Quantitative verification of weighted Kripke structures

Patricia Bouyer, Patrick Gardy, Nicolas Markey

LSV – CNRS & ENS Cachan – France

Abstract. Extending formal verification techniques to handle quantitative aspects, both for the models and for the properties to be checked, has become a central research topic over the last twenty years. Following several recent works, we study model checking for (one-dimensional) weighted Kripke structures with positive and negative weights, and temporal logics constraining the total and/or average weight. We prove decidability when only accumulated weight is constrained, while allowing average-weight constraints alone already is undecidable.

1 Introduction

Quantitative verification. Model checking [CGP00] has been developed for almost 40 years as a formal method for verifying correctness flushing out bugs of computerized systems: this technique first consists in representing the system under study as a mathematical model (a finite-state transition system, in the most basic setting), expressing the correctness property in some logical formalism, and running an algorithm that exhaustively explores the set of behaviours of the model for proving or disproving the property. Model checking has been successfully applied on various real-life case studies.

Though model checking has primarily concentrated on pure qualitative analysis via the development of various temporal logics [Pnu77,CE82,QS82], rich models and logics have also been developed in order to take into account quantitative aspects of reactive systems and of their correctness properties.

In particular, there has been quite a lot of efforts invested in the study of *weighted* discrete transition systems, like weighted Kripke structures [CC95] or counter automata (or VASS) [EN94]. In these models, the weight gives some quantitative information on the system, which might be timing information and constraints, or energy consumption, or value of a discrete variable, etc. This weight can either be some information that we observe on the system (like in weighted Kripke structures) or its value can constrain the further behaviour of the system (like in counter automata).

On such models, there are many interesting verification questions that can be asked. First one can be interested in qualitative structural and logical properties of the system, that can for instance be expressed using some logical formalism.

Partly supported by ERC Starting Grant EQualIS and EU FP7 project Cassting.

Then one can be more interested in quantitative properties of the system, like (among others) mean-payoff constraints [ZP96] (the limit-average of the weight along an execution satisfies some constraint), or energy constraints [BFL$^+$08] (all along the execution, the weight satisfies some constraint). More interestingly, one might be interested in properties that might mix qualitative logical properties and quantitative constraints. For instance, in a robot-planning system, one would like to verify that an autonomous robot can always go back to its home base without running out-of-energy.

In this setting, weights in weighted transition systems have been most often restricted to range over the *nonnegative integers* (mostly for representing timing information), and temporal logics have been augmented either with constrained modalities [Koy90,EMSS92] or with explicit variables [AH94], and efficient algorithms have been developed and implemented [BLN03,JLSØ13]. Models with both positive and negative weights have also been studied, but mostly qualitative behaviours alone have been analyzed (this is the case in counter automata—see [HKOW09,GHOW10,Haa12] for recent references), or quantitative constraints have been analyzed (in weighted Kripke structures or games [CDHR10,CRR12]). Mixing qualitative logical properties and quantitative constraints has only poorly been addressed so far, and many works only consider specifications given as a conjunction of a qualitative logical property and a quantitative constraint: this is for instance the case of optimal reachability, mean-payoff parity games [BMOU11], energy parity games [CD12], mean-payoff LTL synthesis [BBFR13].

In that direction, the most relevant and advanced propositions are those of [DG09] and of [BCHK11]. In [DG09], LTL is (roughly) extended with Presburger constraints over weights and interpreted over one-counter automata (and an extension thereof). The satisfiability and model checking problems are addressed, and it is shown that only a restriction to a single weight leads to decidability. In [BCHK11], CTL and LTL are extended with (prefix) accumulative values over finitely many variables: these logics embed numerical assertions such as $\mathsf{Sum}(x) \sim c$ (*e.g.* to compare the accumulated amount of some resource x against some value c) and $\mathsf{Avg}(x) \sim c$ (*e.g.* to constrain the average consumption of some resource x). In this context, the authors of [BCHK11] show undecidability of the logics in general, and propose several fragments for which model checking is decidable: (*i*) LTL with limit-average, where (roughly) a property can be rewritten as a conjunction of a qualitative logical property and a quantitative constraint; and (*ii*) CTL restricted to **EF** and **EX** modalities, with all kinds of numerical assertions, but where the qualitative logical part of the formula is rather poor. Notice that this decidability result is rather surprising, since reachability in two-counter machines is undecidable. The difference is that here counters can go negative, and taking the Parikh image of a path is enough for checking properties expressed in the **EF**-fragment. Designing logical languages that can express intricate properties mixing qualitative logical features and quantitative constraints, and for which model checking remains decidable, seems therefore to be a real challenge!

Our contribution. We investigate further the temporal logics with prefix accumulation that has been proposed in [BCHK11], and we study the impact of restricting the logic to a single weight. The logics we consider are therefore based on CTL and LTL, and they extend the standard logics with two kinds of numerical assertions: $\mathsf{Sum} \sim c$ expresses that the accumulated weight satisfies the constraint $\sim c$, and $\mathsf{Avg} \sim c$ expresses that the current average of the weight satisfies the constraint $\sim c$. The extension of CTL is called WCTL and the extension of LTL is called WLTL. Those two logics will be interpreted on weighted Kripke structures, and we will be interested in the model checking problem.

We prove in this paper that when using only Sum constraints, model checking for both CTL and LTL extensions is decidable. On the contrary, allowing Avg constraints leads to undecidability for both branching and linear time. This undecidability result for our logic $\mathsf{WLTL}_{\mathsf{Avg}}$ is to be compared with the decidability of LTL with limit-average modalities of [BCHK11]: limit-average constraints can be made disjoint from the logical property expressed by the formula of LTL, whereas average modalities are really mixed with the logical property. Finally, we define a *flat* fragment of WLTL allowing both Sum and Avg constraints, but restricting the way they can be nested in the formula; we prove that this fragment has decidable model checking.

