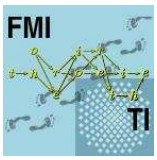
Institut für Formale Methoden
der Informatik
Abteilung Theoretische Informatik
Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgarte

Institut für Softwaretechnik
und Programmiersprachen
Universität zu Lübeck
Ratzeburger Allee 160
D-23562 Lübeck



UNIVERSITÄT ZU LÜBECK
INSTITUTE OF SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Diplomarbeit Nr. 3125

Temporal Logic for Properties with Relative Frequency

Normann Decker

Course of Study: Computer Science

Examiner: Prof. Volker Diekert

Supervisor: Prof. Martin Leucker

Commenced: 1 January 2011

Completed: 1 July 2011

CR-Classification: F.4.1, D.2.4, F.3.1

Abstract. Inherently unreliable or fault-tolerant systems demand for a specification formalism that allows the user to express a required ratio of certain observations. Such a requirement can be, e.g. that deadlines in a real-time system must be met in at least 80% of all cases. Logics and in particular temporal logics provide powerful, flexible and well established specification formalisms. We therefore propose $fLTL$, an extension to linear-time temporal logic that allows for expressing relative frequencies by an intuitive generalization of the temporal operators. We develop a game theoretical methodology and a semantics for temporal logic with counters. For our novel logic, we establish an undecidability result regarding the satisfiability problem but identify a decidable fragment which strictly increases the expressiveness of linear-time temporal logic by allowing, e.g., to express non-context-free properties.

Keywords. Specification and Verification, Temporal Logic, Relative Frequency, Availability, Counter Logic

Contents

1 Introduction

Formal specification, modelling and verification of software systems relies on concepts that allow the user, for example programmer, customer, certifier or authority, to precisely formulate requirements, assumptions or behaviour. It is often convenient, and might even be straight forward, to detail out a system's architecture, to explain behaviour, in natural language. Unfortunately, it is nearly impossible to avoid ambiguity, to ensure that texts accurately reflect the authors' intention. Secondly, a specification should allow for, or even encourage, *automated* reasoning about a system with regard to that specification on one hand, and analysing the specification itself on the other, for example in terms of consistency or completeness.

Logics provide powerful and flexible languages to formulate properties of software systems, processes, algorithms or models of any kind. Their fragments and dialects address various areas of application and provide problem specific and intuitive means of articulation. First or second order logic (FO, MSO), for example is a very general framework that is rather unbiased subject to the domain or the types of statements. That allows the intuitive application in many settings. Temporal logics are suitable to express relations between events, actions or observations in an ordered, time-like domain.

Based on the relationship between theories of logics and e.g. formal languages, automata or games, a variety of methods has been developed. Methods, that help developers to verify and analyse systems regarding to requirements and therefore to produce more reliable software.

One particular domain of specification concerns so called soft requirements. In contrast to strict requirements, that stipulate explicitly a behaviour, a timely relationship of events or the effects of an action, soft requirements are allowed to be violated by the system to a certain extend. This can reflect for example properties for systems that are inherently unreliable or be just more convenient since formulating requirements and considering a less strict interpretation may be more intuitive and less complex than an explicit specification. In a real-time application, it might be required that all processes deliver results within a certain interval. While this is a strict demand, the intention of the developer might actually be a soft deadline that can be missed up to a certain number of times per interval. Formulating explicitly, that a streaming protocol can allow a certain percentage of transmission errors might be complex in contrast to just stating the strict property, that all transmissions must be correct and then adding that this is required to hold only in "most" cases. Still, any formalism that can express such relaxed properties must have a formal, clearly defined semantics.

The aim of the work reported on here was to develop and study a logical specification formalism, for properties allows for relaxing obligations in terms of a relative frequencies.

Extending a logical framework by some intuitive operation can provide more convenient means of specifying properties and encourage developers to make use of them, leading to better quality of software. However, the structure of a logical framework might be very sensitive to small changes and the effect of any extension to the framework can demand for extensive study.

We lay the focus on linear-time temporal logic (*LTL*) which was suggested for reasoning about computer programs first by Armir Pnueli in [Pnu77]. Since, it has been extensively studied and proved to be a suitable formalism for formal specification and verification of software.

Our approach starts by extending the until operator U from *LTL* by annotating a constant frequency c to it. In *LTL*, the until operator connects two properties, represented by sentences φ and ψ. A formula $\varphi \,\mathrm{U}\, \psi$ expresses an *eventuality* in that the property specified by ψ must hold eventually, i.e. at some point in the "future". Additionally, the formula imposes an *obligation* in terms of φ. This obligation must hold at every point in time as long, as the eventuality has not been fulfilled, which *resolves* the formula, and releases the obligation. The obligation φ is thus bound to a scope – from the point of instantiation until the eventuality holds – in which it must always be satisfied. Following the idea of weakening strict requirements, an annotated frequency, i.e. a rational number between zero and one, renders the obligation satisfied already, if at least a fraction c of all positions within the scope satisfy the property φ. We will formally define syntax and semantics of this extension and review the meaning of derived operators in comparison to the original ones. We study in particular how the frequency affects the semantics of the release operator R, which is defined as the dual for U. Apart from the pure syntactical extension we observe that the annotation of frequencies increases expressiveness in terms of language theoretic models.

In order to better understand the formalism and to step further in the direction of application we will investigate the satisfiability problem of the new logic. It turns out to be undecidable in general.

Nevertheless, we examine two approaches, based on games. Lange and Stirling introduce so called *focus games* in [LS01] which can be used to decide the satisfiability problem for *LTL*. One of the players plays for satisfiability and tries to construct a model for the given formula. In order to adopt this approach to $fLTL$ we develop a semantics based on *counters*. This allows us to define a notion of unfolding, which is essential for any approach based on tableaux and formula rewriting. We approximate infinite plays in our extended focus games in order to determine a winner. This requires certain assumptions about the formulae. To overcome some of these restrictions A second approach tries to overcome some of the problems that arise in the first

approach by analysing a folded version of the game graph. We thereby identify a fragment of $fLTL$ that is still more expressive then LTL and for which satisfiability can be decided.

Outline

This report is structured in five chapters. The first chapter introduces logic as specification formalism and motivates the presented work. It contains a short abstract and references related work. In Chapter 2 we give formal foundations and relate language theory and logic. Sections 2.2 and 2.3 provide the crucial basis for the work presented here. The temporal logic LTL is defined and focus games are discussed, providing a game theoretical approach to satisfiability of LTL formulae. The major contribution of this work is detailed out in Chapter 3 and 4. We propose the the extended temporal logic $fLTL$ in Section 3.1 and present first results regarding the relation to LTL and expressivity. Section 3.2 establishes undecidability of $fLTL$ by reduction of Minsky machines. A counter semantics for $fLTL$ is defined in Section 4.1 which provides the basis for extending focus games to $fLTL$ in Section 4.2. Extended focus games enable us to identify an expressive and decidable fragment of $fLTL$ at the end of the fourth chapter. The conclusions in Chapter 5 describe soma alternatives that we did not focus on and provide an outlook and ideas regarding future work.

Related work

In [HMO10], Hoenicke, Meyer and Olderog suggest an extension to regular expressions which they call regular availability expressions. They consider availability of a system as the ratio between time of execution and correct functioning. While they do not consider actual timed models, finite words are used as a discrete representation. For prefixes of a finite word, the availability of some set of letters A is the fraction of observed letters from A and the length of that prefix.

The motivation for availability expressions is related to the idea of frequentness. They express properties that require a system to fulfill some property to a certain extend. Availability expressions contain special symbols, checkmarks, that define a variable scope. The words defined by the expressions contain that symbol and in a second processing step the check-marks are removed and the according availability is checked for the prefix up to the marked position.

Their methodology is based on regular expression and finite automata models, while the approach of $fLTL$ is based on logic and games. Furhtermore, we use ω-words as modles for infinite computation while availability is defined on finite words.

Probabilistic approaches to modeling and analysing systems in a quantitative setting are well studied, e.g. in [dA97], and sophisticated tools are available, such as the

probabilistic model checker PRISM [HKNP06]. In [SLSR07], Sammapun et al. study run-time checking of probabilistic properties.

However, these methods relay on stochastic models, such as Markov chains which we do not consider in the work presented here. While probabilistic methods aim at modelling actual statistic processes, our approach uses the notion of frequency to formulate properties that would require a large explicit specification due to their extensive combinatorics.

2 Logic on Words

Considering the behaviour of systems as chains of discrete observations, events, actions or states, leads to the domain of formal languages. We can make the assumption, that in every time unit, i.e. in every position of such a "chain", one of finitely many observations is made, e.g. a particular action or one of finitely many system states. The observations then yield a finite alphabet for our language theoretical framework, runs of systems are represented as words over this alphabet and behaviours form languages.

Atomic propositions provide an alternative that allow us to abstract from particular observations. Take, for example, debug information in some computer program. An observation, we can make at some point in time, is that a program variable X has a value of 5 and at some other point it might have a value of 4. If the actual value is not of interest but only the assertion that X is greater then zero, we can treat these observations, and many more, identically. If we fix a finite set AP of such properties we can consider the power set of AP our alphabet, distinguishing only observations that differ substantially in the particular setting of verification.

We aim at reasoning about ideally infinite computations which reflect in particular the nature of reactive systems such as web-servers, operating or control systems. Therefore our (idealised) models of computation are infinite, i.e. ω-words.

There are many formalisms to characterize infinite words and the goal is to find a suitable framework for a certain task. In the following we want to introduce words as logical models in general. Logics can provide intuitive languages to formulate properties of words and to specify behaviour.

With logical formulae, a user does not explicitly state how a word must look like but only state properties that are important. Compared to e.g. automata the user does not have to have a strong imagination of an exact behaviour, but only about some aspects, general rules that it must obey. Logic suits in particular the notion of atomic propositions.

2.1 Notation and first-order logic

We start by clarifying some basic notation and introduce the setting of *first-order logic* over words as linear structures.

