Frequency Linear-time Temporal Logic

Benedikt Bollig
LSV, ENS Cachan, CNRS & INRIA
Cachan, France
bollig@lsv.ens-cachan.fr

Normann Decker Martin Leucker
Institute for Software Engineering and Programming Languages
Universität zu Lübeck, Lübeck, Germany
{*decker, leucker*}*@isp.uni-luebeck.de*

***Abstract*—We propose fLTL, an extension to linear-time temporal logic (LTL) that allows for expressing relative frequencies by a generalization of temporal operators. This facilitates the specification of requirements such as the deadlines in a real-time system must be met in at least 95% of all cases. For our novel logic, we establish an undecidability result regarding the satisfiability problem but identify a decidable fragment which strictly increases the expressiveness of LTL by allowing, e.g., to express non-context-free properties.**

***Keywords*-temporal logic; LTL; specification; frequency; availability; counters; symbolic tableau**

I. Introduction

Linear-time Temporal Logic (LTL) has been introduced to the setting of formal verification of computer programs in 1977 by Pnueli [1]. Meanwhile it has become a well-established specification formalism that is used in many different areas and for different purposes, but especially for verification. It is the basis of PSL (Property Specification Language, [2]), which is a standardized specification language used in the hardware domain. To foster its industrial application, it has been supported by specification patterns [3] and by syntactic sugar [4].

Despite its extensive use in many different application areas it has been noted that LTL has a limited expressiveness. Wolper [5] showed that LTL is not able to express all ω-regular properties. More specifically, he showed that LTL is not able to express that p holds in every other moment. In essence it means that LTL is not able to count.

Over the past decades LTL has been extended in many directions to enrich its expressiveness. Starting with Wolper [5], there is a line of work extending LTL's expressiveness to capture the regular ω-languages [6], [7], [8].

Another line of work extends the expressiveness of LTL towards quantitative measures. Demri considers LTL over integers rather than atomic propositions [9], giving the resulting logic the possibility to reason over sequences of integers and to allow counting modulo constants. In the context of probabilistic model checking, where a system description is given as a Markov chain, the set of traces satisfying an LTL formula is measured allowing one to give an idea about the probability to which extent the underlying formula holds [10]. A similar concern of giving a measure to which extent the formula is satisfied is also pursued in the setting of runtime verification. Here, however, only a single trace is given from which several finite behaviours of the underlying system are derived to estimate the probability to which extent the formula is satisfied. A notable work in this area is given by [11].

We also extend LTL to allow for the specification of quantitative means within our formal logic. In contrast to the existing work we do not extend the underlying structures towards integers nor do we rely on profound probability theory arguments. Our extension focuses on the *until*-operator present in LTL. The standard meaning of φ *until* ψ (denoted by $\varphi \,\mathrm{U}\, \psi$) is that there is a future moment in which a property ψ holds and up to this moment a property φ has to hold in each position. Our main idea is now to relax the number of positions in which φ has to hold by allowing to say that, e.g., only at 95 % of the positions φ has to hold.

A similar concept has been worked out by Hoenicke et al. in the setting of regular expressions [12]. Their notion of availability in finite words appears closely related but while the relationship between LTL-definable languages and regular ω-languages is well studied, the formal link between our logic fLTL and the so called availability expressions remains subject to further investigation.

An extension of LTL towards counters has also been considered recently by Laroussinie et al. [13]. In contrast to our approach, this is a syntactic variation in terms of expressiveness, as they reside in the class of LTL-definable ω-languages. There is, however, an interesting correspondence, which we point out in Section IV.

Acknowledgement

We thank the anonymous reviewer for valuable comments.

II. Preliminaries

Words: Let AP be a finite, non-empty set of *atomic propositions*. We consider words over the alphabet $\Sigma = 2^{\mathsf{AP}}$. Powers of finite words and letters are to be read in the common way, i.e. $w^0 = \varepsilon$ and $w^{n+1} = ww^n$. For an infinite word $w = a_0a_1a_2\ldots \in \Sigma^\omega$ ($a_i \subseteq \mathsf{AP}$) we denote the finite prefix of length n by $w|_n = a_0a_1\ldots a_{n-1}$ and the n-th true (infinite) suffix by $w|^n = a_na_{n+1}\ldots$. Thus, $w|_0 = \varepsilon$ and $w = w|^0 = w|_nw|^n$. This notation is analogously used for linear sequences in general, such as paths. We sometimes specify alphabets directly when propositions are not needed explicitly. The reader may assume any set AP that allows for

distinguishing propositionally at least the number of letters needed and possibly some more, which are not being used.

Formulae: Furthermore, we use letters $a \in \Sigma$ (i.e. sets of propositions) in formulae to keep them concise and readable. They abbreviate an exactly characterizing *conjunction* $(\bigwedge_{p\in a} p) \wedge (\bigwedge_{p\notin a} \neg p)$. In general, a *set* Γ *of formulae* is considered as $\bigwedge_{\varphi\in\Gamma}\varphi$. *Sets of letters* indicate their *disjunction*. For example, let M be the set of letters $\{a, b\}$. When used in a formula, we interpret M as

$$\Big((\bigwedge_{q\in a} q) \wedge (\bigwedge_{q\notin a} \neg q)\Big) \vee \Big((\bigwedge_{q\in b} q) \wedge (\bigwedge_{q\notin b} \neg q)\Big).$$

III. The temporal logic *f*LTL

The idea of *f*LTL is to allow for relaxation of the until operator in terms of an annotated frequency. The usual intuition for a formula $\varphi\,\mathrm{U}\,\psi$ is that ψ must hold at some point in the future, and before that φ has to hold *always*. Instead of "always", we consider the less strict formulation "sufficiently often" referring to a minimum ratio $c \in [0, 1]$ of positions, the *frequency* of φ.

Definition 1 (Syntax and semantics of *f*LTL)**.** *The syntax of Frequency Linear-time Temporal Logic (fLTL) formulae is given by*

$$\varphi ::= \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathrm{X}\,\varphi \mid \varphi\ \mathrm{U}^c\ \varphi \mid p \quad (p \in \mathsf{AP})$$

where each U*-operator is annotated by a rational number* $c \in \mathbb{Q}$ *with* $0 \le c \le 1$*. fLTL formulae are interpreted over words* $w \in \Sigma^\omega$*,* $w = a_0a_1a_2\ldots$ *as follows:*

$$\begin{array}{lll} w \models \top & & \\ w \models p & \textit{if} & p \in a_0 \quad (p \in \mathsf{AP}) \\ w \models \neg\varphi & \textit{if} & w \not\models \varphi \\ w \models \mathrm{X}\,\varphi & \textit{if} & w|^1 \models \varphi \\ w \models \varphi \wedge \psi & \textit{if} & w \models \varphi \textit{ and } w \models \psi \\ w \models \varphi\,\mathrm{U}^c\,\psi & \textit{if} & \exists_n : w|^n \models \psi \textit{ and} \\ & & \#_{\varphi,w}(n) \ge c \cdot n \end{array}$$

In the definition above we write

$$\#_{\varphi,w}(n) := |\{i \mid 0 \le i < n,\ w|^i \models \varphi\}|$$

for the number of positions before n satisfying a formula φ.