2 Definitions

2.1 Weighted Kripke structures

Definition 1. *Let* AP *be a finite set of atomic propositions. A* weighted Kripke structure *over* AP *is a tuple* $\mathcal{K} = \langle S, R, \ell\rangle$ *where* S *is a finite set of states,* $R \subseteq S \times W \times S$ *(where* $W \subseteq \mathbb{Z}$ *is the set of weights*[1] *of* $\mathcal{K}$*, which we assume are given in binary notation) is a weighted transition relation (which we assume total, meaning that for all* $s \in S$*, there exists* $w \in \mathbb{Z}$ *and* $s' \in S$ *s.t.* $(s, w, s') \in R$*), and* $\ell\colon S \to 2^{\mathsf{AP}}$ *is a function labelling the states with atomic propositions.*

A weighted Kripke structure with zero-tests *is a weighted Kripke structure with extra zero-test transitions, that is,* $R \subseteq S \times (W \cup \{=0\}) \times S$*.*

A weighted Kripke structure (with zero-tests) is unitary *if its set of weights* W *is included in* $\{-1, 0, +1\}$*.*

Let $\mathcal{K} = \langle S, R, \ell\rangle$ be a weighted Kripke structure, $s_0 \in S$ be a state of $\mathcal{K}$, and $w_0 \in \mathbb{Z}$. A *run in* $\mathcal{K}$ *from* (s_0, w_0) is a (finite or infinite) sequence $\pi = (q_i, w_i)_{i\in I}$ such that $q_0 = s_0$, I is an interval of $\mathbb{N}$ containing 0, and for all $i \in I \setminus \{0\}$, $(q_{i-1}, w_i - w_{i-1}, q_i) \in R$. When I is finite, we write $|\pi|$ for the length of π (the cardinal of I), and we write $\mathsf{last}(\pi)$ (resp. $\mathsf{last}_q(\pi)$, $\mathsf{last}_w(\pi)$) for the configuration $(q_{\max(I)}, w_{\max(I)})$ (resp. the state $q_{\max(I)}$, the weight $w_{\max(I)}$). Given a path $\pi = (q_i, w_i)_{i\in I}$ and $k \in I$, the prefix of π up to k is the path $\pi_{\le k} = (q_i, w_i)_{i\in[0,k]}$; the suffix of π from k is the path $\pi_{\ge k} = (q_{k+i}, w_{k+i})_{i\in\mathbb{N}\cap(I-k)}$. When $\mathcal{K}$ has

[1] Rational weights would easily be handled, after scaling the values by the least common multiple of their denominators.

zero-tests, a run in $\mathcal{K}$ is a sequence $\pi = (q_i, w_i)_{i \in I}$ where for all $i \in I \setminus \{0\}$, either $(q_{i-1}, w_i - w_{i-1}, q_i) \in R$, or $w_i = w_{i-1} = 0$ and $(q, =0, q') \in R$.

Weighted Kripke structures (with or without zero-tests) are related to counter automata [Min61]. A *one-counter automaton*[2] is a weighted Kripke structure with zero-tests in which runs are restricted to only visit configurations with nonnegative weight: the state space is $S \times \mathbb{N}$, and it is not possible to take a transition that would make the counter (or weight) negative. It is also usual to have branching zero-tests in one-counter automata, instead of our simple "guarded transitions" (zero-tests transitions). However, branching tests in one-counter automata can easily be implemented in weighted Kripke structures with zero-tests (for instance, to test whether the value of the counter is positive, we put in sequence a decrementation of -1 followed by an incrementation of $+1$).

The execution tree of a weighted Kripke structure $\mathcal{K}$ from some configuration (s_0, w_0) is the $(S \times \mathbb{Z})$-tree

$$\mathcal{T} = \{\pi \in (S \times \mathbb{Z})^* \mid \pi \text{ is a run from } (s_0, w_0) \text{ in } \mathcal{K}\}$$

For convenience, we label each node π of $\mathcal{T}$ with $\mathsf{last}(\pi)$, which then relates each node of $\mathcal{T}$ with its corresponding state of $\mathcal{K}$. A *branch* of $\mathcal{T}$ is an infinite run π from (s_0, w_0) in $\mathcal{K}$: every prefix $\pi_{\leq i}$ ($i \in \mathbb{N}$) is then an element of $\mathcal{T}$. We write $B(\mathcal{T})$ for the set of branches of $\mathcal{T}$.

2.2 Weighted temporal logics

We extend both branching-time temporal logic CTL [CE82,QS82] and linear-time temporal logic LTL [Pnu77] with numerical constraints on weights:

Definition 2. *A* numerical assertion *is built on the following grammar:*

$$\alpha ::= \mathit{Sum} \sim c \mid \mathit{Avg} \sim c$$

where $\sim$ ranges over $\{<, \leq, =, \geq, >\}$ and c ranges over $\mathbb{Q}$.

Fix a set AP *of atomic propositions. The syntax of* WCTL *over* AP *is given as*

$$\phi ::= p \mid \alpha \mid \neg\,\phi \mid \phi \wedge \phi \mid \mathbf{EX}\,\phi \mid \mathbf{AX}\,\phi \mid \mathbf{E}\phi\,\mathbf{U}\,\phi \mid \mathbf{A}\phi\,\mathbf{U}\,\phi \mid w \cdot \phi$$

where p ranges over AP *and α ranges over numerical assertions.*

The syntax of WLTL *over* AP *is given as*

$$\phi ::= p \mid \alpha \mid \neg\,\phi \mid \phi \wedge \phi \mid \mathbf{X}\,\phi \mid \phi\,\mathbf{U}\,\phi \mid w \cdot \phi$$

where p ranges over AP *and α ranges over numerical assertions.*

The operation $w \cdot \phi$ in both logics is called the reset *operation.*

[2] Counter automata are often required to have set of weights included in $\{-1, 0, +1\}$, which we call *unitary* counter automata in the sequel. The counter automata we consider here correspond to *succinct* counter automata of [HKOW09,GHOW10].

In the sequel, we write $\mathsf{WCTL}_{\mathsf{Sum}}$ (resp. $\mathsf{WCTL}_{\mathsf{Avg}}$) for the fragments of WCTL using only Sum (resp. Avg) in numerical assertions. We also write $\mathsf{WCTL}^{\mathsf{rf}}$, $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ and $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Avg}}$ for the respective fragments with no reset operations. Also, we write $\mathsf{WLTL}_{\mathsf{Sum}}$ (resp. $\mathsf{WLTL}_{\mathsf{Avg}}$) for the fragments of WLTL using only Sum (resp. Avg) in numerical assertions. We also write $\mathsf{WLTL}^{\mathsf{rf}}$, $\mathsf{WLTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ and $\mathsf{WLTL}^{\mathsf{rf}}_{\mathsf{Avg}}$ for the respective fragments with no reset operations.

The semantics of numerical assertions is defined on finite runs π of a weighted Kripke structure $\mathcal{K}$ as follows (boolean combinations omitted):

$$\begin{aligned} \pi \models \mathsf{Sum} \sim c \quad &\text{iff} \quad \mathsf{last}_w(\pi) \sim c \\ \pi \models \mathsf{Avg} \sim c \quad &\text{iff} \quad \mathsf{last}_w(\pi) \sim c \cdot |\pi| \end{aligned}$$

Such constraints can for instance be used to express the so-called *energy constraints* [BFL$^+$08,CDHR10], requiring that $\mathsf{Sum} \geq 0$ all along a run. We can also reinforce this condition by additionally requiring that, at the end of the run, the average energy level (over the prefix) has remained within a given range.