Variables $n, m, k \ldots$ usually denote natural numbers and we follow the convention that the natural numbers $\mathbb{N} = \{1, 2, \ldots\}$ and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. φ,ψ,Φ,Ψ, $\ldots$ denote formulae. Σ always denotes a finite alphabet, Σ^* the set of finite and Σ^ω the set of infinite words over Σ and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. Letters are mostly denoted a, b, and u, v, w are used for finite or infinite words over Σ. An infinite repetition of some finite word u is denoted u^ω. For a word $w = a_0 a_1 a_2 \ldots$ we write $|w|$ for the number of letters in w, $|w| = \omega$ if w is infinite. For prefixes of w we write $w^n = a_n a_{n+1} \ldots$, for suffixes $w_{(n)} = a_0 \ldots a_n$, thus $w^0 = w_{(|w|+1)} = w$. For sets $U \subseteq \Sigma^*$, $V \subseteq \Sigma^\infty$ of words we define the concatenation $UV = \{uv \mid u \in U, v \in V\}$. 2^M is the power set of some set M.

The standard Boolean connectives $\wedge$, $\vee$ and $\neg$ are used. For a set M of formulae we may write short $\bigwedge M$ or $\bigvee M$ for

$$\bigwedge_{m \in M} m, \quad \text{or} \quad \bigvee_{m \in M} m,$$

respectively. With first-order quantifiers $\forall$ and $\exists$ we bind variables $x, y, \ldots$ from a set V ranging over some universe U.

For formulae in first-order logic we use the elements of AP, interpreted as unary predicates $p(x)$, $q(x)$, $\ldots$, and we assume a single additional binary predicate $\leq$, a *linear order* on U. Symbols $\bot$ and $\top$ are the logical constants *true* and *false*, respectively. The syntax of first-order formulae is defined inductively by

$$\varphi ::= \top \mid \bot \mid x \leq x \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \forall x \varphi \mid \exists x \varphi \mid p(x) \quad (p \in AP)$$

We set brackets to indicate the exact order of evaluation if necessary. We may also use the usual syntactic abbreviations such as $\rightarrow$, $<$ or $=$. For example we have $\forall x p(x)$, $\exists x \forall y (x < y \rightarrow p(y))$ or $\neg\forall x (p(x) \vee q(x))$. Universal and existential quantification of variables refers to values from U. Free variables, are those which are not bound by any quantifier.

In our setting of formal languages we interpret formulae in terms of structures (w, σ), where $w \in \Sigma^\omega$ is an infinite word consisting of letters from Σ. The mapping $\sigma : V \rightarrow U$ interprets free variables.

A word $w = a_0 a_1 a_2 \ldots$ with $a_i \in \Sigma = 2^{AP}$ defines a logical model by taking the positions in w as universe $U = \{i \in \mathbb{N}_0 \mid i < |w|\}$. We interpret a predicate $p(x)$ as the set of positions that carry a letter including p: $p(x) = \{i \in U \mid p \in a_i\}$, thus the semantics of some predicate is given by

$$(w, \sigma) \models p(x) \quad \text{iff} \quad p \in a_{\sigma(x)} \quad (p \in AP,\ x \in V).$$

We naturally also have a linear order on the positions that allows us to interpret the binary predicate $\leq$ accordingly.

To first-order sentences with at most one free variable x, we give a semantics in terms of languages in Σ^ω by

$$\mathcal{L}(\varphi) = \{w \in \Sigma^\omega \mid (w, x \mapsto 0) \models \varphi\}.$$

By *FO*, we denote the here defined set of first-order formulae with linear order over the alphabet Σ or the languages definable by such formulae. One may write $FO_\Sigma[\leq]$ to specify the allowed predicates explicitly – unary ones by the alphabet $\Sigma = 2^{AP}$ and the one binary relation $\leq$. Since we do not vary this setting these annotations are omitted.

We can use *FO* as specification formalism to describe properties or behaviour of some system but we will concentrate on an alternative, namely linear-time temporal logic, that we introduce next. First-order logic needs to be mentioned, however, since it forms an important basis of formal reasoning. Recalling the interpretation of this "standard" logic in the context of languages is certainly beneficial for the understanding of logics in general and the following forthcoming considerations in particular.

A rather algebraic methodology for characterizing first-order definable languages is given by Volker Diekert and Paul Gastin in [DG08]. For a broad perspective on the relationship between a variety of specification formalisms such as logic, automata or games, and language theory the reader may also be referred to the survey "Languages, Automata and Logic" by Wolfgang Thomas [Tho97] and the handbook "Automata, Logics and Infinite Games" [GTW02].

2.2 Linear-time temporal logic

First-order logic formulae rely on an explicit handling of positions. The generalization from propositions, which are just true or false, to predicates of positions allows for evaluating propositions depending on the context (the position in a word). The argument for choosing logic to formulate properties of words was the intuitive handling by the user. Now it appears that explicit variables still pose obstacles to natural intuition. While a purely propositional formula is easy to formulate and to understand it can not relate arbitrarily many positions to each other. An alternative to explicitly attach a position to a proposition comes from modal logic. Modals allow, in a sense, to switch the "setting". Leibniz' formulation of several "possible worlds"[1] often serves as intuition. Propositions are still only true or false but they depend on the current "world". In our setting these worlds are the positions of a word, but in contrast to predicates we do not touch them directly. Instead we say that something happens "possibly" or "necessarily". In the formal framework for modal logic introduced by Kripke [Kri59], connections between worlds play an important role as they define what is "possible", namely what is possible to reach from the current world. Considering word models, the position are linked by a linear order. Therefore, if it is "possible" to reach a world having a certain property this will happen *eventually*. On a one-dimensional chain of positions, we can not go around any of them while moving on. Positions in words are also considered as points in time on a discrete scale, they correspond to temporal

[1] Gottfried Wilhelm Leibniz published in 1710 his "Essays on the Goodness of God, the Freedom of Man and the Origin of Evil" (original French title: "Essais de théodicée sur la bonté de Dieu, la liberté de l'homme, et l'origine du mal") in which he introduced the idea of different possible worlds.

sequences of worlds and so Prior suggested the *temporal interpretation* of modal operators [Pri57, Pri67] in the setting of such linear structures. The association with time led to additional operators such as *next* and *until*. Consequently also corresponding past operators were introduced.

> "Temporal Logic introduces a clear distinction between variability within a state, which is described using classic connectives and quantifiers, and the variability over time, moving from one state to another." [GPSS80]

However, in Linear-time Temporal Logic (*LTL*) which we consider here, no classical quantifiers will be used. The application of temporal logic in computer science was proposed by Burstall [Bur74] and Pnueli used in [Pnu77] temporal logic to reason about liveness properties of concurrent programs which is regarded the starting point of using temporal logic in program specification and verification.

Definition 2.1 (Linear-time temporal logic)**.** Let AP be a finite set of atomic propositions and $\Sigma = 2^{AP}$ a finite alphabet. The syntax of Linear-time Temporal Logic (*LTL*) formulae is given by

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

Let φ be such an *LTL* formula and $w \in \Sigma^\omega$ an infinite word. The semantics of φ is defined in terms of the relation $\models$ as follows:

$$\begin{array}{lll}
w \models \top & & \\
w \models p & \text{iff} & p \in w_0 \quad (p \in AP) \\
w \models \neg\varphi & \text{iff} & w \not\models \varphi \\
w \models \mathrm{X}\,\varphi & \text{iff} & w^1 \models \varphi \\
w \models \varphi \wedge \psi & \text{iff} & w \models \varphi \text{ and } w \models \psi \\
w \models \varphi\,\mathrm{U}\,\psi & \text{iff} & \exists_n : w^n \models \psi \text{ and} \\
 & & \forall_{i<n} : w^i \models \varphi
\end{array}$$

For *LTL* formulae φ, ψ, the following abbreviations may also be used. We let $\overline{AP}$ be the set of negated atomic propositions.

$$\begin{array}{rll}
\bot := \neg\top & & \\
\overline{p} := \neg p & & (p \in AP) \\
\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi) & & \\
\varphi\,\mathrm{R}\,\psi := \neg(\neg\varphi\,\mathrm{U}\,\neg\psi) & & (\text{“release”}) \\
\mathrm{F}\,\varphi := \top\,\mathrm{U}\,\varphi & & (\text{“eventually“}) \\
\mathrm{G}\,\varphi := \neg\,\mathrm{F}\,\neg\varphi & & (\text{“globally”})
\end{array}$$

The set of models of a formula is $\mathcal{L}(\varphi) = \{w \in \Sigma^\omega \mid w \models \varphi\}$.

We may use propositions $p \in AP$ as letters in words. Those refer to the singleton letters $\{p\} \in \Sigma = 2^{AP}$.

Example ([Wol85])**.** Consider the following examples for *LTL* formulae:

p is satisfied by all words in which p is true at the first position. Any continuation is then possible.

$\mathrm{G}(p \to \mathrm{X}\, q)$ states, whenever p holds in a position, then q must hold in the subsequent one.

$\mathrm{G}(p \to \mathrm{X}(\neg q\, \mathrm{U}\, r))$ is satisfied by words where if p is true at some position then q must not hold from the subsequent position on, until the next position in which r holds. Furthermore, by means of eventuality of the until operator, it must happen that r holds at some point. If r holds already in the first position, i.e. directly after p, the obligation $\neg q$ does not have to hold at all.

$\mathrm{G}\,\mathrm{F}\, p$ states that p must hold always eventually. There is no position, that is not eventually preceded by p or, in other words, p must hold at infinitely many positions.

$\mathrm{G}\, p \wedge \mathrm{F}\, \neg p$ is not satisfiable.

$\mathrm{F}\, p \to (\neg p\, \mathrm{U}\, p)$ is a tautology, i.e. is satisfied by any word. The formula is *valid*.

Remark. Observe that we could also define the release operator R directly as

$$w \models \varphi\, \mathrm{R}\, \psi \quad \text{iff} \quad \begin{array}{l} \forall_n : w^n \models \psi \text{ or} \\ \exists_{i<n} : w^i \models \varphi \end{array}$$

by negating the definition for U. This reveals an intuition for the operator: The formula ψ has to always hold *except* there was once a positions that satisfied φ. Therefor evaluating φ to true at some point *releases* the obligation ψ from the next position on. In contrast to the obligation of an until formula, this ψ does not *need* to be released, there is no eventuality that is necessary to hold at some position. The word p^ω satisfies the $q\, \mathrm{R}\, p$ but not the formulae $p\, \mathrm{U}\, q$.

Based on the abbreviations $\bot$, $\vee$ and R, that semantically represent the dual operators of $\top$, $\wedge$ and U, respectively, we define a *positive normal form* of *LTL* formulae where negation appears only (and even implicitly) in front of atomic propositions.