For $c = 1$, φ has to hold at all the positions before ψ holds, which coincides with the U-operator in LTL and we thus consider $\varphi\,\mathrm{U}\,\psi$ as abbreviation for the special case $\varphi\,\mathrm{U}^1\,\psi$. Note, that our definition is less strict as it could be regarding the eventuality. Consider, for example $p\,\mathrm{U}^{\frac{1}{2}}\,q$ and the word $w = \{r\}\{q\}\{p\}\{p\}\{q\}\{q\}\{r\}^\omega$. The observed frequency of p at positions 1 and 5 is $\frac{\#_{p,w}(1)}{1} = \frac{0}{1}$ and $\frac{\#_{p,w}(5)}{5} = \frac{2}{5}$, respectively, which is too low. Yet, at position 4, q is satisfied *and* the frequency constraint is met. w is thus a model since the frequency constraint is not necessarily required to hold at the first position where q holds. In our conclusion, we discuss a variant of *f*LTL that considers only the frequency up to the first satisfaction of ψ.

We use the standard abbreviations $\bot := \neg\top$ and $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$. Inspecting the negation $\neg(\varphi\,\mathrm{U}^c\,\psi)$ we obtain a notion of "sufficiently often $\neg\varphi$ releases $\neg\psi$ at that position" where "sufficiently often" amounts to a least frequency of $1 - c$. Hence, the dual operator for U^c can be seen as $\varphi\,\mathrm{R}^c\,\psi := \neg(\neg\varphi\,\mathrm{U}^{1-c}\,\neg\psi)$ and we confirm coincidence with the traditional R-operator for $c = 0$. Also note, that our definition is robust considering the operators F (eventually) and G (always). We can use the common definition $\mathrm{F}\,\varphi := \top\,\mathrm{U}\,\varphi$ with an implicit frequency $c = 1$ while any other frequency would neither formally nor intuitively change the semantics. Allowing $\top$ to hold less frequently ($c < 1$) does not change anything as it always holds. Dually, we let $\mathrm{G}\,\varphi := \neg\,\mathrm{F}\,\neg\varphi = \bot\,\mathrm{R}^c\,\varphi$ for $c = 0$ since $\bot$ does not hold, particularly not "more often" ($c > 0$).

IV. *f*LTL is not context-free

We observe that LTL is not only a syntactic but also a *true semantic fragment* of *f*LTL regarding ω-languages: Consider alphabets $\Sigma_n = \{a_1, \ldots, a_n, b\}$ and the family of languages

$$L_n = \{a_1^k a_2^k \ldots a_n^k b^\omega \mid k \in \mathbb{N}_0\},$$

which are not context-free for $n > 2$. While the LTL-definable languages are (strictly) contained in the class of regular ω-languages, each of the languages L_n is definable by an *f*LTL formula

$$\varphi_n = \left(\bigwedge_{i=1}^{n} a_i\,\mathrm{U}^{\frac{1}{n}}\,\mathrm{G}\,b\right) \wedge \left(\bigwedge_{i=1}^{n-1} \mathrm{G}(a_{i+1} \to \mathrm{G}\,\neg a_i)\right),$$

providing the following theorem.

Theorem 1. *fLTL can express non-context-free languages.*

Corollary 1. *fLTL can express non-context-free languages without nesting of frequency-until operators.*

At this point it is worth having a closer look at the work done by Laroussinie et al. in [13]. They consider an extension to LTL that allows for counting the number of positions where certain properties hold. They show that there is a translation from their logic, which they call CLTL, to LTL. This strictly distinguishes CLTL form *f*LTL. Interestingly it turns out that only a subtle change to their formalism effects the observed gap in terms of expressiveness.

CLTL allows for formulae $\varphi\,\mathrm{U}_{[C]}\,\psi$ where the until operator is annotated by a constraint of the form

$$C ::= \top \mid C \wedge C \mid \neg C \mid \textstyle\sum_i \alpha_i \cdot \#\eta_i \sim k$$

where $\sim \in \{<, \le, =, \ge, >\}$, k, α_i are *positive* integers and η_i is again a CLTL formula. Semantically, $w \models \varphi\,\mathrm{U}_{[C]}\,\psi$ for a word $w \in \Sigma^\omega$ if $\exists_n : w|^n \models \psi$ and $(w, n) \models C$ and $\forall_{0\le i<n} : w|^i \models \varphi$. To obtain the semantics of $(w, n) \models C$, the terms $\#\eta$ in C are interpreted as the number of positions before n that satisfy η, i.e. $\#_{\eta,w}(n)$.

For trying to express $\varphi\,\mathrm{U}^c\,\psi$ nonetheless in CLTL we could write $\varphi\,\mathrm{U}_{[C]}\,\psi$ and let $C = (\frac{\#\varphi}{\#\top} \ge c)$. Equivalently,

with $c = \frac{n}{m}$, we could write $C = (m \cdot \#\varphi - n \cdot \#\top \geq 0)$. Hence, allowing for *negative* weights α_i in constraints obviously completely changes the character of CLTL.

V. *f*LTL IS UNDECIDABLE

Considering the expressiveness of fLTL, the question arises whether it is decidable. One of our main contributions is the following answer.

Theorem 2. *Satisfiability of fLTL formulae is undecidable.*

In the remainder of this section we sketch a proof, which is inspired by [12]. Its main idea is to reduce the (undecidable) termination problem of two-counter Minsky machines [14] to the satisfiability problem of fLTL. More specifically, for a given Minsky machine $\mathcal{M}$, we construct a formula $\varphi_{\mathcal{M}}$ that expresses the (deterministic) computation of $\mathcal{M}$. Then, $\varphi_{\mathcal{M}} \wedge \mathrm{F}\, l_{\text{final}}$ is satisfiable, if and only if $\mathcal{M}$ terminates in line l_{final}, denoting the halting location of $\mathcal{M}$.

Minsky machines: A Minsky machine uses two instructions $\mathsf{inc}(K_i, l)$ and $\mathsf{dect}(K_i, l_1, l_2)$ where K_i refers to one of the two counters and $l, l_1, l_2 \in L$ are locations in the program. inc increases the counter and jumps to location l, while the dect instruction tests K_i for zero and directly jumps to l_1 in that case, or otherwise decreases K_i and the next instruction to be executed is the one at location l_2.

W.l.o.g. we consider instructions for each counter instead of using the counter as argument. Also, dect is split into a pure test for zero and a decrement instruction that has no effect, if a counter is zero. Thus, our instruction set is

$$\mathsf{IS} := \{\mathsf{inc}_1, \mathsf{inc}_2, \mathsf{dec}_1, \mathsf{dec}_2, \mathsf{testz}_1, \mathsf{testz}_2\}$$

A Minsky machine is a tuple $\mathcal{M} = (\pi, L, l_{\text{init}}, l_{\text{final}}, n_0, m_0)$ where L is a non-empty, finite set of *locations*, $l_{\text{init}}, l_{\text{final}} \in L$ are the initial and final location, respectively, $n_0, m_0 \in \mathbb{N}_0$ are the *initial counter values*, the *program* $\pi : L \to \mathsf{IS} \times L \times L$ is a mapping of locations to *commands* (tuples of instructions and locations). A *configuration* of $\mathcal{M}$ is a tuple $C = (l, n, m) \in L \times \mathbb{N}_0 \times \mathbb{N}_0$ representing the current location and the values of both counters. The *computation* of $\mathcal{M}$ is the unique, infinite sequence $C_0 \to C_1 \to C_2 \to \dots$ of configurations C_i, such that $C_0 = (l_{\text{init}}, n_0, m_0)$ is the initial configuration and for any $C_i = (l, n, m)$, the subsequent configuration C_{i+1} is computed according to the program π in the expected manner. Note, that the second location in a command is taken into account only by the testz instructions.