Semantics of WCTL. The semantics of WCTL is defined inductively, on the execution tree of a weighted Kripke structure. Let $\mathcal{K}$ be a weighted Kripke structure and (s_0, w_0) be an initial configuration. Let $\mathcal{T}$ be the execution tree of $\mathcal{K}$ from (s_0, w_0); fix a branch π of $\mathcal{T}$, a position $i \in \mathbb{N}$ along π (with the intended meaning that i corresponds to the node $\pi_{\leq i}$ of $\mathcal{T}$; in particular, the node at position 0 corresponds to (s_0, w_0)). The semantics of WCTL is defined as follows (atomic propositions and boolean operators omitted):

$$\begin{aligned}
\mathcal{T}, \pi, i \models \alpha \quad &\text{iff} \quad \pi_{\leq i} \models \alpha \\
\mathcal{T}, \pi, i \models \mathbf{EX}\, \phi \quad &\text{iff} \quad \exists \pi' \in B(\mathcal{T}).\ (\pi_{\leq i} = \pi'_{\leq i} \text{ and } \mathcal{T}, \pi', i+1 \models \phi) \\
\mathcal{T}, \pi, i \models \mathbf{AX}\, \phi \quad &\text{iff} \quad \forall \pi' \in B(\mathcal{T}).\ (\pi_{\leq i} = \pi'_{\leq i} \Rightarrow \mathcal{T}, \pi', i+1 \models \phi) \\
\mathcal{T}, \pi, i \models \mathbf{E}\phi_1 \,\mathbf{U}\, \phi_2 \quad &\text{iff} \quad \exists \pi' \in B(\mathcal{T}). \exists j \geq i.\ (\pi_{\leq i} = \pi'_{\leq i} \text{ and} \\
&\qquad \mathcal{T}, \pi', j \models \phi_2 \text{ and } \forall i \leq k < j.\ \mathcal{T}, \pi', k \models \phi_1) \\
\mathcal{T}, \pi, i \models \mathbf{A}\phi_1 \,\mathbf{U}\, \phi_2 \quad &\text{iff} \quad \forall \pi' \in B(\mathcal{T}). \exists j \geq i.\ (\pi_{\leq i} = \pi'_{\leq i} \Rightarrow \\
&\qquad \mathcal{T}, \pi', j \models \phi_2 \text{ and } \forall i \leq k < j.\ \mathcal{T}, \pi', k \models \phi_1) \\
\mathcal{T}, \pi, i \models w \cdot \phi \quad &\text{iff} \quad \mathcal{T}', \pi', 0 \models \phi \text{ where } \mathcal{T}' \text{ is the execution tree of } \mathcal{K} \\
&\qquad \text{from } (\mathsf{last}_q(\pi_{\leq i}), 0) \text{ and } \pi' \in B(\mathcal{T}')
\end{aligned}$$

Notice that the value of $\mathcal{T}', \pi', 0 \models \phi$ in the semantics of $w \cdot \phi$ does not depend on the choice of the branch π', so the semantics is well-defined. We can generalize that remark:

Lemma 3. *Pick $\pi, \pi' \in B(\mathcal{T})$.*

– *If $\pi_{\leq i} = \pi'_{\leq i}$ for some position $i \in \mathbb{N}$, then for every formula $\phi \in \mathsf{WCTL}$,*

$$\mathcal{T}, \pi, i \models \phi \quad \textit{iff} \quad \mathcal{T}, \pi', i \models \phi.$$

– *If* $\mathsf{last}(\pi_{\leq i}) = \mathsf{last}(\pi'_{\leq j})$ *for some* $i, j \in \mathbb{N}$*, then for all* $\phi \in \mathsf{WCTL}_{\mathsf{Sum}}$*,*

$$\mathcal{T}, \pi, i \models \phi \quad \text{iff} \quad \mathcal{T}, \pi', j \models \phi.$$

The value of $\mathcal{T}, \pi, 0 \models \phi$ does not depend on the choice of the branch $\pi \in B(\mathcal{T})$; we then define the truth value of $\mathcal{K}, (s_0, w_0) \models \phi$ as that of $\mathcal{T}, \pi, 0 \models \phi$, where $\mathcal{T}$ is the execution tree of $\mathcal{K}$ from (s_0, w_0) and π is any branch of $\mathcal{T}$.

Definition 4 (WCTL model-checking problem). *Let* ϕ *be a* WCTL *formula, let* $\mathcal{K}$ *be a weighted Kripke structure and* s_0 *be an initial state. The model-checking problem with fixed initial credit asks, for a given* $w_0 \in \mathbb{Z}$*, whether* $\mathcal{K}, (s_0, w_0) \models \phi$*. The model-checking problem with unknown initial credit asks whether there exists* $w_0 \in \mathbb{Z}$ *such that* $\mathcal{K}, (s_0, w_0) \models \phi$*.*

Semantics of WLTL. The semantics of WLTL is defined inductively over infinite runs of a weighted Kripke structure. Let $\mathcal{K}$ be a weighted Kripke structure, π be an infinite run in $\mathcal{K}$ from some configuration (s_0, w_0), and $i \in \mathbb{N}$ be a position along π. The semantics of WLTL is defined as follows (simple cases omitted):

$\pi, i \models \alpha$	iff	$\pi_{\leq i} \models \alpha$
$\pi, i \models \mathbf{X}\,\phi$	iff	$\pi, i+1 \models \phi$
$\pi, i \models \phi_1 \,\mathbf{U}\, \phi_2$	iff	$\exists j \geq i.\ (\pi, j \models \phi_2$ and $\forall i \leq k < j.\ \pi, k \models \phi_1)$
$\pi, i \models w \cdot \phi$	iff	$\pi', 0 \models \phi$ where π' is the run of $\mathcal{K}$ from $(\mathsf{last}_q(\pi_{\leq i}), 0)$ that follows the same transitions as $\pi_{\geq i}$.

As for WCTL, the reset operator imposes that further numerical assertions will count from the current position only, this is why the position is reset to 0.

We write $\mathcal{K}, (s_0, w_0) \models \phi$ whenever there exists an infinite run π from (s_0, w_0) in $\mathcal{K}$ such that $\pi, 0 \models \phi$. Note that the choice of an existential semantics is arbitrary and harmless, given that the logic is closed under negation.