Definition 2.2 (Positive Normal Form of *LTL* formulae)**.** The syntax of the positive normal form of *LTL* formulae is given by

$$\begin{array}{rcccccccccc} \varphi ::= & \top & | & \varphi \wedge \varphi & | & \varphi\, \mathrm{U}\, \varphi & | & p & | & \mathrm{X}\, \varphi & | \\ & \bot & | & \varphi \vee \varphi & | & \varphi\, \mathrm{R}\, \varphi & | & \overline{p} & & & (p \in \mathrm{AP}) \end{array}$$

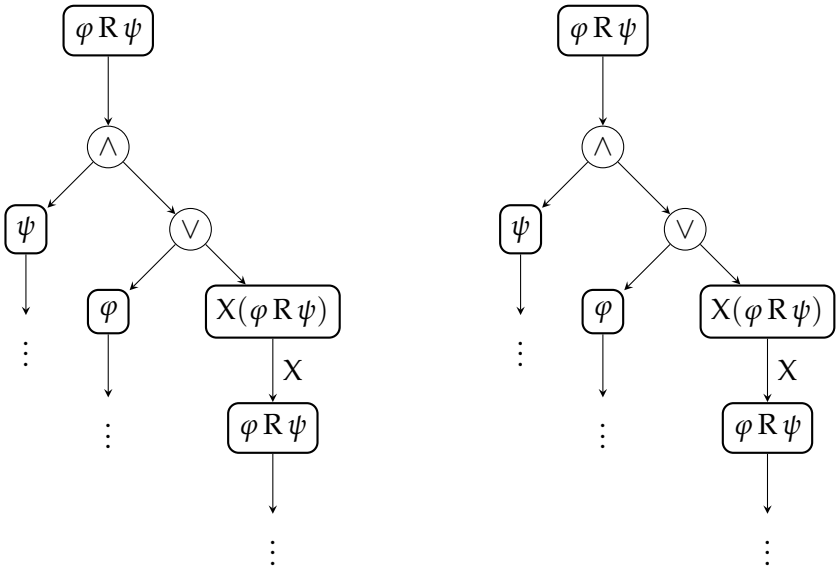


Figure 2.1: Graphical representation of the unfolding of formulae $\varphi \,\mathrm{U}\, \psi$ and $\varphi \,\mathrm{R}\, \psi$.

Note that for all *LTL* formulae φ there exists a formula φ' in positive normal form such that $\mathcal{L}(\varphi) = \mathcal{L}(\varphi')$.

A central equality that follows from the definition of *LTL* is the so called *unfolding* equation

$$w \models \varphi \,\mathrm{U}\, \psi \quad \Leftrightarrow \quad w \models \psi \vee (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}\, \psi)).$$

It allows a syntactical decomposition of *LTL* formulae into one aspect that must hold now at the current position and another that is postponed to the future.

Dually, such an unfolding equation is derived for the release operator by negation:

$$w \models \varphi \,\mathrm{R}\, \psi \quad \Leftrightarrow \quad w \models \psi \wedge (\varphi \vee \mathrm{X}(\varphi \,\mathrm{R}\, \psi)).$$

For the analysis of *LTL* formulae these equations are very valuable. They allow us to draw a *tableau* of the formula, a graphical representation of the timely development described by the formula (see Figure 2.1). In such a graph, a next-formula leads by an edge to a sub-graph describing what happens from the next point in time (i.e. at the next position of a word model). Boolean connectives $\vee$ and $\wedge$ describe a choice and a composition of obligations, respectively. With the unfolding equation an until-formula can be graphically represented in that manner, however, they lead in general to infinite graphs as the unfolding can be repeated arbitrarily often on the same formula.

We define an operator to process formulae based on that equality for a syntactic, yet semantically neutral, unfolding.

Definition 2.3 (Unfolding of *LTL* formulae)**.** For *LTL* formulae Φ, φ and ψ in positive normal form the unfolding $\mathsf{unf}(\Phi)$ is defined as

$$\mathsf{unf}(\Phi) := \begin{cases} \psi \vee (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}\, \psi)) & \text{if } \Phi = \varphi \,\mathrm{U}\, \psi \\ \psi \wedge (\varphi \vee \mathrm{X}(\varphi \,\mathrm{R}\, \psi)) & \text{if } \Phi = \varphi \,\mathrm{R}\, \psi \\ \Phi & \text{otherwise.} \end{cases}$$

Remark. Commonly the notation $LTL_\Sigma[\mathrm{X}, \mathrm{U}]$ is used in order to refer to the formulae following the syntax from Definition 2.1 or the languages over Σ that can be expressed therewith. In this report, the alphabet is usually fixed and is therefore omitted, furthermore we only consider infinite models and thus only ω-languages in Σ^ω. Unless stated otherwise, we assume the two temporal operators X and U, just as defined.

Note however, that e.g. [GPSS80] follows a slightly different notation, namely $LTL[\mathrm{XU}]$, where only a single temporal operator XU is defined as

$$w \models \varphi \,\mathrm{XU}\, \psi \quad \text{iff} \quad \begin{array}{l} \exists_{n>0} : w^n \models \psi \text{ and} \\ \forall_{0<i<n} : w^i \models \varphi \end{array}$$

The difference is that XU basically ignores the first position in a word. $LTL[\mathrm{XU}]$ is still expressively equivalent to $LTL[\mathrm{X}, \mathrm{U}]$ since X and U can be translated as

$$\begin{aligned} \mathrm{X}\,\varphi &= \bot \,\mathrm{XU}\, \varphi \quad \text{and} \\ \varphi \,\mathrm{U}\, \psi &= \psi \vee (\varphi \wedge \varphi \,\mathrm{XU}\, \psi) \end{aligned}$$

and

$$\varphi \,\mathrm{XU}\, \psi = \mathrm{X}(\varphi \,\mathrm{U}\, \psi).$$

We stick to X and U since this equivalence is not given in the more general setting of $fLTL$, which will be introduced in Section 3.1.

The definition of LTL semantics is based on first-order logic. In the definition only two different names for variables were used and with an additional variable indicating the current position, we can thus translate any *LTL* formula into an *FO* formula with at most three names for variables. Hence *LTL* is at most as expressive as the according fragment FO^3 of *FO*. Kamp first established the fact, that it is *exactly* as expressive as *FO* [Kam68]. While Kamp used additional past operators Gabbay et al. sharpened the result in [GPSS80] to the pure future fragment of *LTL*, that is considered here. Also their result is more general as it allows for arbitrary linearly ordered structures, in particular not isomorphic to the natural numbers.

Regarding the class of languages, expressible in *LTL*, there are more characterizations including star-free expressions, various automata models and an algebraic characterization based on aperiodic monoids. In [DG08], Diekert and Gastin survey translations between these theories.

2.3 Satisfiability of LTL formulae

An important issue about any formalism used to describe formal languages is emptiness. Methods checking a language for emptiness provide an insight and are furthermore crucial for practical applications such as consistency checking of specification or especially model checking.

When specified by logical formulae, a language is none-empty if and only if the formula is *satisfiable*, i.e. if there is a model, a word in our setting, under which the formula evaluates to true. For LTL formulae, there exist several approaches to satisfiability checking, common ones are based on tableaux and automata construction.

Tableaux, as e.g. described for first-order logic in [Smu68], syntactically decompose logical formulae. They provide a graphical depiction of semantic relations by a syntactical decomposition. An example is the use of the unfolding for the until and release operator as shown in Figure 2.1. Many approaches to analysing formulae are based on this principle. A method for satisfiability checking of LTL formulae is described in [Wol85]. One advantage is usually, that the graph is constructed during the analysis and needs not to be stored completely which is exploited, e.g. in the method described in [GPVW95].

Related to the graphs of such tableaux are automata constructions, translations of formulae to automata recognizing exactly according models. While classical tableaux yield a result themselves, as they represent a calculus, e.g. for satisfiability, the automata approach interprets the tableau graph as an automaton model which is then analysed with appropriate methods. There is a variety of automata constructions, the classical approach is due to Vardi and Wolper [VW86, VW94]. A widely recognized optimization by the use of alternating and generalized Büchi automata is due to Gastin and Oddoux [GO01]. Again, the concepts are technically related, however, the different intuitional interpretation may provide different inspiration and motivate other techniques.

A third notion is the interpretation in terms of games. While being technically also very similar to tableaux constructions and automata, the notion of two players provides a natural perspective. This helps to investigate and understand the nature of problems, in particular for combinatorially challenging analysis, such as reasoning about infinite words.

In none-deterministic and deterministic automata models the transition relation is much easier to overlook while alternation often provides a much more concise representation. Arguing about strategies of players is often more intuitive and convenient than arguing about some graph properties directly because strategies can be none-positional, yet still intuitive. Equivalent automata models are preferably none-deterministic, or even deterministic, which involves an exponential blow up and are therefore often harder to understand and to reason about.

2.3.1 Focus games

In [LS01], Lange and Stirling suggest so called *focus games* for satisfiability checking of LTL formulae. We discuss that approach in the following and extend it in Section 4.2 to analyse our extended logic $fLTL$.

Solving the question of satisfiability for a formula reduces to either provide a satisfying model or to show that no such model exists. Intuitively, we can consider two players, let us call them E and A, that argue about the existence of such a model. Player E claims, that she can build such a model and Player A tries to prove that the model provided violates the formula. Now if E comes up with a potential model and we are convinced, that both players are smart enough, we can conclude that if A can not disprove it, the formula is actually satisfied and, on the other hand, if he did pick a false model then only for the reason a true one does not exist.

A game consists of a set of *configurations* that are owned by exactly one of the players and *moves* that turn one configuration into another. We thus, have a graph where the nodes are configurations and directed edges defined by the moves. Furthermore there must be a winning condition that determines a winner for a *play*, i.e. a path in the game graph. Players can choose a move according to a *strategy* that determines for the current situation which move to make.

