phi_2 \rrbracket \times \prod_{0 \leq i < j} \llbracket \phi_1 \rrbracket) = \prod_{p \in \gamma(s)} \prod_{i \geq 0} (\llbracket -\phi_2 \rrbracket + -\prod_{0 \leq i < j} \llbracket \phi_1 \rrbracket) = \prod_{p \in \gamma(s)} \prod_{i \geq 0} (\llbracket -\phi_2 \rrbracket + \sum_{0 \leq i < j} \llbracket -\phi_1 \rrbracket) = \prod \mathbf{G}-\phi$. \square

Lemma A.10 *$+$ distributes over \times iff \times is idempotent.*

Proof. Direction \Leftarrow is shown in [5]. We show the opposite direction. Assume that $+$ distributes over \times . Then, for all $a \in A$ we have $a+(a \times a) = (a+a) \times (a+a)$, which is $a \times a$ since $+$ is idempotent. By definition of \leq_S , this means that $a \leq_S a \times a$. On the other hand, $a \times a \leq_I a$ because \times is intensive. It follows that a equals $a \times a$ for all $a \in A$. Hence, \times is idempotent. \square

Proof. (of Theorem 3.3)

Let τ be $\lambda y. \llbracket \phi_2 + (\phi_1 \times \kappa \mathbf{X}z) \rrbracket$. For $k = 1$ it is easy to see that $\llbracket \kappa[\phi_1 \mathbf{U}^k \phi_2] \rrbracket = \tau^k(\mathbf{0})$. More precisely, $\llbracket \kappa[\phi_1 \mathbf{U}^1 \phi_2] \rrbracket(s)$ is $\kappa_{p \in \gamma(s)} \llbracket \phi_2 \rrbracket(s)$, while $\tau(\mathbf{0})$ is $\llbracket \phi_2 \rrbracket(s)$. Since κ is idempotent, the theorem holds for $k = 1$.

Assume for induction that $\llbracket \kappa[\phi_1 \mathbf{U}^n \phi_2] \rrbracket = \tau^n(\mathbf{0})$, then applying τ to both

$$\begin{aligned}
 & \text{sides of the equality we get } \tau(\llbracket \kappa[\phi_1 \mathbf{U}^n \phi_2] \rrbracket) = \tau^{n+1}(\mathbf{0}). \text{ But } \tau(\llbracket \kappa[\phi_1 \mathbf{U}^n \phi_2] \rrbracket) = \\
 & = \llbracket \phi_2 + (\phi_1 \times \kappa \mathbf{X} \llbracket \kappa[\phi_1 \mathbf{U}^n \phi_2] \rrbracket) \rrbracket(s) \\
 & \quad (\text{By def. of } \tau) \\
 & = \llbracket \phi_2 \rrbracket(s) + (\llbracket \phi_1 \rrbracket(s) \times \kappa_{p \in \gamma(s)} \sum_{1 \leq i < n+1} (\llbracket \phi_2(s_i^p) \rrbracket \times \prod_{1 \leq j < i} \llbracket \phi_1(s_j^p) \rrbracket)) \\
 & \quad (\text{Various def.}) \\
 & = \llbracket \phi_2 \rrbracket(s) + \kappa_{p \in \gamma(s)} (\llbracket \phi_1 \rrbracket(s) \times \sum_{1 \leq i < n+1} (\llbracket \phi_2(s_i^p) \rrbracket \times \prod_{1 \leq j < i} \llbracket \phi_1(s_j^p) \rrbracket)) \\
 & \quad (\times \text{ distr. over } \kappa) \\
 & = \llbracket \phi_2 \rrbracket(s) + \kappa_{p \in \gamma(s)} (\sum_{1 \leq i < n+1} (\llbracket \phi_1 \rrbracket(s) \times \llbracket \phi_2(s_i^p) \rrbracket \times \prod_{1 \leq j < i} \llbracket \phi_1(s_j^p) \rrbracket)) \\
 & \quad (\times \text{ distr. over } +) \\
 & = \llbracket \phi_2 \rrbracket(s) + \kappa_{p \in \gamma(s)} (\sum_{1 \leq i < n+1} (\llbracket \phi_2(s_i^p) \rrbracket \times \prod_{0 \leq j < i} \llbracket \phi_1(s_j^p) \rrbracket)) \\
 & \quad (\times \text{ distr. over } \prod) \\
 & = \kappa_{p \in \gamma(s)} (\llbracket \phi_2 \rrbracket(s) + \sum_{1 \leq i < n+1} (\llbracket \phi_2(s_i^p) \rrbracket \times \prod_{0 \leq j < i} \llbracket \phi_1(s_j^p) \rrbracket)) \\
 & \quad (+ \text{ distr. over } \kappa) \\
 & = \kappa_{p \in \gamma(s)} (\llbracket \phi_2 \rrbracket(s) + \sum_{0 \leq i < n+1} (\llbracket \phi_2(s_i^p) \rrbracket \times \prod_{0 \leq j < i} \llbracket \phi_1(s_j^p) \rrbracket)) \\
 & \quad (\prod_{\leq j < 0} \llbracket \phi_1(s_j^p) \rrbracket = \mathbf{1}) \\
 & = \llbracket \kappa[\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket \\
 & \quad (\text{Various def.})
 \end{aligned}$$