Definition 5 (WLTL model-checking problem). *Let* ϕ *be a* WLTL *formula, let* $\mathcal{K}$ *be a weighted Kripke structure and* s_0 *be an initial state. The model-checking problem with fixed initial credit asks, for a given* $w_0 \in \mathbb{Z}$*, whether* $\mathcal{K}, (s_0, w_0) \models \phi$*. The model-checking problem with unknown initial credit asks whether there exists* $w_0 \in \mathbb{Z}$ *such that* $\mathcal{K}, (s_0, w_0) \models \phi$*.*

Remark 6. Note that the reset operator which is used in both logics is a powerful operator, which can be used to express multiple numerical constraints on various portions of a run. This is a rather standard operator in temporal logics, which is for instance in the core of linear-time timed temporal logic TPTL [AH94]. Note nevertheless that the logics WCTL and WLTL above only allow for one weight variable in a given formula. We will see that the reset operator does not impact much on the model checking of WCTL, but has a strong impact on WLTL model checking.

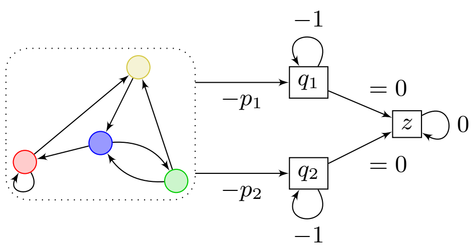

Fig. 1. From $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ to CTL (there is one transition from each state in the dotted box (which is the original weighted Kripke structure) to each state q_i).

3 Algorithm for model checking $\mathsf{WCTL}_{\mathsf{Sum}}$

In this section we prove the decidability of the model-checking of $\mathsf{WCTL}_{\mathsf{Sum}}$ over weighted Kripke structures by reducing it to the model-checking of CTL over one-counter automata. We proceed by first removing numerical assertions from the formulas (which requires to modify also the Kripke structure), and then by building a one-counter automaton and a CTL formula. We then apply the results of [Haa12,Ser06], and carefully analyze the complexity of the algorithm.

We first focus on logic $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Sum}}$, and will explain at the end of the section how we can extend the result to $\mathsf{WCTL}_{\mathsf{Sum}}$.

3.1 Moving quantitative constraints into the model

We prove that model checking $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ is logspace-reducible to model checking CTL on structures allowing zero-tests. This is achieved by adding "tests modules" in the model, and replacing Sum constraints with a CTL condition in the corresponding test modules.

Let $\mathcal{K} = \langle S, R, \ell\rangle$ be a weighted Kripke structure, and ϕ be a $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ formula involving integer constants $\mathcal{P} = \{p_1, ..., p_k\}$. We define a new weighted Kripke structure *with zero tests* $\mathcal{K}'_{\mathcal{P}} = \langle S', R', \ell'\rangle$ as follows:

– $S' = S \cup \{q_i \mid 1 \le i \le k\} \cup \{z\}$,
– $R' = R \cup \{(s, -p_i, q_i), (q_i, -1, q_i), (q_i, = 0, z), (z, 0, z) \mid s \in S,\ 1 \le i \le k\}$,
– $\ell' : \begin{cases} s \in S \mapsto \ell(s) \\ s = q_i \mapsto a_i \\ s = z \mapsto \mathsf{zero} \end{cases}$

The a_i are fresh atomic propositions. The construction is depicted on Fig. 1 for two constants p_1 and p_2. The intuition is as follows: whenever the formula requires comparing the current weight with p_i in a state s of $\mathcal{K}$, the new formula will query the existence of a transition to q_i, and check that the value of the weight when reaching q_i is nonnegative (by testing whether state z is reachable).

Now, after an easy transformation of ϕ into $\widehat{\phi}$ in order not to evaluate subformulas in newly-added states, we get:

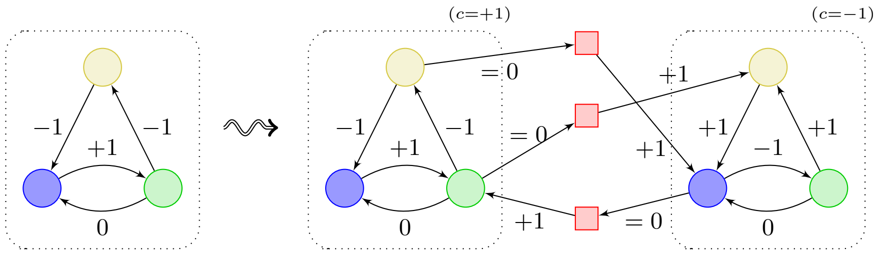

Fig. 2. From weighted Kripke structures to one-counter automata.

Proposition 7. *Let $\mathcal{K} = \langle S, R, \ell \rangle$ be a weighted Kripke structure. Let ϕ be a* $\mathsf{WCTL}^{\mathsf{rf}}_{\mathit{Sum}}$ *formula with integer constants in $\mathcal{P}$, and $s_0 \in S$ and $v \in \mathbb{Z}$. Let $\mathcal{K}'_{\mathcal{P}}$ be the weighted Kripke structure with zero test as defined above, and $\widehat{\phi}$ be the formula obtained from ϕ by the transformation above. Then*

$$\mathcal{K}, (s_0, v) \models \phi \quad \text{iff} \quad \mathcal{K}'_{\mathcal{P}}, (s_0, v) \models \widehat{\phi}.$$

Note that the size of $\mathcal{K}'_{\mathcal{P}}$ is polynomial in $\mathcal{K}$ and ϕ, and so is $\widehat{\phi}$.

3.2 From weighted Kripke structures to one-counter automata

We now reduce the model checking problem for CTL on weighted Kripke structures with zero tests to the same problem on unitary one-counter automata, in order to invoke the algorithms for CTL model-checking of [Haa12,Ser06]. The one-counter automaton will be made of two copies of the weighted Kripke structure: one to be used when the accumulated weight is nonnegative, and one when it is nonpositive. We will have zero-tests between both copies. Prior to this transformation, we first make the Kripke structure unitary, so that no transition will jump from positive to negative weights, or vice-versa, without hitting zero.

We now come to the transformation of this unitary weighted Kripke structure into a unitary one-counter automaton $\mathcal{C}$. The natural idea is to consider two copies of each state: one is used when the accumulated weight is nonnegative, and one when the total weight is nonpositive. By considering the opposite value of the weight in the second copy, we end up with a unitary one-counter automaton. Fig. 2 illustrates this construction on a simple example.

Again rewriting the formula as $\widehat{\phi}$ to "hide" newly added states, we get:

Proposition 8. *Let $\mathcal{K} = \langle S, R, \ell \rangle$ be a weighted Kripke structure with zero-tests, ϕ be a* CTL *formula, $s_0 \in S$ and $w_0 \in \mathbb{Z}$. Let $\mathcal{C}$ be the one-counter automaton obtained above, and $\widehat{\phi}$ be the formula obtained from ϕ. Then*

$$\mathcal{K}, (s_0, w_0) \models \phi \quad \text{iff} \quad \mathcal{C}, ((s_0, \mathit{sign}(w_0)), |w_0|) \models \widehat{\phi}$$

where $\mathit{sign}(0)$ can be taken as either $+1$ or -1.