Encoding the computation: We encode counters unary into words by letters a or $\hat{a}$ for counter 1 and b or $\hat{b}$ for counter 2. a^n,b^m represent counter values n,m *before* and $\hat{a}^{n'}, \hat{b}^{m'}$ values n', m' *after* some operation.

We consider instructions always performing an explicit *operation* on each of the counters, e.g. inc_1 performs an incrementation on counter 1 and a "skip" operation on counter 2. These operations are represented in the encoding by letters from $\mathsf{aOP} := \{i_a, d_a, s_a\}$ for the first and from $\mathsf{bOP} := \{i_b, d_b\}$ for the second counter. i_a indicates an increase of the counter 1 and thus the number of as. s_a indicates a skip operation, namely keeping the number of as constant in the encoding and similarly for the other letters.

Note that, while using letter s_a for not modifying the number of as, we do *not* use an explicit letter s_b for a skip on counter 2. This is due to technicalities when counting letters in the following.

Depending on which instruction is specified by $\pi(l) = (\mathsf{is}, l', l'')$ we represent the computation e.g. as

$$\begin{aligned} &l\ a^n i_a \hat{a}^{n+1}\ b^m \hat{b}^m && \text{for } \mathsf{is} = \mathsf{inc}_1, \\ &l\ a^n s_a \hat{a}^n\ b^m \hat{b}^m && \text{for } \mathsf{is} = \mathsf{testz}_1 \text{ or} \\ &l\ a^n s_a \hat{a}^n\ b^m i_b \hat{b}^{m+1} && \text{for } \mathsf{is} = \mathsf{inc}_2. \end{aligned}$$

The computation

$$(l_0, n_0, m_0) \to (l_1, n_1, m_1) \to (l_2, n_2, m_2) \to \dots$$

of $\mathcal{M}$ is thereby represented as a word of the form

$$l_0\ a^{n_0}\ \mathsf{op}_a\ \hat{a}^{n_1}\ b^{m_0}\$\mathsf{op}_b\ \hat{b}^{m_1}\ l_1\ a^{n_1}\ \mathsf{op}'_a\ \hat{a}^{n_2}\ b^{m_1}\$\mathsf{op}'_b\ \hat{b}^{m_2}\ l_2\ a^{n_2}\ \mathsf{op}''_a\ \hat{a}^{n_3}\ b^{m_2}\$\mathsf{op}''_b\ \hat{b}^{m_3}\ \dots$$

where $l_0 = l_{\text{init}}$ is the initial location and $\mathsf{op}_a, \mathsf{op}'_a, \mathsf{op}''_a \in \mathsf{aOP}, \mathsf{op}_b, \mathsf{op}'_b \in \mathsf{bOP} \cup \{\varepsilon\}$. For purely technical reasons we add a separator sign \$ between the bs and $\hat{b}s$. The encoding yields the alphabet we use:

$$\Sigma = \{a, \hat{a}, b, \hat{b}, \$, i_a, i_b, d_a, d_b, s_a\} \cup L.$$

Ordering of symbols: In order to ensure the correct ordering of the symbols in the encoding of the computation we use a formula φ_{enc} that consist of the following conjuncts:

- Labels $l \in L$ and the letter a are followed by a or $\mathsf{op} \in \mathsf{aOP}$: $(L \vee a) \to \mathrm{X}(a \vee \mathsf{aOP})$.
- An operation $\mathsf{op} \in \mathsf{aOP}$ on counter 1 and letters $\hat{a}$ are followed by $\hat{a}$, b or \$: $(\mathsf{aOP} \vee \hat{a}) \to \mathrm{X}(\hat{a} \vee b \vee \$)$.
- Letter b is followed by another b or \$: $b \to \mathrm{X}(b \vee \$)$.
- After each \$ there is an operation on b, a number of $\hat{b}$ or the next label: $\$ \to \mathrm{X}(\mathsf{bOP} \vee \hat{b} \vee L)$.
- A label must follow directly after symbols $\hat{b}$, or directly after an operation on b: $(\mathsf{bOP} \vee \hat{b}) \to \mathrm{X}(\hat{b} \vee L)$.

Additionally, the computation has to start with the finite prefix $l_0\ a^{n_0}\ \mathsf{op}_a\ \hat{a}^{n_1}\ b^{m_0}\$$ according to the initial configuration. It can be described similarly by an LTL formula φ_I.

Specifying program instructions: Next, for $\mathsf{is} \in \mathsf{IS}$, $l_1, l_2 \in L$, we construct formulae $\varphi(\mathsf{is}, l_1, l_2)$. Assuming the correct ordering of symbols, i.e. in conjunction with $\mathrm{G}\,\varphi_{\text{enc}}$, these formulae enforce the correct number of $\hat{a}s$ and $\hat{b}s$ according to the number of as and bs and also the correct choice for the next location. E.g., the formula $\varphi(\mathsf{inc}_2, l_1, l_2)$ shall enforce the pattern $a^n s_a \hat{a}^n\ b^m \$ i_b \hat{b}^{m+1}\ l_1 \dots$ and $\varphi(\mathsf{testz}_1, l_1, l_2)$ enforces either $a^{n+1} s_a \hat{a}^{n+1}\ b^m \$ \hat{b}^m\ l_2 \dots$ $(K_1 > 0)$ or $s_a\ b^m \$ \hat{b}^m\ l_1 \dots$ $(K_1 = 0)$. The formula

$b\,\mathrm{U}^{\frac{1}{2}}\,l \wedge \hat{b}\,\mathrm{U}^{\frac{1}{2}}\,l$ enforces the pattern $b^m\hat{b}^m\ l\ldots$. Based on this idea we use

$$\beta_{\mathsf{inc}}(l) := (b \vee \$) \wedge \mathrm{X}((b \vee \$ \vee i_b)\,\mathrm{U}^{\frac{1}{2}}\,l \wedge \hat{b}\,\mathrm{U}^{\frac{1}{2}}\,l)$$

to express the effect of an inc_2 operation on counter 2, namely the pattern $b^m\$i_b\hat{b}^{m+1}\ l\ldots$. The complete formula must also impose equality between the number of as and $\hat{a}s$ which can be done similarly. Therefore we let

$$\varphi(\mathsf{inc}_2, l_1, l_2) :=$$
$$(a \vee s_a) \wedge \mathrm{X}\Big((a \vee s_a)\,\mathrm{U}^{\frac{1}{2}}(\beta_{\mathsf{inc}}(l_1)) \ \wedge\ \hat{a}\,\mathrm{U}^{\frac{1}{2}}\,\beta_{\mathsf{inc}}(l_1)\Big)$$

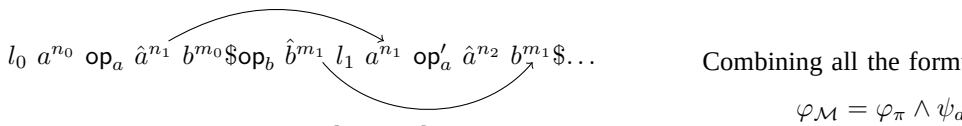
stating that there must be a sequence of letters a, s_a followed by the pattern form $\beta_{\mathsf{inc}}(l_1)$ as above. Additionally, between the first position and the beginning of the $\beta_{\mathsf{inc}}(l_1)$ pattern, half of the positions must carry an $\hat{a}$ and the other half and the first position must carry either a or s_a. Assuming, in virtue of φ_{enc}, that s_a occurs exactly once, the overall number of as and $\hat{a}$s must thus be equal.