Focus games are based on the following idea: Take a set of positive formulae as obligations. Assume Player E is our side in favour, we try to construct a word letter by letter which satisfies all of them. Now if one of the formulae is an atomic proposition, we must choose a letter which contains that proposition and, on the contrary, if we face a negated proposition we must not chose a letter containing it. In particular, if we have obligations p and $\neg p$ for any $p \in AP$, we are lost and can not construct any model satisfying both at a time. We need to avoid that at any cost. A conjunction $\varphi \wedge \psi$ can be broken up, adding φ and ψ to the set of obligations since we need to satisfy both of them. A disjunction actually allows us to make a choice, namely of which part we believe is easier to satisfy. Thus for each formula $\varphi \vee \psi$ we can discard one side and keep the other. What is left are temporally quantified formulae. A X-formula does not directly impose any constraint on the letter we can choose now so we leave them untouched until we examined all obligations for the current position of the word. The local influence of U- and R-formulae is characterized by their unfolding: For an U-formula we have the chance to finally "fulfill" it by choosing the rear part, but only if that does not lead to two contradicting propositions in the set. For the release, we have a similar choice. The only difference is that we will have to fulfill any U eventually while R-formulae can be postponed forever. Once the set of obligation consists only of (none contradicting) atomic propositions and X-formulae, we choose the subset of all positive propositions as letter for the current position in the word and proceed by keeping only formulae φ for which we still have an obligation $\mathrm{X}\,\varphi$.

Since conjunctions do not require a choice Player A appears to have nothing to do at all but to point out if we added contradicting propositions. Yet, there is one more

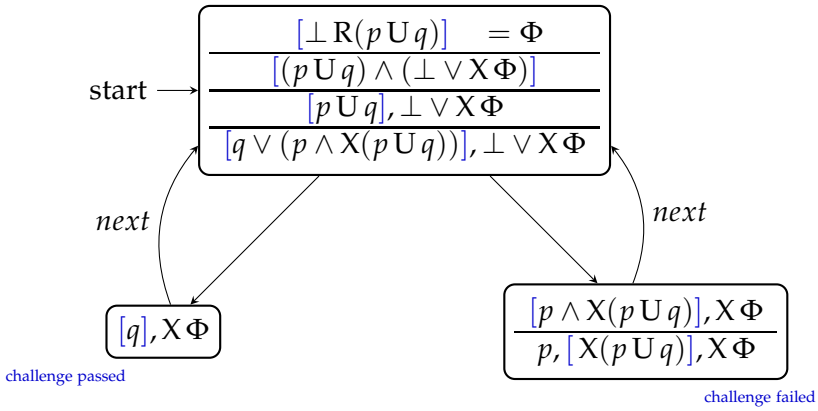


Figure 2.2: Tableaux graph on the formula $\bot \mathrm{R}(p \mathrm{U} q)$. A focus distinguishes greatest and least fixed points.

issue to keep track of. Consider the formula

$$\Phi = \bot \mathrm{R}(p \mathrm{U} q) = (p \mathrm{U} q) \wedge (\bot \vee \mathrm{X}\Phi)$$

as the obligation in the set $\{\Phi\}$. By breaking up the $\wedge$-operator we get $\{p \mathrm{U} q, \bot \vee \mathrm{X}\varphi\}$. For the second formula we have no actual choice since $\bot$ is unsatisfiable. With the first formula unfolded we have thus $\{q \vee (p \wedge \mathrm{X}(p \mathrm{U} q)), \mathrm{X}\varphi\}$ that gives us the first choice, namely either to discard q or $(p \wedge \mathrm{X}(p \mathrm{U} q)$. Even though there is no reason not to keep q in this case, let us consider both options as shown in Figure 2.2 and see what makes the difference in terms of constructing a model, i.e. winning the game. Taking the left loop infinitely often, Player E builds a word $\{q\}^\omega$, which is a model for the formula. Choosing the right loop does not violate the formula locally at any time but the word constructed would be $\{p\}^\omega$ which is not a model because the obligation $p \mathrm{U} q$ was never fulfilled but postponed forever.

Giving Player A the possibility to point out obligations which are illegally postponed forever is done by introducing a so called *focus*. Player A can focus on an U-formula, indicating that he does not believe that this particular sub-formula will eventually be satisfied. In case of conjunctions that carry the focus, Player A chooses the one that inherits it. Now Player E has not only to avoid local contradictions but also prove, that the formula in focus can be fulfilled. Once that is proven A can challenge E again by moving the focus, but only to a formula that has not had the focus before.

With these considerations we can formalize moves and configurations in more detail.

Definition 2.4 (Sub-formulae)**.** For a formula Φ in positive normal form the set $\mathsf{sub}(\Phi)$

of unfolded sub-formulae is defined inductively by

$$\begin{array}{lcll}
\mathsf{sub}(p) & := & \{p\} & (p \in AP \cup \overline{AP}) \\
\mathsf{sub}(\varphi \wedge \psi) & := & \{\varphi \wedge \psi\} \cup \mathsf{sub}(\varphi) \cup \mathsf{sub}(\psi) & \\
\mathsf{sub}(\varphi \vee \psi) & := & \{\varphi \vee \psi\} \cup \mathsf{sub}(\varphi) \cup \mathsf{sub}(\psi) & \\
\mathsf{sub}(\mathrm{X}\,\varphi) & := & \{\mathrm{X}\,\varphi\} \cup \mathsf{sub}(\varphi) & \\
\mathsf{sub}(\varphi\,\mathrm{U}\,\psi) & := & \{\mathsf{unf}(\varphi\,\mathrm{U}\,\psi), \varphi \wedge \mathrm{X}(\varphi\,\mathrm{U}\,\psi), \mathrm{X}(\varphi\,\mathrm{U}\,\psi)\} & \\
 & & \cup\, \mathsf{sub}(\varphi) \cup \mathsf{sub}(\psi) & \\
\mathsf{sub}(\varphi\,\mathrm{R}\,\psi) & := & \{\mathsf{unf}(\varphi\,\mathrm{R}\,\psi), \varphi \vee \mathrm{X}(\varphi\,\mathrm{R}\,\psi), \mathrm{X}(\varphi\,\mathrm{R}\,\psi)\} & \\
 & & \cup\, \mathsf{sub}(\varphi) \cup \mathsf{sub}(\psi) &
\end{array}$$

Additionally, we always have $\top, \bot \in \mathsf{sub}(\Phi)$.

Definition 2.5 (Focus Games [LS01])**.** Let Φ be an *LTL* formula in positive normal form. The *focus game* $\mathcal{G}(\Phi)$ is a graph (V, E) where $V \subseteq \mathsf{sub}(\varphi) \times 2^{\mathsf{sub}(\Phi)}$ is a set of *game configurations* and $E \subseteq V \times V$ is a transition relation. For any configuration $C = ([\varphi], \Gamma)$ it is required that $\varphi \in \Gamma$. φ indicates the formula that carries the focus which we indicate by the surrounding brackets. If Φ is not already unfolded, we set $\mathcal{G}(\Phi) := \mathcal{G}(\mathsf{unf}(\Phi))$.

We identify configurations if they are equal after application of the normalization rules from Figure 2.3. Two configurations c_1, c_2 are connected by a dircted edge, i.e. $(c_1, c_2) \in E$ iff one of the *moves* from Figure 2.3 can be applied to c_1 and results in c_2. Normalization rules and moves are read from top to bottom. Note, that the order of application of moves or rules does not matter. The most important rule considering the constructed word is the next rule. It discards all atomic propositions and exactly those are required to hold in the current position. With application of the next rule we move one step further in the construction of the word but that is possible if and only if no other move or rule can be applied anymore. To make the game graph syntactically explicit we can assume an arbitrary order, that ensures that all rules that can be applied will be applied eventually. For visualizing games we might combine moves whenever convenient.

In a configuration $C = ([\mathrm{X}\,\varphi_1], \{\mathrm{X}\,\varphi_1, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\})$ a winner may have become apparent. If there is a propositional contradiction $q_i = \overline{q_j}$, player E obviously did not succeed in constructing a model for the initial formula and thus the game is lost for her. On the other hand, if $n = 0$ all requirements have been met and the game is won by E. There is no rule that applies to an empty configuration, anyway. Furthermore the configuration may have occurred already. In that case the game can go on forever, repeating that loop ad infinitum. By means of the focus a winner can nevertheless be anticipated: Such an infinite sequence depends on the repeated unfolding of an until or release formula. Assuming there was no contradiction so far in the play, the only reason why E should not go on like this forever, building a model that way is that there is an eventuality that needs to be fulfilled. That is some until formula was repeatedly postponed to "fulfill" and therefore stays in the configuration.

Player E

$$\frac{[\varphi_0 \vee \varphi_1], \{\varphi_0 \vee \varphi_1\} \cup \Gamma}{[\mathsf{unf}(\varphi_i)], \{\mathsf{unf}(\varphi_i)\} \cup \Gamma} \qquad \frac{[\psi], \{\varphi_0 \vee \varphi_1\} \cup \Gamma}{[\psi], \{\mathsf{unf}(\varphi_i)\} \cup \Gamma}$$

Player A

$$\frac{[\varphi_0 \wedge \varphi_1], \{\varphi_0 \wedge \varphi_1\} \cup \Gamma}{[\mathsf{unf}(\varphi_i)], \{\mathsf{unf}(\varphi_{1-i})\} \cup \Gamma} \qquad \frac{[\varphi], \{\varphi, \psi\} \cup \Gamma}{[\psi], \{\varphi, \psi\} \cup \Gamma} \text{ (change)}$$

Normalization rules

$$\frac{[\psi], \{\varphi_0 \wedge \varphi_1\} \cup \Gamma}{[\psi], \{\mathsf{unf}(\varphi_0), \mathsf{unf}(\varphi_1)\} \cup \Gamma} \qquad \frac{[\mathrm{X}\,\varphi_1], \{\mathrm{X}\,\varphi_1, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}}{[\mathsf{unf}(\varphi_1)], \{\mathsf{unf}(\varphi_1), \ldots, \mathsf{unf}(\varphi_n)\}} \text{ (next)}$$

Figure 2.3: Game moves and normalization rules for focus games. Rules are applied after each game move to cover obligatory transformations. φ and ψ denote arbitrary formulae, q denotes possibly negated propositions or $\top$. Note, once the formula $\bot$ occurred, the next-rule can *not* be applied anymore. Adopted from [LS01].

If that is the case A is able to point out that formula by putting it in focus and has no obligation to change the focus afterwards. Only if A can not focus on such a formula, E can force A to change the focus when resolving the respective until. Therefore A has lost if C is repeated and the focus moved in between or A could only avoid to change by setting a release formula in focus. In the latter case E's strategy is legal since the release formula allows to be postponed forever.

Definition 2.6 (Plays and winning conditions)**.** A *play* on a game $\mathcal{G}(\Phi)$ is a sequence of configurations $P = C_0C_1\ldots$ such that $C_0 = ([\Phi], \{\Phi\})$ and $(C_i, C_{i+1}) \in E$. A play *ends* at the first configuration $C_n = ([\Psi], \{\mathrm{X}\,\varphi_1, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}$ with $\Psi = \mathrm{X}\,\varphi$ or $\Psi = q_i \in AP$, such that one of the players wins $P = C_0C_1\ldots C_n$ according to the following *winning conditions*.