3.3 $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ model checking over weighted Kripke structures

We now come to the main result of this section:

Theorem 9. *Model checking* $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ *over weighted Kripke structures with fixed initial credit is* EXPSPACE*-complete. It is* PSPACE*-complete when starting from a unitary weighted Kripke structure.*

Proof. The algorithms are obtained by applying the previous transformations from $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ model checking over weighted Kripke structures to CTL model checking over unitary one-counter automata, and then relying on the PSPACE algorithm of [Haa12,Ser06] for model checking unitary one-counter automata.

Hardness is easily proved by reducing the model-checking problem of CTL over one-counter automata to that for $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ over weighted Kripke structures. The former problem was proved EXPSPACE-complete in [GHOW10], and PSPACE-complete over unitary one-counter automata in [GL10]. The reduction is rather straightforward, by reinforcing the CTL formula in order to enforce nonnegative value of the accumulated weight all along the paths. □

We can now extend the above algorithm to handle the reset operator: when a formula ϕ contains $w \cdot \psi$ as a subformula, we first evaluate ψ in all the states of the Kripke structure, assuming initial weight zero, and apply a classical labelling algorithm. In the end:

Theorem 10. *Model checking* $\mathsf{WCTL}_{\mathsf{Sum}}$ *over weighted Kripke structures with fixed initial credit is* EXPSPACE*-complete; it is* PSPACE*-complete when starting from a unitary weighted Kripke structure.*

3.4 Model checking $\mathsf{WCTL}_{\mathsf{Sum}}$ with unknown initial credit

The model checking of $\mathsf{WCTL}_{\mathsf{Sum}}$ with unknown initial credit can be reduced to the fixed initial credit case by adding an initial module which allows to set the weight to any value. We can state the following result:

Theorem 11. *Model checking* $\mathsf{WCTL}_{\mathsf{Sum}}$ *over weighted Kripke structures with unknown initial credit is* EXPSPACE*-complete; it is* PSPACE*-complete when starting from a unitary weighted Kripke structure.*

This result has to be compared with the lower-bound problems in weighted timed automata which is PSPACE-complete with unknown initial credit, but becomes undecidable with fixed initial credit [BLM14].

4 Model checking $\mathsf{WCTL}_{\mathsf{Avg}}$ is undecidable

In this section, we prove that constraining the average of the weight value leads to undecidability:

Theorem 12. *Model checking* $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Avg}}$ *(and therefore* $\mathsf{WCTL}_{\mathsf{Avg}}$*) over weighted Kripke structures is undecidable.*

Proof. We encode the halting problem for (deterministic) two-counter machines into our model-checking problem with fixed initial value. The values c_1 and c_2 of the counters at a given position along an execution is encoded by the length of the path being of the form $2^{c_1} \cdot 3^{c_2} \cdot 5^a$, where a is a nonnegative integer. Decrementing counter c_1 will then amount to multiplying the length of the path by 5/2, which will be achieved by taking a self-loop on a state until the total average value of the weight reaches a given value.

We illustrate the construction on an example. Figure 3 depicts a *module* from (q, A_k) to $(q', A_{k'})$. This module will be used to modify (increment or decrement) the counters. The $\mathsf{WCTL}^{\mathsf{rf}}_{\mathsf{Avg}}$ formula will enforce that the average weight value of a path ending in a state labelled A_k be exactly k, for all $0 \leq k \leq 2$. Consider such a path, of length n, ending in state (q, A_k) of Figure 3 (so that the accumulated weight is $k \cdot n$). With j taking values 2, 3, 5/2 or 5/3 will implement all four instructions modifying the counters. Now, we extend this path following the depicted module, until reaching $(q', A_{k'})$. Write $x - 2$ for the number of times we take the self-loop on state (r, k, k'). Then the length of the path when reaching $(q', A_{k'})$ is $n + x$, and its total accumulated weight is $k \cdot n + \frac{(jk'-k)\cdot x}{j-1}$. The requirement that the average be k' in $(q', A_{k'})$ entails

$$k \cdot n + \frac{(jk' - k) \cdot x}{j - 1} = k' \cdot (n + x).$$

One easily checks that, provided $k \neq k'$, this implies $x = (j - 1) \cdot n$, so that the length of the whole path when reaching $(q', A_{k'})$ is $j \cdot n$ (assuming $j > 1$), and the average is indeed k'.

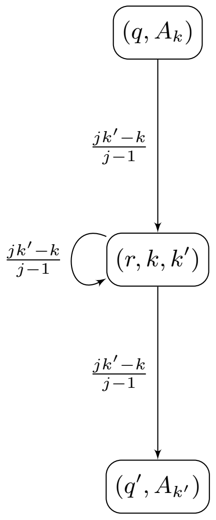

Fig. 3. Updating counters

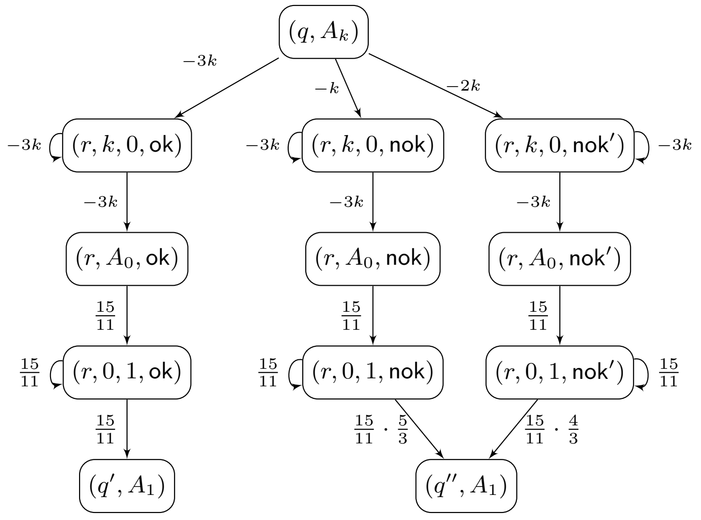

Fig. 4. Testing counters (here c_2)

Similarly, Figure 4 is a module testing whether counter c_2 equals zero. Starting in state (q, A_k) with a path of length n, and total accumulated weight $k \cdot n$, it is rather clear that we can reach state (r, A_0, ok) if, and only if, n is an integer multiple of 3 (which means that $c_2 > 0$). State (r, A_0, ok) is then reached by a path of total length $4n/3$, and accumulated weight zero. Then the path goes to $(r, 0, 1, \mathsf{ok})$, and takes the loop $x - 2$ times, and reaches (q', A_1). The accumulated weight then is $15x/11$, and the length of the path is $4n/3 + x$. The average is 1 exactly when $x = 11n/3$, for which the total length of the path is $5n$; this way, the counters are back to their original value when reaching (q', A_1).