□

Proof. (of Theorem 3.4)

The proof is similar to the previous one. Note that in the previous one we use the fact that \times distributes over $+$. Since we have the symmetric case we need $+$ to distribute over \times , but this is equivalent to requiring \times to be idempotent. □

Proof. (of Lemma 4.1)

$\llbracket \mathbf{F}\phi \rrbracket(pq)$ is $\llbracket \phi \rrbracket(s_0^p) + \dots + \llbracket \phi \rrbracket(s_{|p|-1}^p) + \llbracket \phi \rrbracket(s_0^q) + \dots$. Applying associativity of $+$ and, the definition of $\llbracket \mathbf{F}\phi \rrbracket$ we obtain $\llbracket \mathbf{F}\phi \rrbracket(p) + \llbracket \mathbf{F}\phi \rrbracket(q)$. □

Proof. (of Lemma 4.2)

By Lemma 4.1 and associativity of $+$ we have that $\llbracket \mathbf{F}\phi \rrbracket(p^i)$ is equal to $(\sum_{1 \leq j \leq i} \llbracket \mathbf{F}\phi \rrbracket(q)) + \llbracket \mathbf{F}\phi \rrbracket(s)$. Since $+$ is idempotent we obtain $\llbracket \mathbf{F}\phi \rrbracket(q) + \llbracket \mathbf{F}\phi \rrbracket(s)$. Applying Lemma 4.1 again this equals $\llbracket \mathbf{F}\phi \rrbracket(p)$.

Since s coincides with the first state of p we have $\llbracket \phi \rrbracket(s_0^p) = \llbracket \phi \rrbracket(s)$ and $+$ is associative, commutative and idempotent one can easily see that $\llbracket \mathbf{F}\phi \rrbracket(p)$ equals $\llbracket \mathbf{F}\phi \rrbracket(q)$. □

Proof. (of Lemma 4.3)

Let us denote with $reach(s, k)$ the set of states that are reachable from s through at most k transitions. Since $+$ is idempotent, associative and commutative then $[[\mathbf{F}^k \phi]](p) = \sum_{s' \in \{s_0^p, \dots, s_{k-1}^p\}} [[\phi]](s')$, i.e. the addition of $[[\phi]](s')$ for all distinct states s' in the prefix of length k of path p . Hence, it is easy to see that $[[\sum \mathbf{F}^k \phi]](s) = \sum_{p \in \gamma(s)} \sum_{s' \in \{s_0^p, \dots, s_{k-1}^p\}} [[\phi]](s')$ is $\sum_{s' \in reach(s, k)} [[\phi]](s')$, i.e. the addition of $[[\phi]](s')$ for all distinct states s' that are reachable from s in at most k transition. Since for any $k > |S|$ we know that every state reachable from s is reachable in at most $|S|$ transitions we have $reach(s, k) = reach(s, |S|)$. Hence $[[\sum \mathbf{F} \phi]] = [[\sum \mathbf{F}^{|S|} \phi]]$. \square

What follows is the proof of Lemma 4.5. We abbreviate $(\prod_{0 \leq i < j} [[\phi_1]](s_i^p)) \times [[\phi_2]](s_j^p)$ with $t_p(s_0^p \dots s_j^p)$ in the following. We call it a term. Observe that $[[\phi_1 \mathbf{U}^k \phi_2]]$ is $\prod_{0 \leq j < k} t_p(s_0^p \dots s_j^p)$. We first define some helpful lemmas.

Lemma A.11 *Let p be an infinite path such that $s_i^p = s_j^p$ for some $0 \leq j < i$. Then $[[\phi_1 \mathbf{U}^{i+1} \phi_2]](p) = [[\phi_1 \mathbf{U}^i \phi_2]](p)$.*

Proof. Observe that $[[\phi_1 \mathbf{U}^{i+1} \phi_2]](p) = [[\phi_1 \mathbf{U}^i \phi_2]](p) + t_p(s_0^p \dots s_i^p)$. Because s_i^p is exactly one s_j^p with $j < i$ we have a term $t_p(s_0^p \dots s_j^p)$ in $[[\phi_1 \mathbf{U}^i \phi_2]](p)$. By absorption $t_p(s_0^p \dots s_j^p) + t_p(s_0^p \dots s_i^p)$ is $t_p(s_0^p \dots s_j^p)$. Hence, $[[\phi_1 \mathbf{U}^{i+1} \phi_2]](p) = [[\phi_1 \mathbf{U}^i \phi_2]](p)$. \square

Lemma A.12 *Let $p = p'c^\omega$ be a path such that $c = qs$ is a cycle. Then $[[\phi_1 \mathbf{U}^k \phi_2]](p) = [[\phi_1 \mathbf{U}^{|p'q|} \phi_2]](p)$ for every $k \geq |p'q|$.*

Proof. See that Lemma A.11 holds for every $k \geq |pq|$. \square

Lemma A.13 *Let p be an infinite path such that $s_i^p = s_j^p$ for some $0 \leq i < j$. Let p' be a path $s_0^p \dots s_{i-1}^p c^\omega$, where $c = s_i^p \dots s_j^p$. For every prefix p'_1 of p' with $|p'_1| < j$, there is a prefix p_1 of p such that $t_p(p'_1) = t_p(p_1)$.*

Proof. For every prefix of p' of length less than j there is an identical prefix of p since, up to the j -th state both paths are equal. \square

Proof. (of Lemma 4.5)

We give the proof for temporal operator \mathbf{U} and path quantifier $\kappa = \prod$. The rest of the proofs a similar.