The play P is won by Player E if and only if

- $C_n = ([q_1], \{q_1, q_2, \ldots, q_m\})$ where q_i is either the constant $\top$ or a (possibly negated) proposition and $p_i \neq \neg p_j$ for $i, j = 1\ldots m$, or
- $C_n = C_i = ([\mathrm{X}(\varphi\,\mathrm{R}\,\psi)], \Gamma)$ for some $i < n$ or
- $C_n = C_i$ for some $i < n$ and Player A has applied the change-rule between C_i and C_n.

Player A wins the play P if and only if

- $C_n = ([\Psi], \Gamma)$ and $\bot \in \Gamma$ or $p, \overline{p} \in \Gamma$ for some $p \in AP$, or
- $C_n = C_i = ([\mathrm{X}(\varphi \,\mathrm{U}\, \psi)], \Gamma)$ for some $i < n$ and between C_i and C_n the change-rule was not applied.

Players can follow a certain strategy, that indicates which move to make after a partial play $C_0 C_1 \dots C_k$. Strategies may depend on the complete sequence of configurations played so far.

Definition 2.7 (Strategies)**.** A *strategy* for a Player X on a game $\mathcal{G}(\varphi) = (V, E)$ is a partial mapping $\pi : V^* \to V$, that obeys the game rules, i.e. $C_1 C_2 \dots C_n \mapsto C_{n+1}$ only if $(C_n, C_{n+1}) \in E$.

A Player X uses a strategy π in a play $P = C_1 C_2 \dots$ if and only if for all configurations C_i in P, such that X is to make a move in C_i, the choice is $C_{i+1} = \pi(C_1 C_2 \dots C_i)$.

A strategy π is *winning* for a game $\mathcal{G}(\varphi)$ iff all plays, for which X uses π, are won by X and the game is said to be won by X if there is a winning strategy for X.

Remark. Plays end after the first recurrence of a configuration. Since the state space $V \subseteq \mathsf{sub}(\Phi) \times 2^{\mathsf{sub}(\Phi)}$ is finite, every play has finite length. Also, the winning conditions are mutually exclusive and one of the conditions must apply. A none-repeating play can only end with a purely propositional configuration and if a configuration is repeated without the focus being changed in between, the formula in focus must either be a release or an until formula. The shape of such a recurring formula is sooner or later $\mathrm{X}(\varphi \,\mathrm{U}\, \psi)$ or $\mathrm{X}(\varphi \,\mathrm{R}\, \psi)$, respectively. The winner of any play in a focus game $\mathcal{G}(\Phi)$ is thus uniquely determined and therefore the winner of the game itself.

Theorem 2.1 (Soundness and completeness [LS01])**.** *Player E has a winning strategy for a focus game $\mathcal{G}(\Phi)$ on an LTL formula Φ if and only if Φ is satisfiable.*

Proof. In the following we describe a strategy for A and show that if E can win against this strategy, there is a model for Ψ. We therefore call this strategy *optimal* for A.

Player A maintains a priority list of all until sub-formulae of Φ. That list will ensure, that A focuses on any until formula a second time only if he went through all other possible until formulae before. By re-focusing an until formula, A risks to revisit a configuration while having changed the focus in between.

Whenever A has to choose where to put the focus, i.e. in a situation $([\varphi \wedge \psi], \Gamma)$ he focuses on

- ψ if it contains an until sub-formula or if φ does not.
- He focuses on φ if it contains an until sub-formula while ψ does not.

Whenever E resolves an until formula in focus, i.e. she chooses ψ in a configuration $([\psi \vee (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}\, \psi))], \Gamma)$, A moves that until formula to the end of the list and focuses on the first one in the list, that is present in Γ. By present we mean that $\mathsf{unf}(\varphi \,\mathrm{U}\, \psi) \in \Gamma$ or $\mathrm{X}(\varphi \,\mathrm{U}\, \psi) \in \Gamma$.

Player A also changes to the first present until formula in the list if the focus ends up on a proposition or a configuration is repeated while the focus remained on a release formula.

Note that, in particular A keeps the focus on until formulae $\varphi \,\mathrm{U}\, \psi$ as long as possible, i.e. until they are fulfilled by E choosing ψ in the unfolding.

($\Rightarrow$). Assume E has a winning strategy for $\mathcal{G}(\Phi)$ and thus wins every play.

Observe, that by the rules of the game, E refines Φ to an under-approximation by choosing only one side of disjunctions. Before the next rule is applied, Φ is reduced to a conjunction

$$\Phi' = \bigwedge \{\mathrm{X}\,\varphi_1, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}$$

and for any $w \models \Phi'$ we have that $w \models \Phi$.

E wins in particular against the optimal strategy for A and the according play $P = C_1 \ldots C_n$ yields a model for Φ:

Every time, the next rule is applied during the play P at a configuration $C_i = ([\xi_i], \Gamma_i)$, we obtain a none-contradicting set of atomic propositions $\{q_1, q_2, \ldots, q_m\}$ since E wins. Hence, restricted to AP, these sets are letters in $\Sigma = 2^{AP}$ and the play yields a word

$$\begin{array}{cccccccccc} C_1 & \ldots & C_{i_1} & \to^{(next)} C_{i_1+1} & \ldots & C_{i_2} & \to^{(next)} C_{i_2+1} & \ldots & C_{i_3} & \to^{(next)} C_{i_3+1} \quad \ldots \\ & & \downarrow & & & \downarrow & & & \downarrow & \\ w := & & a_1 & & & a_2 & & & a_3 & \quad \ldots \end{array}$$

The negated propositions are included implicitly in letters by the absence of their positive duals.

$w \models \Phi'$ iff $w_{(0)} = a_0 \models \bigwedge(\Gamma_{i_1} \cap AP)$ and $w^1 \models \bigwedge \Gamma_{i_1+1}$. The first condition is true by construction. Additionally, that argument applies in any position in w and thus w will not violate Φ' *positionally*. If we considered only greatest fixed points in Φ', w would be certainly a model. Only if there is an unfulfilled eventuality imposed by an until sub-formula in Φ', the word would not be a model. The play ends after finitely many configurations but we can extend the acquired word: If the play ends in an empty configuration the prefix obtained is good[2] and can be extended arbitrarily. If the play ended by a repetition of a configuration we append that part between the repeating configurations infinitely often.

[2] A *good prefix* for an ω-language is a finite word, such that all extensions to an infinite word belong to the language. [KV01]

By the strategy A used in the play we can reject that case and assure that w is indeed a model for Φ': If in no configuration an until formula is present, E could avoid these obligations to fulfill an eventually by her choice on disjunctions. Otherwise, by the strategy of A, the focus would sooner or later stay on all of them and only be moved away when that particular eventually is fulfilled. Also, using the priority list, A assures to not focus on one of them a second time unless all of them have been fulfilled before. If there were an unfulfilled until, A would focus on it before revisiting a configuration, never change the focus again and win. Since that is not the case, there are no unfulfilled eventualities in w.

($\Leftarrow$). Assume Φ is satisfiable and $w \in \Sigma^\omega$, $w = a_1 a_2 a_3 \ldots$, is a model. We can derive a winning strategy for E on $\mathcal{G}(\Phi)$. Suppose that

$$P = C_{0,0} C_{0,1} C_{0,2} \ldots C_{1,0} C_{1,1} \ldots C_{2,0} \ldots$$

is a play where $C_{i,0}$ is the i-th configuration where the (next)-rule is applied. (If that is possible already in the first configuration we count accordingly $C_{0,0}$=$C_{1,0}$.)

$w \models \Gamma_{0,0}$ and that does not change but through a move from E. However, E will preserve that when choosing a move. That is possible since

$$w \models \left(\bigwedge \Gamma\right) \wedge (\varphi \vee \psi) \quad \text{iff} \quad \begin{array}{l} w \models (\bigwedge \Gamma) \wedge \varphi \text{ or} \\ w \models (\bigwedge \Gamma) \wedge \psi. \end{array}$$

Before the (next)-rule is applied in $C_{1,0}$ we hence have the situation that

$$w^0 \models \bigwedge \Gamma_{1,0} = \left(\bigwedge \{\mathrm{X}\,\varphi_1, \ldots, \mathrm{X}\,\varphi_n\}\right) \wedge \left(\bigwedge \{q_1, \ldots, q_m\}\right).$$

Therefore $\{q_1, \ldots, q_m\}$ is obviously satisfiable (by a_0) and

$$w^1 \models \bigwedge \{\varphi_1, \ldots, \varphi_n\}.$$

Player E, maintaining the invariant $w^i \models \bigwedge \Gamma_{(i,j)}$ at every configuration $C_{i,j}$, will thus not be defeated by an inconsistent set of propositions. Additionally, if E has the choice to fulfill an until formula, she will do so: She will choose ψ in a configuration

$$C_{i,j} = ([\xi], \Gamma \cup \{\psi \vee (\varphi \wedge \mathrm{X}(\varphi\,\mathrm{U}\,\psi))\})$$

where

$$w^i \models \left(\bigwedge \Gamma\right) \wedge (\varphi \wedge \mathrm{X}(\psi\,\mathrm{U}\,\psi)) \quad \text{and} \quad w^i \models \left(\bigwedge \Gamma\right) \wedge \psi.$$

Now assume that $C_{i,0} = ([\mathrm{X}(\varphi\,\mathrm{U}\,\psi)], \Gamma_{i,0}) = C_{i',0}$ for $i' > i$ and Player A has not changed the focus during the play since the first time $C_{i,0}$ was reached. Then we have

$$w^i \models \bigwedge C_{i,0} = \bigwedge C_{i',0} =\!| \; w^{i'}.$$

In that case, there is a strategy that continues according $w^{i'}$ already from configuration $C_{i,0}$ on, still maintaining the invariance. Intuitively, we slice some part of w and simply avoid that one repetition in the play. We do that continuously until we obtain a strategy, such that the until formula is fulfilled at latest at $C_{i',0}$ and thus avoid all potential repetitions but force A to change focus. Note that this refinement of strategies terminates since at every configuration $C_{k,0}$, $k > i$, such that the until formula has not yet been fulfilled by E we have

$$w^k \models \left(\bigwedge \Gamma_{k,0}\right) \wedge \neg\psi$$

because E prefers to choose ψ if possible. But since w^i is a model for $\mathrm{X}(\varphi \,\mathrm{U}\, \psi)$, there *is* a position j such that $w^j \models \psi$.