If we follow the middle branch of the module, we reach (r, A_0, nok) with average zero if, and only if, $n - 1$ is an integer multiple of 3, which implies that n is not, so that $c_2 = 0$. The length of the path when reaching (r, A_0, nok) is $n+1+(n-1)/3$. We then reach $(q", A_1)$ after looping $x-2$ times on $(r, 0, 1, \mathsf{nok})$; then the length of the path is $(4n+2)/3+x$, and the total weight is $(x+2/3)\cdot 15/11$. Since we require the average weight to equal 1, we get $x = (11n - 2)/3$, which yields a total final length of $5n$, as expected. A similar computation can be conducted for the rightmost branch. $\square$

Remark 13. Notice that this can be made to work with only nonnegative and/or integer weights, by shifting and/or multiplying all weights and average constraints by some constant.

5 Algorithms for model checking fragments of WLTL

5.1 Decidability of $\mathsf{WLTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ model checking

It follows from the proof of Theorem 12 that model checking $\mathsf{WLTL}^{\mathsf{rf}}_{\mathsf{Avg}}$ is undecidable: the formula we built contains a single "until" modality, and can thus be interpreted as a $\mathsf{WLTL}^{\mathsf{rf}}_{\mathsf{Avg}}$ formula. We believe this result is quite surprising since it was proven in [BCHK11] that the model checking of LTL extending with limit-average constraints (that is, constraints speaking on the long-run value of the average) is decidable, even for several weights.

We thus focus on WLTL with only Sum constraints, and begin with proving the decidability of the case without reset operator:

Theorem 14. *Model checking* $\mathsf{WLTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ *over weighted Kripke structures both with fixed and unknown initial credit is* PSPACE*-complete.*

Our proof closely follows the ideas of [GHOW10] and [DG09]: we apply the same transformation as for WCTL, reducing our weighted Kripke structure into a unitary one-counter automaton. We then plug at each state a module (as displayed on Fig. 5, where M is the maximal absolute value of the constants involved in the formula) to encode numerical assertions into plain LTL. The LTL formula is then checked using Büchi automata.

Notice that in [DG09], the model-checking problem is proven decidable for an extension of LTL with Presburger-defined constraints (but without atomic propositions) over one-dimensional weighted Kripke structures with zero tests.

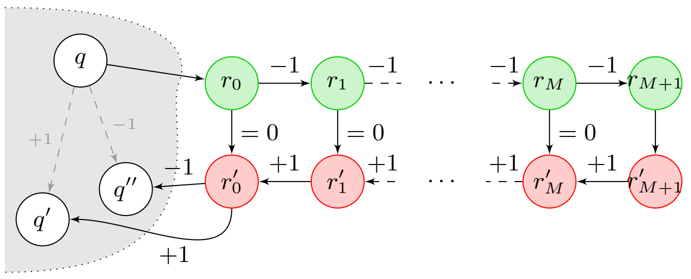

Fig. 5. From $\mathsf{WLTL}^{\mathsf{rf}}_{\mathsf{Sum}}$ to LTL (notice that q' and q'' also have their corresponding test modules, which we omitted to draw for the sake of readability).

How to handle atomic propositions for that problem is not addressed in [DG09]. The extension to unknown initial credit is similar to the case of $\mathsf{WCTL}_{\mathsf{Sum}}$.

5.2 $\mathsf{WLTL}_{\mathsf{Sum}}$ model checking is undecidable

In this section, we prove that contrary to the case of $\mathsf{WCTL}_{\mathsf{Sum}}$, the reset operator makes model checking undecidable. This is not so surprising, and is actually a corollary of a similar result for LTL with one register over one-counter automata [DLS10, Thm. 17].

LTL with registers extends LTL with a way of *storing* the current value of a counter (or other data, depending on the underlying model), and compare the stored value later on during the execution. For instance, $\downarrow \phi$ stores the current value of the counter before evaluating ϕ. Then $\uparrow$ evaluates to true at positions where the value of the counter equals the value stored in the register. For instance, $\downarrow \neg\, \mathbf{X}\, \mathbf{F} \uparrow$, means that it must never be the case that the value of the counter equals its initial value.

The translation of LTL with *one* register into $\mathsf{WLTL}_{\mathsf{Sum}}$ is then straightforward: $\downarrow$ corresponds to setting the weight to zero, and $\uparrow$ simply means $\mathsf{Sum} = 0$. In order to encode the behaviours of a one-counter automaton as a weighted Kripke structure, we have to additionally require that the accumulated weight remains nonnegative, by adding $\mathbf{G}\,(\mathsf{Sum} \geq 0)$ as a global conjunct. The following theorem directly follows:

Theorem 15. *Model checking* $\mathsf{WLTL}_{\mathsf{Sum}}$ *over weighted Kripke structures is undecidable.*

5.3 Model checking a *flat* fragment of WLTL

We conclude this part with a fragment of WLTL which forbids numerical assertions on the left-hand-side of an "until" formula. As we prove below, model checking our fragment is decidable, so that it offers an alternative to the fragments $\mathbf{EF}^{\Sigma}$ and $\mathsf{LTL}^{\mathrm{lim}}$ of [BCHK11], with a lower complexity. Also, our fragment allows us to use multiple variables, as well as average assertions. The syntactic restriction is the price to pay for this, but we believe that the fragment remains interesting in practice when dealing with average values, as it is rarely the case that some average value has to be constrained all along an execution. We call this fragment *flat*. Notice that this adjective was already used in similar contexts, but for different restrictions [CC00].

In this section, we consider multi-dimensional weighted Kripke structures, as our algorithm will be able to handle them; they extend the 1-dimensional Kripke structures in the obvious way. FlatWLTL is defined by the following syntax:

$$\mathsf{FlatWLTL} \ni \phi ::= p \mid \neg p \mid \alpha \mid \phi \vee \phi \mid \phi \wedge \phi \mid \mathbf{X}\,\phi \mid \mathbf{G}\,\psi \mid \psi\,\mathbf{U}\,\phi \mid w \cdot \phi$$

where α ranges over numerical assertions, ψ ranges over LTL and, and w ranges over the set of variables. The semantics follows that of WLTL.

Notice that our fragment allows both Sum and Avg constraints, as well as the reset operator. Also notice that our logic includes a restricted version of the release modality, namely $\phi\,\mathbf{R}\,\psi$, which can be expressed as $\mathbf{G}\,\psi \vee \psi\,\mathbf{U}\,(\phi \wedge \psi)$.