We proof by induction that $[[\prod [[\phi_1 \mathbf{U}^k \phi_2]]]] = [[\prod [[\phi_1 \mathbf{U}^{|S|} \phi_2]]]]$ for $k \geq |S|$. This holds trivially for $k = |S|$. Assume for induction that it holds for $k = n \geq |S|$. We shall see that it holds for $k = n + 1$.

Consider a path $p \in \gamma(s)$ and its $n + 1$ -st state. There are two cases: if it coincides with a state preceding it in p we know that $[[\phi_1 \mathbf{U}^{n+1} \phi_2]](p) = [[\phi_1 \mathbf{U}^n \phi_2]](p)$ by Lemma A.11.

The second case is when the $n + 1$ -st state does not coincide with any proceeding state. Since $n + 1 > |S|$, path p contains at least one cycle and, thus, we know that $p = s_0^p \dots s_i^p \dots s_j^p \dots s_n^p \dots$, where $s_i^p = s_j^p$ for at least

two i, j such that $0 \leq i < j < n$. Let p' be the path $s_0^p \dots s_{i-1}^p c^\omega$, where we $c = s_i^p \dots s_j^p$. Clearly, $p' \neq p$ and $p' \in \gamma(s)$.

$$\begin{aligned}
 & \llbracket [\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket(p) = \\
 & \quad (\text{by Lemma A.12.}) \\
 & \llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket(p) = \\
 & \quad (\times \text{ idempotent implies } + \text{ distributes over } \times.) \\
 & (\llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^n \phi_2] \rrbracket(p)) + (\llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times t_p(s_0^p \dots s_n^p)) = \\
 & \quad (\delta = \{t_p(s_0^p \dots s_l^p) \mid 0 \leq l < n\}, \delta' = \{t_{p'}(s_0^{p'} \dots s_l^{p'}) \mid 0 \leq l < j\}) \\
 & (\sum_{t \in \delta'} t \times \sum_{t \in \delta} t) + (\sum_{t' \in \delta'} t' \times t_p(s_0^p \dots s_n^p)) = \\
 & \quad (\text{By commutativity and associativity of } +. \text{ Lemma A.13 where } t_0 = t'_0) \\
 & (\sum_{t \in (\delta' / \{t'_0\}} t \times \sum_{t \in (\delta / \{t_0\})} t) + (\sum_{t \in (\delta' / \{t'_0\}} t \times t_p(s_0^p \dots s_n^p)) + \\
 & \quad ((t'_0 \times t_0) + (t'_0 \times t_p(s_0^p \dots s_n^p))) = \\
 & \quad (\times \text{ is idempotent (thus } t'_0 \times t_0 = t'_0) \text{ and absorption law.)} \\
 & (\sum_{t \in (\delta' / \{t'_0\}} t_{p'} \times \sum_{t \in (\delta / \{t_0\})} t) + (\sum_{t' \in (\delta' / \{t'_0\}} t' \times t_p(s_0^p \dots s_n^p)) + (t'_0 \times t_0) = \\
 & \quad (\text{By commutativity and associativity of } +.) \\
 & (\sum_{t \in \delta'} t \times \sum_{t \in \delta} t) + (\sum_{t' \in (\delta' / \{t'_0\})} t' \times t_p(s_0^p \dots s_n^p))
 \end{aligned}$$

Since this holds for every $t'_0 \in \delta'$ and δ' is finite, we can repeat this reasoning and conclude that $(\llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^n \phi_2] \rrbracket(p)) + (\llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times t_p(s_0^p \dots s_n^p))$ is just $\llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^n \phi_2] \rrbracket(p)$.

Hence, $\llbracket [\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket(p) = \llbracket [\phi_1 \mathbf{U}^n \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^n \phi_2] \rrbracket(p)$.

Applying this results in an inductive reasoning on $\gamma(s)$ as in proof of lemma 4.4 one can show that $\llbracket \prod[\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket$ is $\llbracket \prod[\phi_1 \mathbf{U}^n \phi_2] \rrbracket$.

Finally applying the first induction hypothesis, namely $\llbracket \prod[\phi_1 \mathbf{U}^n \phi_2] \rrbracket = \llbracket \prod[\phi_1 \mathbf{U}^{|\mathcal{S}|} \phi_2] \rrbracket$ we arrive to the desired result. \square