The other instructions can be reflected similarly and we compose these instruction formulae according to the program π to a formula

$$\varphi_\pi := \mathrm{G}\Big(\bigwedge_{l\in L} l \to \mathrm{X}\,\varphi(\pi(l))\Big).$$

which enforces that any model must mimic the instructions of the Minsky machine at every position where an according label occurs.

Propagation: From each computation step to the next, we need to ensure that the result of a computation, i.e. the powers of $\hat{a}^n$ and $\hat{b}^m$, are *copied* correctly.



Assuming, again, correct encoding and computation, we observe an invariant in between the blocks of $\hat{a}$s and as, i.e. the sub-term $b^m\$\mathsf{op}_b\hat{b}^{m'}l$. If $\mathsf{op}_b = i_b$, then the number of occurrences of letters from the set $A_{\text{left}} := \{\hat{a}, b, \$, i_b\}$ is equal to the number of letters from the set $A_{\text{right}} := \{\hat{b}, a, d_b\}\cup L$ since $m' = m+1$. This also holds for $\mathsf{op}_b = d_b$, where $m' + 1 = m$, and $\mathsf{op}_b = \varepsilon$ with $m' = m$. Thus, $n = n'$ in $\hat{a}^n b^m\$\mathsf{op}_b\hat{b}^{m'}la^{n'}$ iff the number of occurring letters from A_{left} equals the number of occurring letters from A_{right}. Thus we can ensure the correct propagation of the value of counter 1 (in terms of as) independently of op_b by

$$\psi_a := \mathrm{G}\Big(\mathsf{aOP} \to \mathrm{X}\big((A_{\text{left}}\,\mathrm{U}^{\frac{1}{2}}\,\mathsf{aOP}) \wedge (A_{\text{right}}\,\mathrm{U}^{\frac{1}{2}}\,\mathsf{aOP})\big)\Big).$$

For propagating the value of the second counter we enforce the pattern $\ldots\hat{b}^m\ l\ a^n\ \mathsf{op}_a\hat{a}^{n'}b^m\$\ldots$. To the left of op_a we find the symbols $B_{\text{left}} := \{\hat{b}, a\} \cup L$ and to the right $B_{\text{right}} := \{\hat{a}, b\}$. We distinguish three cases for op_a.

Case $\mathsf{op}_a = i_a$: Counter 1 is increased and we have the pattern $\hat{b}^m\ l\ a^n i_a\hat{a}^{n+1}b^{m'}$. We see that $m = m'$ if and only if the number of occurrences of symbols from B_{right} is equal to the number of of symbols from the set $B_{\text{left}}\cup\{i_a\}$ minus one which is expressed by the formula

$$\psi_{\mathsf{inc}} := \mathrm{X}\Big(\big((B_{\text{left}} \vee i_a)\,\mathrm{U}^{\frac{1}{2}}\,\$\big) \ \wedge\ (B_{\text{right}}\,\mathrm{U}^{\frac{1}{2}}\,\$)\Big).$$

Case $\mathsf{op}_a = d_a$: If the counter is decreased we have $\hat{b}^m\ l\ a^{n+1}d_a\hat{a}^n b^{m'}$ and we enforce exactly one more symbol from B_{left} than from $B_{\text{right}}\cup\{d_a\}$ by

$$\psi_{\mathsf{dec}} := \mathrm{X}\Big((B_{\text{left}}\,\mathrm{U}^{\frac{1}{2}}\,\$) \wedge \big((B_{\text{right}} \vee d_a)\,\mathrm{U}^{\frac{1}{2}}\,\$\big)\Big).$$

Note that in order to represent the offset of one we can safely remove the first symbol using the X-operator and then enforce an equal number because we know the pattern starts with at least one symbol from B_{left}.

Case $\mathsf{op}_a = s_a$: A neutral operation on a yields the pattern $\hat{b}^m\ l\ a^n s_a\hat{a}^n b^{m'}$ and we can guarantee equality of m and m' by an equal number of symbols from B_{left} and $B_{\text{right}}\cup\{s_a\}$:

$$\psi_{\mathsf{skip}} := \big(B_{\text{left}}\,\mathrm{U}^{\frac{1}{2}}\,\$\big) \wedge \big((B_{\text{right}} \vee s_a)\,\mathrm{U}^{\frac{1}{2}}\,\$\big).$$

Combining the cases, we obtain a formula which expresses that in any case, the propagation is done correctly. The property must always hold right after the position of the operator symbol for counter 2. Recall, that these symbols for counter 2 might not be explicitly present since the skip operation was expressed by ε. We therefore describe the position that triggers the copy-property for $\hat{a}$ by either finding symbol from $\mathsf{bOP} = \{i_b, d_b\}$ explicitly or just the \$ symbol without a following operation from bOP:

$$\psi_b := \mathrm{G}\Big((\mathsf{bOP} \vee (\$ \wedge \neg\,\mathrm{X}\,\mathsf{bOP})) \to \mathrm{X}\big(\psi_{\mathsf{inc}} \vee \psi_{\mathsf{dec}} \vee \psi_{\mathsf{skip}}\big)\Big).$$

Combining all the formulae we obtain

$$\varphi_{\mathcal{M}} = \varphi_\pi \wedge \psi_a \wedge \psi_b \wedge (\mathrm{G}\,\varphi_{\text{enc}}) \wedge \varphi_I$$

describing exactly the computation of $\mathcal{M}$ and $\varphi_{\mathcal{M}} \wedge \mathrm{F}\,l_{\text{final}}$ is satisfiable if and only if $\mathcal{M}$ eventually reaches the final location, i.e. terminates. This proves Theorem 2.

VI. A DECIDABLE FRAGMENT OF *f*LTL

In this section we consider the *f*LTL fragment of *simple frequentness properties* (*sf*LTL) that is still more expressive than LTL but has a *decidable* satisfiability problem. Formulae in that fragment are restricted in terms of nesting and negation and have the form

$$\varphi ::= \psi_{\text{LTL}} \mid (\psi_{\text{LTL}})\,\mathrm{U}^c(\varphi) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{X}\,\varphi$$

where ψ_{LTL} denotes a standard LTL formula, which we assume w.l.o.g. to be in positive normal form, i.e. where negation occurs only in front of atomic propositions. Hence, for *sf*LTL we consider positive boolean combinations of formulae including frequency-until operators that are not nested in the first operand. Note, that *sf*LTL is still more expressive than LTL since the non-context-free properties

used to show that *f*LTL is more expressive than LTL (Theorem 1) do neither rely on nesting nor negating frequency-until formulae.

In the remainder of this section we show the following.

Theorem 3. *The satisfiability problem of sfLTL is decidable.*

We outline a decision procedure that reduces satisfiability of *sf*LTL formulae to the integer linear programming problem, i.e. to solving systems of linear inequalities, which can be solved by using known algorithms, c.f. [15, Part IV]. We prove that there is a model to the formula if and only if the constructed inequality systems have a (natural) solution. The reduction comprises three steps:

1) We first introduce a notion of our logic using counters.
2) Based on that we build a labelled tableau graph from a given formula and check satisfiability by solving constrained reachability problems in that graph.
3) We construct integer linear programs in order to find paths that obey constraints imposed by the edge labels.