Additionally, using the techniques in [BCHK11], this allows us to transform Avg-assertions into Sum-assertions (if we have to check $\mathsf{Avg}(x) \leq c$, we introduce a new variable x_c whose updates are shifted by $-c$ compared to the updates of x, and then check $\mathsf{Sum}(x_c) \leq 0$). Hence we can assume that our formula does not contain Avg assertions.

We first prove that a path satisfying a formula in FlatWLTL can be decomposed into finitely many segments, delimited with positions where some numerical assertions have to be checked.

Pick $\phi \in \mathsf{FlatWLTL}$, and a path π in a weighted Kripke structure $\mathcal{K}$ such that $\pi, 0 \models \phi$. We inductively build a finite set of positions along π at which we may have to evaluate numerical assertions; at all other positions, only pure-LTL formulas will need to be evaluated. For this, we consider the tree of ϕ, and we proceed inductively. We first decorate the root of the tree of ϕ with 0 (to indicate that ϕ holds true at position 0). If ϕ is pure LTL, then we end the labelling. Then, from a node representing subformula ψ that has been decorated with integer i, we distinguish between the different types of nodes:

- if the node corresponds to an atomic proposition, the negation thereof, or a numerical assertion, we are done;
- if the node corresponds a reset operator (that is, $\psi = w \cdot \psi'$), we label its successor node, which corresponds to subformula ψ', with i;
- if the node is a conjunction of subformulas (that is, $\psi = \psi_1 \wedge \psi_2$), we mark all successors of this node with i. Notice that indeed all subformulas have to hold true at position i of π;
- if the node is a disjunction of subformulas, then one of the disjunct has to hold true at position i of π. We label this successor with i;
- if the node is a **X**-modality, we label its successor node with $i + 1$;
- if the node if a **G**-modality, we decorate its successor node with the interval $[i, +\infty)$ (hence the inductive labelling ends here for this branch, since the formula is flat);

– if the node is an **U**-modality, then there must be a position $j \geq i$ along π at which the right-hand-side subformula of this **U**-formula holds true. We decorate the right-hand-side successor node of the present node with j, and the left-hand-side node with $[i, j-1]$. The inductive labelling ends here for the left-hand-side branch since the formula is flat.

The following trivially holds: for every node that has been labelled by an integer i, if that node corresponds to subformula ψ, then $\pi, i \models \psi$. Conversely, if for some run π we can label consistently the tree representation of ϕ with integers (or intervals on pure LTL formulas) such that the root is labelled with 0, then $\pi, 0 \models \phi$.

This way, we have identified sufficiently many witnessing positions where some numerical assertions may have to be checked. Let $\mathcal{P} = \{i_0 = 0, i_1, ..., i_k\}$ be the set of integers (named *breakpoints* hereafter) labelling the tree of ϕ, assuming $i_l < i_m$ whenever $l < m$. By construction, we have that $k \leq |\phi|$. Between any two such consecutive positions, the decorated tree gives us (conjunctions of) LTL subformulas to be checked at every intermediary position (the above labelling tells us that all positions between two checkpoints have to satisfy the same LTL subformulas of ϕ). The tree also indicates those breakpoints where we reset some of the weight variables. Note that given two labellings of the formula tree yielding the same order on breakpoints and making the same choices in the disjunctions, the very same formulas have to be verified between two breakpoints.

As a first step of our (non-deterministic) algorithm, we pick a number $k+1$ of (at most $|\phi|$) breakpoints, and guess a labelling of the formula tree with the indices of the breakpoints (or intervals) that respects the rules defined earlier. Then we uniquely associate with each $h \in [0, k]$ an LTL "right-hand-side" subformula ξ_h and a numerical assertion α_h to be checked at breakpoint i_h, and an LTL formula ζ_h that has to be checked at intermediary positions before the next breakpoint (with ζ_k being enforced at all positions after the last breakpoint). As noted above, those formulas are uniquely fixed by the order of breakpoints and the labelling of the formula tree; moreover ξ_h and ζ_h are conjunctions of pure LTL subformulas of ϕ whereas α_h are conjunctions of numerical assertions appearing as subformulas of ϕ or negations of such subformulas. See Fig. 6 for an example of a formula tree labelled with breakpoints.

With each formula ξ_h selected above, we associate a Büchi automaton $\mathcal{A}_{\xi'_h}$ where ξ'_h is the formula $\mathbf{G}\,(b_h \Rightarrow \xi_h)$ and b_h is a fresh atomic proposition that only holds true at breakpoint i_h (note that the value of i_h is not known). Similarly, with formulas ζ_h, we associate an automaton $\mathcal{A}_{\zeta'_h}$ enforcing formula $\zeta'_h = \mathbf{G}\,(c_h \Rightarrow \zeta_h)$, where atomic proposition c_h only holds true between i_h and i_{h+1}.

Our algorithm will check the existence of segments between two breakpoints that satisfy the required properties. Each segment corresponds to a finite path in the product $\mathcal{L}$ of the weighted Kripke structure $\mathcal{K}$ and all the Büchi automata built above. When working with the j-th segment, proposition b_j is set to true at the first step, and c_j holds true all along this segment. This way, the automata $\mathcal{A}_{\xi'_j}$ and $\mathcal{A}_{\zeta'_j}$ play their roles of checking ξ_j at the beginning of the segment, and ζ_j at every position in the segment. The automata that have already been "activated"

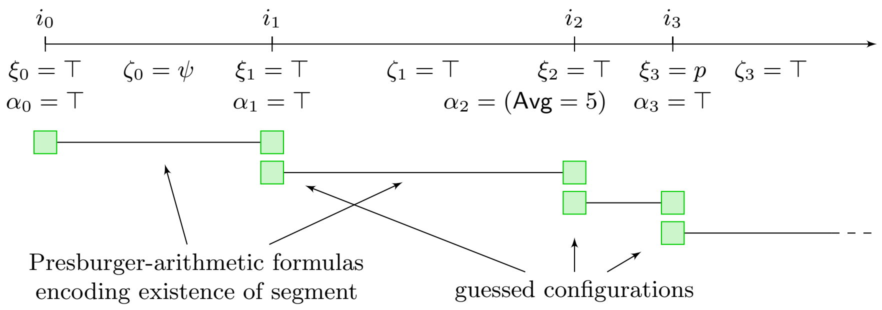

Fig. 7. Schematics representation of our algorithm

(at previous breakpoints) keep on running, finishing their computations, while the automata corresponding to later breakpoints remain "idle".