We do that for all present until formulae and hence obtain a play, where every until formula is released before a configuration is repeated and we conclude that E has a winning strategy for $\mathcal{G}(\Phi)$. □

Deciding satisfiability for a given formula reduces now to guess a strategy for E and let E play with that strategy against the optimal strategy for A. In other words, we traverse all possible plays, knowing they are finite, and search a winning one for E that we can find if and only if the formula is satisfiable.

3 Defining Frequentness

For this section, our aim is to extend *LTL* in an intuitive way in order to specify properties (words, languages) that cover a notion of relative frequency of events. This is meant to reflect the idea of relaxing a property which we believe is a very natural form for descriptions. Whenever we specify an obligation that has to hold for a certain period or at a certain number of positions, e.g. a property like "always p" we can derive a property "sufficiently often p" which is less strict, but should nevertheless be defined exactly.

In the context of infinite computations a definition of relative frequency requires a scope in order to be verifiable. Otherwise a property like "sufficiently often p", that considers the whole infinite word would rather describe some distribution than a discrete property. Even though statistical analysis may allow some reasoning on whether a property holds, we aim at giving the user the option to explicitly specify a scope in which a property must hold and to which extend.

Apart from the notion of relative frequency, the term "often" could mean alternatively a certain constant number of positions but that appears to be a rather inflexible and trivial approach. It can be realized using nested F operators in standard *LTL*. For example a formula $\mathrm{F}(p \wedge \mathrm{X}\,\mathrm{F}(p \wedge \mathrm{X}\,\mathrm{F}\,p))$ would require the proposition p to hold at least three times, somewhere in the word.

A more dynamic approach would be a formula like $\mathrm{G}(p \vee \mathrm{X}\,p \vee \mathrm{X}\,\mathrm{X}\,p)$ which specifies that p has to hold in every frame of length three. This formulation implies a certain frequency of p, however it also implies a certain distribution and is not bound to a scope. A word such as $(\{p\}^4\emptyset^8)^\omega$ would respect the same frequency requirement but violate the formula because the positions for p are not distributed equally enough.

We pursue another option, based on the scope defined by the U operator. For a formula $\varphi\,\mathrm{U}\,\psi$, the scope of φ may end in any position that satisfies ψ and before the end of the scope, φ has to hold always. The until operator directly provides a scope and an obligation. An intuitive relaxation is that the obligation φ is only required to hold in a sufficiently large part of the scope. An example for a property could be the following: A process synchronization must happen at some point and before that at least 80% of all incoming requests must be processed. If this property is violated, a programmer might want a system to allocate additional resources to handle more incoming requests.

3.1 LTL with relative frequencies

Relaxing the semantics of the until operator U appears to be an intuitive and syntactically elegant way in order to introduce frequency into the framework of LTL. The usual intuition for a formula $p \,\mathrm{U}\, q$ is that q must hold at some point in the future, and before that p has to hold *always*. Instead of "always", consider the less strict formulation "sufficiently often". With a relative frequency $c \in [0,1]$ in mind, this is not a vague term since we can refer to an explicit scope. In Definition 2.1, the until operator was given a semantics by

$$w \models \varphi \,\mathrm{U}\, \psi \quad \text{iff} \quad \begin{array}{l} \exists_n : w^n \models \psi \text{ and} \\ \forall_{i<n} : w^i \models \varphi. \end{array}$$

That is equivalent to a formulation based on the *number of positions* before the end of the scope at a position n, instead of a general $\forall$-quantification:

$$w \models \varphi \,\mathrm{U}\, \psi \quad \text{iff} \quad \begin{array}{l} \exists_n : w^n \models \psi \text{ and} \\ |\{0 \leq i < n \mid w^i \models \varphi\}| \geq 1 \cdot n \end{array}$$

Recall that we consider the indices starting with zero, $w = a_0 a_1 \ldots$. Thus there are n letters preceding a_n. For example, consider a formula $p \,\mathrm{U}\, q$ and a word w starting with $\{q\}$, i.e. w^0 satisfies q already. Then the above set of positions before $n = 0$ is empty and $0 \geq 1 \cdot n$. The actual position n is not considered anymore since in the definition we required p only to hold at positions *before* w^n begins, which are exactly n. For a word $w' = \{p\}\{p\}\{p\}\{q\}\ldots$, the eventuality q is satisfied at position $n = 3$. There are 3 letters before that position and therefore p always holds before q only if the number of positions for p is greater then or equal to $1 \cdot n = 3$.

With this reformulation we have a direct way of relaxing the operator, namely by multiplying a factor c, which may be annotated to the U-operator.

Definition 3.1 (Syntax and semantics of $fLTL$)**.** The syntax of Frequency Linear-time Temporal Logic ($fLTL$) formulae is given by

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

where each U-operator is annotated by a rational number $c \in \mathbb{Q}$ with $0 \leq c \leq 1$.

$fLTL$ formulae are interpreted over words $w \in \Sigma^\omega$, $w = a_0 a_1 a_2 \ldots$, with the following semantics.

$$\begin{array}{lll} w \models \top & & \\ w \models p & \text{iff} & p \in a_0 \quad (p \in AP) \\ w \models \neg\varphi & \text{iff} & w \not\models \varphi \\ w \models \mathrm{X}\,\varphi & \text{iff} & w^1 \models \varphi \\ w \models \varphi \wedge \psi & \text{iff} & w \models \varphi \text{ and } w \models \psi \\ w \models \varphi \,\mathrm{U}^c\, \psi & \text{iff} & \exists_n : w^n \models \psi \text{ and} \\ & & |\{i \mid 0 \leq i < n,\ w^i \models \varphi\}| \geq c \cdot n \end{array}$$

Similar to LTL, we consider the following operators as abbreviations.

$$\begin{aligned}
\bot &:= \neg\top \\
\overline{p} &:= \neg p && (p \in AP) \\
\varphi \vee \psi &:= \neg(\neg\varphi \wedge \neg\psi) \\
\varphi \,\mathrm{R}^c\, \psi &:= \neg(\neg\varphi \,\mathrm{U}^{1-c}\, \neg\psi) \\
\varphi \,\mathrm{U}\, \psi &:= \varphi \,\mathrm{U}^1\, \psi \\
\varphi \,\mathrm{R}\, \psi &:= \varphi \,\mathrm{R}^0\, \psi \\
\mathrm{F}\,\varphi &:= \top \,\mathrm{U}\, \varphi \\
\mathrm{G}\,\varphi &:= \neg\,\mathrm{F}\,\neg\varphi
\end{aligned}$$

Example. Consider the formula $p \,\mathrm{U}^{\frac{1}{2}}\, q$ and a the following infinite words.

$p\ p\ p\ r\ r\ r\ q\ r^\omega$ is a model since q holds at position 6 (the 7th letter) and in the prefix before we have three out of six positions that satisfy p.

$r\ r\ q\ r^\omega$ is not a model since the only position at which q holds is the one with index 2. But the two letters before are both not containing p and the ratio is thus $\frac{0}{2} = 0$ which is smaller then the required $\frac{1}{2}$.

$r\ q\ p\ p\ q\ q\ r^\omega$ Here, we have three potential witnesses: positions 1, 4 and 5. The observed frequency of p at positions 1 and 5 is 0 and $\frac{2}{5}$, respectively, which is too low. Yet, at position 4 q is satisfied *and* before position 4 we observe two out of 4 times that p holds. Thus the word is a model for $p \,\mathrm{U}^{\frac{1}{2}}\, q$

Let us examine the duality between U^c and $\mathrm{R}^{(1-c)}$, that is suggested by the above definition, in order to get an intuition for the meaning of "release" in the frequentness setting.

Let $w \in \Sigma^\omega$ and $w \models \varphi \,\mathrm{R}^c\, \psi = \neg(\neg\varphi \,\mathrm{U}^{1-c}\, \neg\psi)$, that is by definition

$$\begin{array}{rlll}
 & \neg\Big(& \exists_n : w^n \models \neg\psi \quad \text{and} & |\{i \mid 0 \le i < n,\ w^i \models \neg\varphi\}| \ge (1-c)\cdot n\Big) \\
\Leftrightarrow & & \forall_n : w^n \models \psi \quad \text{or} & |\{i \mid 0 \le i < n,\ w^i \models \neg\varphi\}| < (1-c)\cdot n.
\end{array} \tag{3.1}$$

If the number of positions satisfying $\neg\varphi$ (i.e. *violating* φ) is *less* than a fraction $(1-c)$ of all positions, then the number of positions *satisfying* φ is *at least* the fraction $1-(1-c) = c$ of all all positions. We hence rewrite Equation 3.1 above to

$$\forall_n : w^n \models \psi \text{ or } |\{i \mid 0 \le i < n,\ w^i \models \varphi\}| > c \cdot n.$$

We quickly confirm that by the following Lemma 3.1. For easier reading, we write for a word $w \in \Sigma^\omega$ and positions i, n in w

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

for the number of position in the prefix $w_0 w_1 \ldots w_{n-1}$ that satisfy an $fLTL$ formula φ.

Lemma 3.1. *Let for a word $w \in \Sigma^\omega$, a position n in w and an fLTL formula φ*

$$\begin{aligned} a &:= \#_{\varphi,w}(n) \quad \textit{and} \\ b &:= \#_{\neg\varphi,w}(n). \end{aligned}$$

By definition, φ and $\neg\varphi$ are mutual exclusive at the single position n while one of them must hold. Hence we have

$$\#_{\varphi,w}(n) + \#_{\neg\varphi,w}(n) = a + b = n \geq 0$$

and confirm for $c \in \mathbb{Q}$, $0 \leq c \leq 1$, that

$$\begin{aligned} & & a &< (1-c)\cdot n \\ \Leftrightarrow & & a+b &< (1-c)\cdot(a+b)+b \\ \Leftrightarrow & a+b-(1-c)\cdot(a+b) &&< b \\ \Leftrightarrow & (1-(1-c))\cdot(a+b) &&< b \\ \Leftrightarrow & c\cdot n &&< b \end{aligned}$$

□

First of all, we see that the explicit interpretation of R^c coincides for $c = 0$ with the classical notion of the release operator and thus justifies abbreviating R^0 by R.