A. Counter semantics for fLTL

We rely on a notion of unfolding in order to build a tableau graph for *f*LTL formulae. Therefore, we consider *f*LTL formulae enriched with a *counter value* $k \in \mathbb{Q}$ for the frequency-until-operators, written $\varphi \,\mathrm{U}^c_k\, \psi$. The annotated counter value can be seen as bias to such a (sub)-formula that reflects some *history*. Intuitively, in the case of $k = 0$ the history is balanced, i.e. neither additional credit should be rewarded for satisfying the obligation φ more often than needed, nor was a due recorded by unfulfilling the obligation too often. A formula $\varphi \,\mathrm{U}^c_0\, \psi$ therefore coincides with the *f*LTL formula without annotation.

Formally, we let $w \models \varphi \,\mathrm{U}^c_k\, \psi$ if and only if

$$\exists_n : w|^n \models \psi \text{ and } k + \#_{\varphi,w}(n) \geq c \cdot n.$$

If an obligation φ is known to hold or known to not hold at the first position, a formula $\varphi \,\mathrm{U}^c_k\, \psi$ can be rewritten – just as the standard unfolding[1] – by an X-formula requiring the very same formula to hold at the next position, though the fact of fulfilling or unfulfilling the obligation φ, respectively, should be recorded in terms of the bias. We hence extend the notion of unfolding to the counter setting.

Definition 2 (Counted unfolding)**.** *Let Φ be an fLTL formula. The unfolding of LTL formulae is extended to the counter semantics as follows.*

$$\mathsf{unf}(\Phi) := \begin{cases} \psi \vee (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}^c_{k+1-c}\, \psi)) & \text{if } \Phi = \varphi \,\mathrm{U}^c_k\, \psi \\ \qquad \vee \mathrm{X}(\varphi \,\mathrm{U}^c_{k-c}\, \psi) & \text{and } k \geq 0 \\ (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}^c_{k+1-c}\, \psi)) & \text{if } \Phi = \varphi \,\mathrm{U}^c_k\, \psi \\ \qquad \vee \mathrm{X}(\varphi \,\mathrm{U}^c_{k-c}\, \psi) & \text{and } k < 0 \\ \Phi & \text{otherwise.} \end{cases}$$

[1]The definition of LTL implies the equality $\varphi \,\mathrm{U}\, \psi \equiv \psi \vee (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}\, \psi))$.

Lemma 1 (Counted unfolding equivalence)**.** *Let Φ be a (possibly biased) fLTL formula and $w \in \Sigma^\omega$. Then, $w \models \Phi$ if and only if $w \models \mathsf{unf}(\Phi)$.*

Proof: For $\Phi \neq \varphi \,\mathrm{U}^c_k\, \psi$ the unfolding does not affect the formula and the result follows trivially. Therefore, let $w \models \Phi = \varphi \,\mathrm{U}^c_k\, \psi$ for some word $w \in \Sigma^\omega$.

Case 1: $k \geq 0$**.** By definition we have $w \models \varphi \,\mathrm{U}^c_k\, \psi$ iff $\exists_{n \geq 0} : (w|^n \models \psi$ and $k + \#_{\varphi,w}(n) - c \cdot n \geq 0)$, where we can distinguish the cases $n = 0$ and $n + 1 \geq 0$. With $k \geq 0$ we have that $w \models \psi$ or $\exists_{n \geq 0} : w|^{n+1} \models \psi$ and $k + \#_{\varphi,w}(n+1) - c \cdot (n+1) \geq 0$. Note, that $w|^{n+1} = w|^1|^n$.

The value of $\#_{\varphi,w}(n+1)$ is $\#_{\varphi,w|^1}(n)$ if $w \not\models \varphi$ or otherwise $\#_{\varphi,w|^1}(n) + 1$.

We conclude that $w \models \varphi \,\mathrm{U}^c_k\, \psi$ if and only if $w \models \psi$ or

$$\begin{aligned} w \models \varphi \text{ and } \exists_n : \quad & w|^1|^n \models \psi \text{ and} \\ & k + (1-c) + \#_{\varphi,w|^1}(n) - c \cdot n \geq 0 \\ & \text{or} \\ w \not\models \varphi \text{ and } \exists_n : \quad & w|^1|^n \models \psi \text{ and} \\ & k + (-c) + \#_{\varphi,w|^1}(n) - c \cdot n \geq 0 \end{aligned}$$

which reduces by the definition of the U- and X-operator to

$$w \models \psi \text{ or } \left(\begin{array}{ll} w \models \varphi \text{ and } w \models \mathrm{X}(\varphi \,\mathrm{U}^c_{k+1-c}\, \psi) & \text{or} \\ w \not\models \varphi \text{ and } w \models \mathrm{X}(\varphi \,\mathrm{U}^c_{k-c}\, \psi) & \end{array} \right)$$

and further to

$$\begin{aligned} w \models \psi \vee (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}^c_{k+1-c}\, \psi)) \vee \mathrm{X}(\varphi \,\mathrm{U}^c_{k-c}\, \psi) & \\ =^{(k \geq 0)} \mathsf{unf}(\varphi \,\mathrm{U}^c_k\, \psi). & \end{aligned}$$

Case 2: $k < 0$**.** The second case is almost identical. For the last step we obtain

$$\begin{aligned} w \models (\varphi \wedge \mathrm{X}\, \varphi \,\mathrm{U}^c_{k+1-c}\, \psi) \vee \mathrm{X}(\varphi \,\mathrm{U}^c_{k-c}\, \psi) & \\ =^{(k<0)} \mathsf{unf}(\varphi \,\mathrm{U}^c_k\, \psi) & \end{aligned}$$

■

B. Graph representation

Given our notion of unfolding we can pursue a tableau construction in the style of [16] in order to check satisfiability of a formula. In contrast to the common setting, we face a potentially infinite number of reachable states in the tableau graph due to the annotated counter values that can in- and decrease arbitrarily.

Our variant therefore constructs a *symbolic* tableau using the rules shown in Figure 1. Starting from the initial formula, disjunctions are split up into two child nodes and conjunctions are written as sets of (sub-)formulae.

As opposed to computing an explicit counter value for formulae in successor nodes we label the edges with the according operation that is performed. Recall that the unfolding of a formula $\varphi \,\mathrm{U}^c_K\, \psi$ depends on the actual value of K and so do the reachable successor nodes. Since we handle the counters symbolically, we label the edges to such nodes with constraints. That is, an edge to a successor that is only reachable if the value of K is greater or equal to zero is labelled with the constraint "$K \geq 0$".

$$\frac{\{\varphi\,\mathrm{U}_K^{c_K}\,\psi\}\cup\Gamma}{\{\varphi,\mathrm{X}(\varphi\,\mathrm{U}_K^{c_K}\,\psi)\}\cup\Gamma}\;{\scriptstyle K:=K+1-c_K}\qquad\frac{\{\varphi\,\mathrm{U}_K^{c_K}\,\psi\}\cup\Gamma}{\{\psi\}\cup\Gamma}\;{\scriptstyle K\geq 0}$$

$$\frac{\{\varphi\,\mathrm{U}_K^{c_K}\,\psi\}\cup\Gamma}{\{\mathrm{X}(\varphi\,\mathrm{U}_K^{c_K}\,\psi)\}\cup\Gamma}\;{\scriptstyle K:=K-c_K}\qquad\frac{\{\varphi\,\mathrm{U}\,\psi\}\cup\Gamma}{\{\psi\wedge(\varphi\wedge\mathrm{X}(\varphi\,\mathrm{U}\,\psi))\}\cup\Gamma}$$

$$(i=0,1)\;\frac{\{\varphi_0\vee\varphi_1\}\cup\Gamma}{\{\varphi_i\}\cup\Gamma}\qquad\frac{\{\psi\,\mathrm{R}\,\varphi\}\cup\Gamma}{\{\varphi\wedge(\psi\vee\mathrm{X}(\psi\,\mathrm{R}\,\varphi))\}\cup\Gamma}$$

$$\frac{\{\varphi_0\wedge\varphi_1\}\cup\Gamma}{\{\varphi_0,\varphi_1\}\cup\Gamma}\qquad\frac{\{\mathrm{X}\,\varphi_1,\ldots,\mathrm{X}\,\varphi_r,q_1,\ldots,q_m\}}{\{\varphi_1,\ldots,\varphi_r\}}\;{\scriptstyle \{q_1,\ldots,q_m\}}$$