One configuration of $\mathcal{L}$ (i.e. a state of $\mathcal{K}$ and a state per Büchi automaton constructed above) can be stored using polynomial space. However, we do not have a bound on the length of the segments, which prevents us from guessing the path on-the-fly. Instead, we guess the configurations of $\mathcal{L}$ at each breakpoint (there are at most $|\phi|$ breakpoints). It remains to decide the existence of a path in $\mathcal{L}$ from the configuration in one breakpoint to the configuration in the next one, and checking that the numerical assertions at each breakpoint are satisfied. Following the ideas of [BCHK11], we can encode the existence of each segment by assigning one variable with each transition of $\mathcal{L}$: each variable represents the number of times this transition will be taken along the segment, and one can easily write a Presburger-arithmetic formula expressing that a valuation for those variables corresponds to a path and that the numerical assertions are fulfilled. Notice that we can easily handle reset operators in those equations. In the end, our formula is in the existential fragment of Presburger arithmetic, and has size exponential, so that our procedure runs in NEXPTIME.

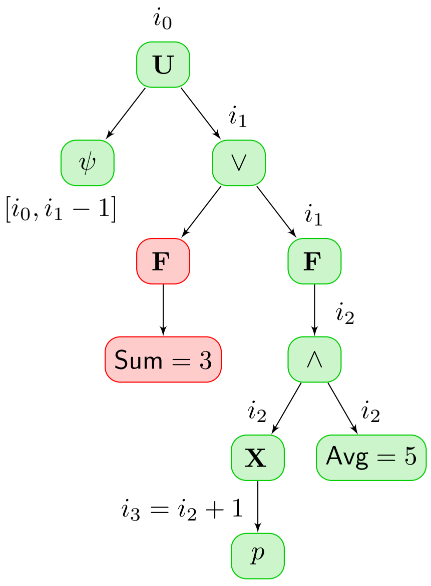

Fig. 6. Labelling of the tree of formula $\psi\,\mathbf{U}\,\big[\mathbf{F}\,(\mathsf{Sum} = 3) \vee \mathbf{F}\,(\mathbf{X}\,(p) \wedge \mathsf{Avg} = 5)\big]$ (assuming $\psi \in \mathsf{LTL}$) with breakpoints.

Theorem 16. *Model checking* FlatWLTL *over weighted Kripke structures is decidable in* NEXPTIME.

References

[AH94] R. Alur and T. A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–203, 1994.

[BBFR13] A. Bohy, V. Bruyère, E. Filiot, and J.-F. Raskin. Synthesis from ltl specifications with mean-payoff objectives. In TACAS'13, p. 169–184, 2013.

[BCHK11] U. Boker, K. Chatterjee, T. A. Henzinger, and O. Kupferman. Temporal specifications with accumulative values. In LICS'11, p. 43–52. IEEE Comp. Soc. Press, 2011.

[BFL^{+}08] P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey, and J. Srba. Infinite runs in weighted timed automata with energy constraints. In FORMATS'08, LNCS 5215, p. 33–47. Springer, 2008.

[BLM14] P. Bouyer, K. G. Larsen, and N. Markey. Lower-bound constrained runs in weighted timed automata. *Performance Evaluation*, 73:91–109, 2014.

[BLN03] D. Beyer, C. Lewerentz, and A. Noack. Rabbit: A tool for BDD-based verification of real-time systems. In CAV'03, LNCS 2725, p. 122–125. Springer, 2003.

[BMOU11] P. Bouyer, N. Markey, J. Olschewski, and M. Ummels. Measuring permissiveness in parity games: Mean-payoff parity games revisited. In ATVA'11, LNCS 6996, p. 135–149, Taipei, Taiwan, 2011. Springer.

[CC95] S. V. A. Campos and E. M. Clarke. Real-time symbolic model checking for discrete time models. In *Real-time symbolic model checking for discrete time models*, AMAST Series in Computing 2, p. 129–145. World Scientific, 1995.

[CC00] H. Comon and V. Cortier. Flatness is not weakness. In CSL'00, LNCS 1862, p. 262–276. Springer, 2000.

[CD12] K. Chatterjee and L. Doyen. Energy parity games. *Theoretical Computer Science*, 458:49–60, 2012.

[CDHR10] K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Generalized mean-payoff and energy games. In FSTTCS'10, Leibniz International Proceedings in Informatics 8. Leibniz-Zentrum für Informatik, 2010.

[CE82] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In LOP'81, LNCS 131, p. 52–71. Springer, 1982.

[CGP00] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking.* MIT Press, 2000.

[CRR12] K. Chatterjee, M. Randour, and J.-F. Raskin. Strategy synthesis for multi-dimensional quantitative objectives. In CONCUR'12, LNCS 7454, p. 115–131, 2012.

[DG09] S. Demri and R. Gascon. The effects of bounding syntactic resources on Presburger LTL. *Journal of Logic and Computation*, 19(6):1541–1575, 2009.

[DLS10] S. Demri, R. Lazić, and A. Sangnier. Model checking memoryful linear-time logics over one-counter automata. *Theoretical Computer Science*, 411(22-24):2298–2316, 2010.

[EMSS92] E. A. Emerson, A. K.-L. Mok, A. P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. *Real-Time Systems*, 4:331–352, 1992.

[EN94] J. Esparza and M. Nielsen. Decidability issues for petri nets - a survey. *Bulletin of the EATCS*, 52:244–262, 1994.

[GHOW10] S. Göller, C. Haase, J. Ouaknine, and J. Worrell. Model checking succinct and parametric one-counter automata. In ICALP'10, LNCS 6199, p. 575–586. Springer, 2010.

[GL10] S. Göller and M. Lohrey. Branching-time model checking of one-counter processes. In STACS'10, Leibniz International Proceedings in Informatics 20, p. 405–416. Leibniz-Zentrum für Informatik, 2010.

[Haa12] C. Haase. *On the Complexity of Model Checking Counter Automata*. PhD thesis, University of Oxford, UK, 2012.

[HKOW09] C. Haase, S. Kreutzer, J. Ouaknine, and J. Worrell. Reachability in succinct and parametric one-counter automata. In CONCUR'09, LNCS 5710, p. 369–383. Springer, 2009.

[JLSØ13] K. Jensen, K. G. Larsen, J. Srba, and L. K. Østergaard. Local model checking of weighted CTL with upper-bound constraints. In SPIN'13, LNCS 7976. Springer, 2013.

[Koy90] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[Min61] M. L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Annals of Mathematics*, 74(3):437–455, 1961.

[Pnu77] A. Pnueli. The temporal logic of programs. In FOCS'77, p. 46–57. IEEE Comp. Soc. Press, 1977.

[QS82] J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In SOP'82, LNCS 137, p. 337–351. Springer, 1982.

[Ser06] O. Serre. Parity games played on transition graphs of one-counter processes. In FoSSaCS'06, LNCS 3921, p. 337–351. Springer, 2006.

[ZP96] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996.