It also reveals that for $\varphi\,\mathrm{R}^c\,\psi$ we can read "sufficiently often φ releases ψ". However, note the particularity: Under this view, the classical notion of φ seen *once* releases ψ *forever* turns out to only be an implication that follows by the "strength" of the classical until. In our setting, which allows "weakening" the until, the release gets much "stronger" in that it imposes an ever recurring obligation in satisfying ψ or φ. While there may be a position in a model that does not need to satisfy ψ because it was preceded by enough positions for φ, this does not "fulfill" the formula and allows an arbitrary suffix thereafter. Consider for example formulae $\Phi = p\,\mathrm{R}\,q$ and $\Phi' = p\,\mathrm{R}^{\frac{1}{4}}\,q$ together with a word starting as follows. (The fraction of positions that satisfy p so far is written underneath.)

$$\begin{array}{cccccccccc} \{q\} & \{q\} & \{p,q\} & \{p\} & \{\} & \{q\} & \{\} & \{\} & \cdots \\ \frac{0}{1} & \frac{0}{2} & \frac{1}{3} & \frac{2}{4} & \frac{2}{5} & \frac{2}{6} & \frac{2}{7} & \frac{2}{8} & \cdots \end{array}$$

In the third position p holds and thus q has not to hold in the fourth. But while the first occurrence of p is enough to release q from all further positions for Φ, the prefix does not allow to judge about the evaluation of Φ'. In the last position the fraction for p is not greater then $\frac{1}{4}$ any more and thus the next position is *not* released and must continue by a letter including q, e.g. $\{q\}$.

For the classical temporal modalities finally (F) and globally (G), which are defined in terms of the until, we find that the modification with frequencies does not affect the semantics. We defined $\mathrm{F}\,\varphi$ by $\top\,\mathrm{U}\,\varphi$ but this is semantically equivalent to $\top\,\mathrm{U}^c\,\varphi$ for

any $c \in [0,1]$ because allowing $\top$ to hold less frequently does not change anything as it always holds. Dually, $\mathrm{G}\,\varphi = \neg\,\mathrm{F}\,\neg\varphi = \bot\,\mathrm{R}^c\,\varphi$ keeps the exact same semantics for any c since $\bot$ does not hold, in particular not "often".

This coincides with the initial idea of specifying frequentness only within some *scope*. The only reasonable meaning of "weakening" G on infinite words could be some kind of probabilistic assertion what we tried to avoid.

By these considerations we confirm that classical LTL (including its abbreviations) forms a fragment of $fLTL$ with restrictions $c = 1$ for U^c and $c = 0$ for R^c. Note that, again, we can assume a *positive normal form* for $fLTL$ formulae and it coincides with the one for LTL.

Moreover, we observe that LTL is also a *true semantic fragment* considering ω-languages.

Theorem 3.1 ($fLTL$ is not context free)**.** *Consider alphabets* $\Sigma_n = \{a_1, \ldots, a_n, b\}$ *and the class of languages*

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

While not being context free for $n > 2$*, all of the languages* L_n *are definable by an* $fLTL$ *formula*

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

□

One more observation we make, is that the unfolding equation

$$\varphi\,\mathrm{U}\,\psi \equiv \psi \vee (\varphi \wedge \mathrm{X}(\varphi\,\mathrm{U}\,\psi))$$

does not hold any more for $\varphi\,\mathrm{U}^c\,\psi$ where $c < 1$. The formula $p\,\mathrm{U}^{0.4}\,q$, for example, has a model $w = \emptyset\{p\}\{q\}^\omega$, the number of positions before $\{q\}$ that satisfy p is $1 \geq 0.4 \cdot 2$). The right-hand side of the unfolding equation is violated since it requires the first position to satisfy either p or q.

The information that is passed on to the future may not be sufficient anymore. In the case of $c = 1$ the formula is immediately violated whenever neither ψ nor φ holds, unless ψ holds before. In other words, the "contribution" of a single position where neither ψ nor φ hold is annihilating any future, whereas for $c < 1$ this position may or may not be required to hold in order to obtain the required frequency in the end. The "weight" of a single position depends on the length of the scope. This also prohibits to adopt the unfolding by modifying the frequency, e.g. to something like

$$\psi \vee (\varphi \wedge \mathrm{X}(\varphi\,\mathrm{U}^{c'}\,\psi)) \vee \mathrm{X}(\varphi\,\mathrm{U}^{c''}\,\psi).$$

Any mapping from c to c' and c'' would depend on the final length of the scope in order to evaluate the influence the single position has with regard to the overall fraction.

3.2 From Minsky machines to $fLTL$

In the following section we will establish an undecidability result regarding satisfiability of $fLTL$ formulae.

Minsky machines [Min67] are a computation model that uses a program built from a simple set of instructions and two unbounded but none-negative counters. The machines are defined using the instruction set

$$\{\mathsf{inc}(k_i, l), \mathsf{dect}(k_i, l_1, l_2)\}.$$

The operation inc increase one of two counters k_1, k_2 and jumps to the instruction labeled with l in the program. The dect operation tests k_i for zero. Is that the case, the machine directly jumps to label l_1. Otherwise k_i is decreased and the next instruction that is executed is the one labelled by l_2.

For the forthcoming construction we decompose the instructions to limit the effect of each to a minimum which allows us to capsule different aspects.

First, the operation $\mathsf{dect}(k_i, l_1, l_2)$ is split. One operation $\mathsf{testz}(k_i, l_1, l_2)$ only tests k_i for zero and jumps to l_1 if that is the case and to l_2 otherwise. A second decrease operation $\mathsf{dec}(k_i, l)$ always jumps to the position l and decreases k_i by one if possible. The second step is now to remove the first argument – the counter – from *all* instructions and consider two copies of each. One copy operates on counter 1, the other on counter 2. We thus obtain the the following set of six operations.

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

Note that we did not lose any expressivity since the original instructions can still be simulated by these. (Nor do we gain any, since the converse holds as well.)

Definition 3.2 (Minsky machine)**.** A Minsky machine is a tuple

$$\mathcal{M} = (\pi, L, l_{\text{init}}, l_{\text{final}}, n_0, m_0),$$

where

- L is a none-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*,
- $\pi : L \times (\mathsf{OP} \times L \times L)$ is a set of labelled instructions, the *program*, where for every location $l \in L$, there is exactly one instruction $(l, \mathsf{op}, l_1, l_2) \in \pi$.

The set OP, as defined above is the set of operations on the two counters which can be used in a program π. Since for every location l, there is a unique tuple $(l, \mathsf{op}, n, m) \in \pi$, we may also interpret π as a mapping of locations to *instructions* and write $\pi(l)$ to denote (op, n, m).

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:

- If $\pi(l) = (\mathsf{inc}_1, l_1, l_2)$ then $C_{i+1} = (l_1, n+1, m)$.
- If $\pi(l) = (\mathsf{inc}_2, l_1, l_2)$ then $C_{i+1} = (l_1, n, m+1)$.
- If $\pi(l) = (\mathsf{dec}_1, l_1, l_2)$ then $C_{i+1} = (l_1, \max(0, n-1), m)$.
- If $\pi(l) = (\mathsf{dec}_2, l_1, l_2)$ then $C_{i+1} = (l_1, n, \max(0, m-1))$.
- If $\pi(l) = (\mathsf{testz}_1, l_1, l_2)$ then $C_{i+1} = (l', n, m)$, if $n = 0$ then $l' = l_1$ else $l' = l_2$.
- If $\pi(l) = (\mathsf{testz}_2, l_1, l_2)$ then $C_{i+1} = (l', n, m)$, if $m = 0$ then $l' = l_1$ else $l' = l_2$.
- If $l = l_{\text{final}}$ then $C_{i+1} = C_i$.

Note, that for simplicity we do not distinguish operations with one or two arguments but always assume two, even if the second one is not regarded at all.

It is possible to reduce the termination problem of Turing machines to that of Minsky machines, which is thus undecidable. The result still holds for our definition since the instruction set is able to simulate the instructions of standard Minsky machines via a translation

$$\begin{aligned} l : inc(k_i, l') &\to (l, \mathsf{inc}_i, l', l') \\ l : dect(k_i, l_1, l_2) &\to (l, \mathsf{testz}_i, l_1, l_2'),\ (l_2', \mathsf{dec}_k, l_2, l_2) \end{aligned}$$

for some newly created location l_2'. Observe, that assuming this translation, a dec operation is never performed on a counter that is zero already. Without loss of generality we assume that for any program π, since we could translate any program π' back to the instruction set $\{\mathsf{inc}, \mathsf{dect}\}$ and forth again, obtaining an equivalent program π that obeys that restriction.

Together with the standard result, the possible translation shows, that our notion of Minsky machines has an undecidable termination problem.

Lemma 3.2 (Termination of Minsky machines is undecidable [Min67])**.** *Given an arbitrary Minsky machine $\mathcal{M}$, it is undecidable whether l_f will be reached eventually or not during the computation of $\mathcal{M}$.*

Our goal is now, to express the chain of computation steps of an arbitrary Minsky machine $\mathcal{M}$ by an $fLTL$ formula $\Phi_{\mathcal{M}}$. The formula shall have exactly one model, which represents the computation of $\mathcal{M}$.

We will use, among others, the elements of L as letters in the alphabet over which the model will be defined. Given a model for $\Phi_{\mathcal{M}}$, the projection to L, will yield a subsequence $l_0 l_2 l_3 \ldots$ which shall correspond to the order, in which the single locations are visited during the execution of the program. Then, the question whether the Minsky machine $\mathcal{M}$ terminates reduces to whether $\Phi_{\mathcal{M}} \wedge \mathrm{F}\, l_{\text{final}}$ is satisfiable.

For the following construction we fix a Minsky machine $M = (\pi, L, l_{\text{init}}, l_{\text{final}}, n_0, m_0)$.

Encoding the computation

We encode counters unary by an according number of letters a or $\hat{a}$ for counter 1 and b or $\hat{b}$ for counter 2. We will use a,b to represent counter values *before* and $\hat{a}, \hat{b}$ for values *after* some operation.