Figure 1. Rules for building the symbolic tableau for an *sf*LTL formula. The X-rule includes possibly negated propositions $q_i \in \{p, \neg p \mid p \in \mathsf{AP}\}$.

Definition 3 (Symbolic tableau)**.** *Let Φ be an sfLTL formula and $\widehat{\Phi}$ the same formula, except that all frequency-until sub-formulae $\varphi\,\mathrm{U}^c\,\psi$ ($c<1$) are* uniquely *annotated by indices $K_1,\ldots,K_n$. The* symbolic tableau *is an edge-labelled graph $\mathcal{G}(\Phi)=(V,E,\lambda)$ where $V\subseteq 2^{\mathsf{sub}(\widehat{\Phi})}$ is the set of nodes of which $\{\widehat{\Phi}\}\in V$ is initial. The set of edges $E\subseteq V\times V$ and the labelling function λ, which associates with every edge $e\in E$ a label $\lambda(e)$, are defined according to the rules shown in Figure 1.*

A node $v\in V$ is called final*, if v contains only pure LTL formulae and the conjunction $\bigwedge_{\varphi\in v}\varphi$ is satisfiable.*

We use $\mathsf{sub}(\Phi)$ to denote the set of sub-formulae of a formula Φ, including possible unfoldings, and c_K to refer to the frequency of a particular frequency-until (sub-)formula with counter index K.

Recall that nestings of frequency-until operators like $(\varphi\,\mathrm{U}_{K_1}^{c_{K_1}}\,\psi_1)\,\mathrm{U}_{K_2}^{c_{K_2}}\,\psi_2$ are not allowed within the *sf*LTL fragment. Without that restriction we would need to individually distinguish (i.e. consider a new counter variable K_i for) each instance of the sub-formula $\varphi\,\mathrm{U}_{K_1}^{c_{K_1}}\,\psi_1$ which is "reproduced" through unfolding. The state space of the tableau would again be possibly infinite. The restriction allows us to directly identify and uniquely index all occurring frequency-until operators in the symbolic tableau since they are already present in the initial formula.

Given a path to a node v in the symbolic tableau, we can compute the actual value of K for a formula $\varphi\,\mathrm{U}_K^{c_K}\,\psi\in v$ by applying all operations on K that occur along the path to the initial value of K, which is 0. Hence, we can also explicitly check whether a certain restricted edge can actually be taken at a specific position on a path.

Definition 4 (Paths)**.** *A* path *in a symbolic tableau graph $\mathcal{G}(\Phi)$ is a finite sequence $\varrho=v_0v_1\ldots v_n$ of nodes $v_i\in V$, such that $(v_i,v_{i+1})\in E$ for $i=0,\ldots,n-1$. We call ϱ* simple*, if $v_i=v_j$ implies $i=j$. We call ϱ a (simple)* loop *if $v_0=v_n$ (and $v_0\ldots v_{n-1}$ is simple). The values that are added to a certain counter K along ϱ are denoted by*

$$\delta_K(v_i)=\begin{cases}1-c_K & \text{if } \lambda(v_i,v_{i+1})=\text{“}K:=K+1-c_K\text{”}\\ -c_K & \text{if } \lambda(v_i,v_{i+1})=\text{“}K:=K-c_K\text{”}\\ 0 & \text{otherwise}\end{cases}$$

The functions δ_K represent the weight of a node with respect to a specific counter K and are extended to paths by $\delta_K(\varrho):=\sum_{i=0}^{n-1}\delta_K(v_i)$.

The path ϱ is called valid *if (1) every label $\lambda(v_i,v_{i+1})=$ “$\{q_1,\ldots,q_m\}$” is non-contradictory, i.e. $q_r\neq\neg q_s$ for $r,s=1,\ldots,m$ and (2) for every label $\lambda(v_i,v_{i+1})=$ “$K\geq 0$” we have $\delta_K(\varrho|_{i+1})\geq 0$.*

That is, the weight of the path $\varrho|_{i+1}$ (ending with v_i) satisfies the corresponding condition and the nodes on ϱ can actually be traversed in that order.

Theorem 4. *An sfLTL formula Φ is satisfiable if and only if there exists a valid path from the initial to a final node in the symbolic tableau graph $\mathcal{G}(\Phi)$.*

Proof: **(⇐).** Assume there is a valid path $\varrho=v_0\ldots v_n$ to a final node v_n in the graph $\mathcal{G}(\Phi)$. W.l.o.g. Φ can be assumed to be in positive normal form, i.e. negation occurs only in front of atomic propositions. The rules used for constructing the graph refine Φ to an under-approximation by dismissing one side from disjunctions. Thus, if there is a model for a node v, i.e. some $w\in\Sigma^\omega$ s.t. $w\models\bigwedge_{\varphi\in v}\varphi$, then there is a model w' for every predecessor of v: The $\wedge$-rule does not effect any change since the set is interpreted as conjunction. Also, let Γ be a set of formulae, then every model for $\varphi_1\wedge\Gamma$ is also a model for $(\varphi_1\vee\varphi_2)\wedge\Gamma$. Exchanging U- and R-formulae without frequency with their respective unfoldings is purely syntactical and w remains a model. The X-rule only applies to nodes of the form $\{\mathrm{X}\,\varphi_1,\ldots,\mathrm{X}\,\varphi_r,q_1,\ldots,q_m\}$ and for any $w\models\varphi_1\wedge\ldots\wedge\varphi_r$ we have that $aw\models\mathrm{X}\,\varphi_1\wedge\ldots\wedge\mathrm{X}\,\varphi_r\wedge q_1\wedge\ldots\wedge q_m$ if $a=\{q_1,\ldots,q_m\}\cap\mathsf{AP}$. We can dismiss all negated propositions since the path was valid and $\{q_1,\ldots,q_m\}$ is thus not contradictory.

The remaining rules unfold frequency-until-formulae. Respecting the according constraints ensures the correct bias. Thus, by Lemma 1, they at most refine the formula, which means that any model for the child node is in particular a model of the parent node, given the constraint is satisfied or the according operation is performed on the counter, respectively.

We hereby construct a model directly from a valid path ϱ, starting with a model w_{v_n} for the last node v_n in the path (which is final) and backwards prepending the letters imposed by the propositional constraints of the applied X-rules (see Figure 2).

(⇒). Assume Φ is satisfiable and a word $w_0\in\Sigma^\omega$ is a model. We can derive a valid path $\varrho=v_0v_1v_2\ldots v_n$ on

$$\begin{array}{ccccccccccc} v_0 & \dots & v_{i_1} & \xrightarrow[l_{i_1}]{\mathrm{X}} & v_{i_1+1} & \dots & v_{i_2} & \xrightarrow[l_{i_2}]{\mathrm{X}} & v_{i_2+1} & \dots & \xrightarrow[l_{i_j}]{\mathrm{X}} \dots \quad v_n \\ & & \downarrow & & & & \downarrow & & & & \downarrow \\ w_\varphi := & & a_1 & & & & a_2 & & & \dots & a_j \dots \quad w_{v_n} \end{array}$$

Figure 2. Following a valid path ϱ directly yields a model if the last position is satisfiable by some word w_{v_n}, which is in particular the case if v_n is final, i.e. represents a pure, satisfiable LTL formula. Negated propositions in labels l_{i_j} are included implicitly in the letters a_j by the absence of their positive duals.

$\mathcal{G}(\Phi)$ to a final node v_n.

We construct ϱ by starting with a path $\varrho_0 = v_0$ that only consists of the initial node. Guided by w_0 we choose a successor v_1 for v_0 and obtain a new path $\varrho_1 = v_0 v_1$. Also, we obtain a word w_1 that shall guide us in the next step. By structural induction on the construction of the tableau graph $\mathcal{G}(\Phi)$ we show that we can always choose a child node v_{i+1} of a node v_i on the path while maintaining the invariant that $\varrho_{i+1} = v_0 \dots v_{i+1}$ is valid and there is a word $w_{i+1} \models v_{i+1}[\sigma(\varrho_{i+1})]$. Here, we let

$$\sigma(\varrho_j) := \{K_1 \mapsto \delta_{K_1}(\varrho_j), \dots, K_r \mapsto \delta_{K_r}(\varrho_j)\}$$

be a substitution that maps the variables K_i to the values they would have after applying all operations on K_i that occur along a path ϱ_j. That substitution is used to substitute all counter variables by actual values in a particular node in order to get a semantics regarding a word. In general, for a formula formula φ and a finite substitution $\sigma = \{K_1 \mapsto k_1, \dots, K_r \mapsto k_r\}$ we write $\varphi[\sigma] = \varphi[k_1/K_1, \dots, k_r/K_r]$ for a formula that is equal to φ but where all occurrences of variables K_i are replaced by values k_i. Note that this is well defined here, regardless of the order of substitution.

The construction so far may yield an infinite path through some loop. However, this would mean that some frequency-until is unfolded infinitely often. The assumption that w_0 is a model for Φ and the existential definition of the U^c-operator ensures that we can finally dismiss the recurring frequency-until-formula during the unfolding of all such formulae and thus reach a node that only contains LTL formulae.

If $\Phi \in$ LTL, the initial node v_0 is already final and we found a valid path of length one. Now, let $\varrho_i = v_0 v_1 \dots v_i$ be a valid path and $w_i \models v_i[\sigma(\varrho_i)]$. We are done if $v_i \subseteq$ LTL. Otherwise we consider the following cases. (1) $v_i = \{\varphi_1 \wedge \varphi_2\} \cup \Gamma$, $\varphi_1 \wedge \varphi_2 \notin$ LTL. Choosing the successor $v_{i+1} := \{\varphi_1, \varphi_2\} \cup \Gamma$ and $w_{i+1} := w_i \models v_{i+1}[\sigma(\varrho_i v_{i+1})]$ maintains the invariant. Otherwise, if (2) $v_i = \{\varphi_1 \vee \varphi_2\} \cup \Gamma$, $\varphi_1 \vee \varphi_2 \notin$ LTL we take either $v_{i+1} := \{\varphi_1\} \cup \Gamma$ or $v_{i+1} := \{\varphi_2\} \cup \Gamma$. For at least one of them $w_{i+1} := w_i \models v_{i+1}[\sigma(\varrho_i v_{i+1})]$ must hold. Similarly, for (3) $v_i = \{\varphi \,\mathrm{U}\, \psi\} \cup \Gamma$ or $v_i = \{\psi \,\mathrm{R}\, \varphi\} \cup \Gamma$ we can safely choose the successor node $\{\psi \vee (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}\, \psi))\} \cup \Gamma$ or $\{\varphi \wedge (\psi \vee \mathrm{X}(\psi \,\mathrm{R}\, \varphi))\} \cup \Gamma$, respectively, since this unfolding does not change the semantics and $w_{i+1} := w_i$ remains a model. If neither of the previous cases applies, then consider (4) $v_i = \{\varphi \,\mathrm{U}^c_K\, \psi\} \cup \Gamma$, $c < 1$. Among the successor nodes in the graph, we have $x = \{\psi\} \cup \Gamma$, $y = \{\varphi, \mathrm{X}(\varphi \,\mathrm{U}^c_K\, \psi)\} \cup \Gamma$ and $z = \{\varphi, \mathrm{X}(\varphi \,\mathrm{U}^c_K\, \psi)\} \cup \Gamma$. By Lemma 1, $\delta_K(\varrho_i) \geq 0$ and $w_i \models x[\sigma(\varrho_i)]$ or w_i is a model for y or z respecting the according operations on K, i.e. $w_i \models y[\sigma(\varrho_i y)]$ or $w_i \models z[\sigma(\varrho_i z)]$. We can choose a suitable one as v_{i+1}. In any case $\varrho_{i+1} = \varrho_i v_{i+1}$ remains a valid path, and $w_{i+1} := w_i \models v_{i+1}[\sigma(\varrho_{i+1})]$. The case remaining is (5) $v_i = \{\mathrm{X}\,\varphi_1, \dots \mathrm{X}\,\varphi_r, q_1, \dots, q_m\}$ with $q_i \in \{p, \neg p \mid p \in \mathsf{AP}\}$. Since $w_i \models v_i[\sigma(\varrho_i)]$, we take $w_{i+1} := w_i|^1 \models v_{i+1} = \{\varphi_1, \dots, \varphi_r\}$ and the path remains valid since $\{q_1, \dots, q_m\}$ can not be contradictory. ■

C. Solving reachability with constraints

Theorem 4 reduces the satisfiability problem of *sf*LTL formulae to (constrained) reachability in the tableau graph.

In order to check whether there is a valid path to a final node we examine all simple paths to final nodes. Any valid path must then be an extension of some simple path by a number of loops. More precisely, we can assume these loops to be simple loops that are possibly extended by a number of such loops themselves. This hierarchy is of bounded depth since the valid path itself is finite.

Let $\varrho = v_0 \dots v_r$ be a path on a tableau graph $\mathcal{G}$. We denote $\mathsf{loops}_\mathcal{G}(\varrho)$ the set of all simple loops $l = u_0 \dots u_k u_0$ on $\mathcal{G}$, $u_i \in V$, s.t. u_0 occurs on ϱ. Further, let $\mathsf{loops}^*_\mathcal{G}(\varrho)$ be the smallest set s.t. $\mathsf{loops}_\mathcal{G}(\varrho) \subseteq \mathsf{loops}^*_\mathcal{G}(\varrho)$ and $l \in \mathsf{loops}^*_\mathcal{G}(\varrho) \Rightarrow \mathsf{loops}_\mathcal{G}(l) \subseteq \mathsf{loops}^*_\mathcal{G}(\varrho)$.

We construct equation systems that are imposed by the constraints on the path. If there were no constraints, simple paths would be fine already. For a path to be valid, any constraint edge must be preceded by a prefix-path with a weight satisfying the respective constraint (c.f. Definition 4). Hence, for each final node v_f we investigate all *simple* paths $\varrho = v_0 v_1 \dots v_n$ that start at the initial node and end in $v_n = v_f$ and consider the extending loops $l_m \in \mathsf{loops}^*_\mathcal{G}(\varrho|_{i+1})$ before a node v_{i+1} on ϱ. These loops can possibly influence the actual counter value when reaching v_i. Therefore we introduce a variable n_m for the number of traversals of a loop l_m. For every edge (v_i, v_{i+1}) on the so far simple path ϱ that is labelled by a constraint $\lambda(v_i, v_{i+1}) =$ "$K \geq 0$", we construct an inequality

$$\delta_K(\varrho|_{i+1}) + \Big(\sum_{l_m \in \mathsf{loops}^*_\mathcal{G}(\varrho|_{i+1})} n_m \cdot \delta_K(l_m) \Big) \geq 0$$

that represents all those paths that follow ϱ but may additionally traverse some of the loops.

If ϱ is extended by a subordinate loop $l_{r'} \in \mathsf{loops}_G(l_r)$ of some loop l_r, ϱ is necessarily extended by l_r itself at least once. Thus, for each such pair, we extend the equation system by this implication: $n_{r'} = 0 \vee n_r > 0$. Note that equations in a system are considered in conjunction but we can resolve the disjunction by using a copy of the system for each disjunct and obtain a set $\mathsf{Eqn}(\varrho)$ of equation systems.

There is a natural solution, in terms of the variables n_m, for at least one of them iff ϱ can be extended by loops in order to satisfy all constraints along it. Furthermore, if we consider the union $\mathsf{Eqn}(\mathcal{G}(\Phi)) = \bigcup_{\varrho} \mathsf{Eqn}(\varrho)$ for all simple paths ϱ to a final node in $\mathcal{G}(\Phi)$, there is a system $\mathsf{Eqn}(\varrho) \in \mathsf{Eqn}(\mathcal{G}(\Phi))$ that has a natural solution iff there is any path that can be successfully extended to satisfy every constraint along it and hence iff Φ is satisfiable.

Theorem 5. *An sfLTL formula Φ is satisfiable iff there is an equation system $\mathsf{Eqn}(\varrho) \in \mathsf{Eqn}(\mathcal{G}(\Phi))$ that has a natural solution.*

This completes the reduction and proves Theorem 3.

VII. Conclusion

Motivated by our experience of applying LTL in the setting of verification, we introduced and studied a generalization of LTL's until operator by relaxing the obligation φ of a formula $\varphi \,\mathrm{U}\, \psi$. In the resulting logic, which we call frequency LTL (fLTL), it is now possible to write $\varphi \,\mathrm{U}^c\, \psi$ to denote that φ has to hold at least at a fraction c of positions up to a moment in which ψ holds.

We have shown that fLTL is far more expressive than plain LTL as it allows us to formulate non-context-free properties. The price to pay for this expressiveness is that satisfiability of fLTL is undecidable. We show this using an encoding of the termination problem of Minsky machines. Expressing a frequentness within a certain scope via the frequency-until operator allows us to simulate counters by comparing numbers of letters, similar to [12].

We also identified a decidable fragment of fLTL which still allows the formulation of context-free properties and thus extends LTL's expressiveness. The decidability proof is given via a tableau construction. However, a typical approach, as for example introduced for LTL in [16], fails as fLTL's counters would result in infinite tableaux. To overcome this problem we introduced a *symbolic* tableau construction. As plain reachability algorithms are not suitable anymore to find valid paths in the according tableau graph we encode the tableau as integer linear programs. In other words, we reduce the satisfiability problem of fLTL to ILP, which may then be solved using standard algorithms.

One might change the semantics of fLTL's until operator, in a formula $\varphi \,\mathrm{U}^c\, \psi$, by considering the frequency of φ only up to the very *first* position in which ψ holds. An easy inspection of the undecidability proof shows, however, that the satisfiability problem of the resulting logic remains undecidable. This indicates a certain robustness of the concepts developed in this paper.

The work initiated in this paper may be extended in several directions. First, it would be interesting to reduce the gap between the decidable fragment *sf*LTL and fLTL itself. To this end, it might be worth to study also a variant of fLTL in which no credit is given in the following sense: For a formula $\varphi \,\mathrm{U}^c\, \psi$, a position may only violate φ if φ has been satisfied often enough in previous positions. Finally, in [8], LTL's expressiveness has been extended to capture the full class of regular ω-languages. It might be worthwhile to extend also this logic by a concept of frequencies.

References

[1] A. Pnueli, "The temporal logic of programs," in *FOCS*. IEEE, 1977, pp. 46–57.

[2] C. Eisner and D. Fisman, *A practical introduction to PSL*, ser. Series on integrated circuits and systems. Springer, 2006.

[3] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Property specification patterns for finite-state verification," in *FMSP*, M. A. Ardis and J. M. Atlee, Eds. ACM, 1998, pp. 7–15.

[4] A. Bauer and M. Leucker, "The theory and practice of SALT," in *NASA Formal Methods*, ser. LNCS, M. G. Bobaru, K. Havelund, G. J. Holzmann, and R. Joshi, Eds., vol. 6617. Springer, 2011, pp. 13–40.

[5] P. Wolper, "Temporal logic can be more expressive," *Information and Control*, vol. 56, no. 1/2, pp. 72–99, Jan./Feb. 1983.

[6] M. Y. Vardi, "A temporal fixpoint calculus," in *POPL*, 1988, pp. 250–259.

[7] J. G. Henriksen and P. S. Thiagarajan, "Dynamic linear time temporal logic," *Ann. Pure Appl. Logic*, vol. 96, no. 1-3, pp. 187–207, 1999.

[8] M. Leucker and C. Sánchez, "Regular linear temporal logic," in *ICTAC 2007*, ser. LNCS, C. B. Jones, Z. Liu, and J. Woodcock, Eds., vol. 4711. Springer, 2007, pp. 291–305.

[9] S. Demri, "LTL over integer periodicity constraints," *Theor. Comput. Sci.*, vol. 360, no. 1-3, pp. 96–123, 2006.

[10] C. Courcoubetis and M. Yannakakis, "Verifying temporal properties of finite-state probabilistic programs," in *FOCS*. IEEE Computer Society, 1988, pp. 338–345.

[11] U. Sammapun, I. Lee, O. Sokolsky, and J. Regehr, "Statistical runtime checking of probabilistic properties," in *RV*, ser. LNCS, O. Sokolsky and S. Tasiran, Eds., vol. 4839. Springer, 2007, pp. 164–175.

[12] J. Hoenicke, R. Meyer, and E.-R. Olderog, "Kleene, Rabin, and Scott are available," in *CONCUR*, ser. LNCS, P. Gastin and F. Laroussinie, Eds., vol. 6269. Springer, 2010, pp. 462–477.

[13] F. Laroussinie, A. Meyer, and E. Petonnet, "Counting LTL," in *TIME*, N. Markey and J. Wijsen, Eds. IEEE Computer Society, 2010, pp. 51–58.

[14] M. L. Minsky, *Computation: finite and infinite machines*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1967.

[15] A. Schrijver, *Theory of linear and integer programming*, ser. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999.

[16] O. Lichtenstein and A. Pnueli, "Checking that finite state concurrent programs satisfy their linear specification," in *POPL*. New York: ACM, Jan. 1985, pp. 97–107.