The effect of an operation $\mathsf{op} \in \mathsf{OP}$ is represented by two letters, one for the effect on each of the counters individually. The considered letters appear slightly inconsistent in first place which is due to technical counting issues in the construction. The operations with their representative combination of letters is shown in Table 3.1. For example, the operation inc_2 increases counter 2, represented by letter i_b and does not affect counter 1, which can be considered as a neutral operation "skip", performed on counter 1 instead. In contrast to s_a, the effect of such a neutral operation on counter 2 is expressed implicitly by the absence of a letter, i.e. the empty word ε. We denote the two types of "operational" letters on the number of a and b by the sets

$$\mathsf{aOP} = \{i_a, d_a, s_a\} \qquad \text{and} \qquad \mathsf{bOP} = \{i_b, d_b\},$$

respectively. We have, for example, computations

$$a^n i_a \hat{a}^{n+1}, \qquad a^{n+1} d_a \hat{a}^n \quad \text{or} \quad b^n \hat{b}^n$$

that increase counter 1, decrease counter 1 or have no effect for counter 2, respectively.

To reflect whole instructions, we combine the single operations on each counter and precede the according location label from L.

$$l\ a^n \mathsf{op}_1 \hat{a}^{n'}\ b^m \mathsf{op}_2 \hat{b}^{m'}$$

OP	Letter for effect on counter 1	Letter for effect on counter 2
inc_1	i_a	ε
inc_2	s_a	i_b
dec_1	d_a	ε
dec_2	s_a	d_b
testz_1	s_a	ε
testz_2	s_a	ε

Table 3.1: The letters representing an operation on individual counters.

Consider the configuration $C = (l, n, m)$. Depending on which operation is specified by $\pi(l) = (\mathsf{op}, l', l'')$ we may represent the computation of the Minsky machine, e.g. as

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

Now, the current instruction at location l implies the next location l', possibly depending on the counter values, which we simply append after the computation on the counters. Since we want to reflect a sequence of operations, the counter values *after* the first operation and *before* the subsequent operation must coincide. A computation step might therefore be

$$l\ a^n i_a \hat{a}^{\mathbf{n+1}}\ b^m \hat{b}^m\ l'\ a^{\mathbf{n+1}}\ s_a \hat{a}^{n+1}\ b^m d_b \hat{b}^{m-1}$$

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

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

where $l_0 = l_I$ is the initial location of $\mathcal{M}$. For purely technical reasons (we need to start counting soon) we added a separator sign # between the representation of counter 2 in terms of b^n and the symbol representing the effect of the current operation on that counter.

Along with the encoding we implicitly constructed the alphabet over which we will interpret the final formula $\Psi_{\mathcal{M}}$. We use the symbols $a, \hat{a}, b, \dots$ as letters to describe the structure in the word as well as in formulae. Following our convention, they should therefore be atomic propositions, and as letters an abbreviation for the singletons $\{a\}, \{\hat{a}\}, \{b\}, \dots$. Thus we use the set

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

for the atomic propositions. Since $\Sigma = 2^{AP}$, the alphabet contains many letters that are not acutally used. We assume that these letters do not occur in any model and will asure that in the final formula.

Specifying program instructions

We shall continue with constructing formulae $\varphi(\mathsf{op}, l_1, l_2)$ that hold for words which start with a correct representation of the according instruction. These are formulae φ such that

$$\begin{aligned}
\mathcal{L}(\varphi(\mathsf{inc}_1, l_1, l_2)) &= \{a^n i_a \hat{a}^{n+1}\ b^m \# \hat{b}^m\ l_1 \mid n, m \in \mathbb{N}_0\}\ \Sigma^\omega \\
\mathcal{L}(\varphi(\mathsf{inc}_2, l_1, l_2)) &= \{a^n s_a \hat{a}^n\ b^m \# i_b \hat{b}^{m+1}\ l_1 \mid n, m \in \mathbb{N}_0\}\ \Sigma^\omega \\
\mathcal{L}(\varphi(\mathsf{dec}_1, l_1, l_2)) &= \{a^{n+1} d_a \hat{a}^n\ b^m \# \hat{b}^m\ l_1 \mid n, m \in \mathbb{N}_0\}\ \Sigma^\omega \\
\mathcal{L}(\varphi(\mathsf{dec}_2, l_1, l_2)) &= \{a^n s_a \hat{a}^n\ b^{m+1} \# d_b \hat{b}^m\ l_1 \mid n, m \in \mathbb{N}_0\}\ \Sigma^\omega \\
\mathcal{L}(\varphi(\mathsf{testz}_1, l_1, l_2)) &= \{a^{n+1} s_a \hat{a}^{n+1}\ b^m \# \hat{b}^m\ l_2 \mid n, m \in \mathbb{N}_0\}\ \Sigma^\omega \\
&\quad \cup \{s_a\ b^m \# \hat{b}^m\ l_1 \mid m \in \mathbb{N}_0\}\ \Sigma^\omega \\
\mathcal{L}(\varphi(\mathsf{testz}_2, l_1, l_2)) &= \{a^n s_a \hat{a}^n\ b^{m+1} \# \hat{b}^{m+1}\ l_2 \mid n, m \in \mathbb{N}_0\}\ \Sigma^\omega \\
&\quad \cup \{a^n s_a \hat{a}^n\ \#\ l_1 \mid n \in \mathbb{N}_0\}\ \Sigma^\omega
\end{aligned}$$

Before constructing the actual formulae we define a sub-formula that we can reuse quite often. As we see above, we often need to express $b^m \# \hat{b}^m\ l$ and we define a formula $\beta_\varepsilon(l)$ with

$$\mathcal{L}(\beta_\varepsilon(l)) = \{b^m \# \hat{b}^m\ l \mid m \in \mathbb{N}_0\}\ \Sigma^\omega.$$

Obviously, the words need to start either by b or by #. Thus we note $b \vee \#$. Then we need to require, that *as many* b *as* $\hat{b}$ occur, which we cannot do directly. However, we can express $b^m \hat{b}^m l \Sigma^\omega$ (c.f. Theorem 3.1) by

$$b\, \mathrm{U}^{\frac{1}{2}}\, l \wedge \hat{b}\, \mathrm{U}^{\frac{1}{2}}\, l$$

We now use a trick, which is similar to one from [HMO10]: The formula

$$((b \vee \#)\, \mathrm{U}^{\frac{1}{2}}\, l) \wedge \hat{b}\, \mathrm{U}^{\frac{1}{2}}\, l$$

expresses that the number of b *and* # together is equal to the number of $\hat{b}$. Assuming that there can be only one symbol # (which we will assert by another formula later), this is used to represent $b^m \# b^{m+1}$ or, if we cut off one b by shifting the starting point to the second position we get what we aimed at by

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

Again, we assume that there can be only one # and moreover, that the order in which symbols can occur is restricted to what we expect, for now.

Equipped that way, we proceed straight forward to define all operations.

$\varphi(\mathsf{inc}_1, l_1, l_2)$: The symbol i_a is used the same way as # was before, namely to enforce that the symbol $\hat{a}$ is present exactly once more than a:

$$\varphi(\mathsf{inc}_1, l_1, l_2) := (a \vee i_a)\, \mathrm{U}^{\frac{1}{2}} (\beta_\varepsilon(l_1)) \ \wedge\ \hat{a}\, \mathrm{U}^{\frac{1}{2}} (\beta_\varepsilon(l_1))$$

$\varphi(\mathsf{inc}_2, l_1, l_2)$: Again, we first define an auxiliary formula $\beta_{\mathsf{inc}}(l)$, which expresses that there is one more $\hat{b}$ than symbols b, using the same ideas and cutting off one symbol since we have an additional #: $b^m \# i_b \hat{b}^{m+1} l$.

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

The number of a is to be kept, and we obtain

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

$\varphi(\mathsf{dec}_1, l_1, l_2)$: We do the same, except that the operator symbol now counts for $\hat{a}$ since they should be one less.

$$\varphi(\mathsf{dec}_1, l_1, l_2) := a\, \mathrm{U}^{\frac{1}{2}}\, \beta_\varepsilon(l_1) \ \wedge\ (d \vee \hat{a})\, \mathrm{U}^{\frac{1}{2}}\, \beta_\varepsilon(l_1)$$

$\varphi(\mathsf{dec}_2, l_1, l_2)$: For decreasing the number of b we employ the auxiliary formula

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

and define

$$\varphi(\mathsf{dec}_1, l_1, l_2) := (a \vee s_a) \wedge \mathrm{X}\left((a \vee s_a)\, \mathrm{U}^{\frac{1}{2}}\, \beta_{\mathsf{dec}}(l_1) \ \wedge\ \hat{a}\, \mathrm{U}^{\frac{1}{2}}\, \beta_{\mathsf{dec}}(l_1)\right)$$

$\varphi(\mathsf{testz}_1, l_1, l_2)$: For the test operation we require that there is either no a, and thus the word starts with the skip-symbol s_a or there is at least one a in the beginning. In the former case the last letter will be the label l_1, in the latter the label l_2. We perform neutral operation on both counters, thus having s_a for counter 1 and ε for counter 2.

$$\begin{gathered}\varphi(\mathsf{testz}_1, l_1, l_2) := \\ s_a \ \wedge\ \mathrm{X}\left((b \vee \#) \wedge \mathrm{X}\left((b \vee \#)\, \mathrm{U}^{\frac{1}{2}}\, l_1 \ \wedge\ \hat{b}\, \mathrm{U}^{\frac{1}{2}}\, l_1\right)\right) \\ \vee \\ a \ \wedge\ \mathrm{X}\left((a \vee s_a)\, \mathrm{U}^{\frac{1}{2}}\, \beta_\varepsilon(l_2) \ \wedge\ \hat{a}\, \mathrm{U}^{\frac{1}{2}}\, \beta_\varepsilon(l_2)\right)\end{gathered}$$

$\varphi(\mathsf{testz}_2, l_1, l_2)$: For the last operation we assure that there is a skip symbol between an equal number of a and $\hat{a}$ and then that there is no b at all but a # followed by

the label l_1 or there is at least one b and then the number of $\hat{b}$ must be equal and the label must be the second, l_2.

$$\varphi(\mathsf{testz}_2, l_1, l_2) :=$$
$$(a \vee s_a) \ \wedge \ \mathrm{X}\left((a \vee s_a)\,\mathrm{U}^{\frac{1}{2}}(\# \wedge \mathrm{X}\, l_1)\right) \wedge \ \mathrm{X}\left(\hat{a}\,\mathrm{U}^{\frac{1}{2}}(\# \wedge \mathrm{X}\, l_1)\right)$$
$$\vee$$
$$(a \vee s_a) \ \wedge \ \mathrm{X}\left((a \vee s_a)\,\mathrm{U}^{\frac{1}{2}}(b \wedge \beta_\varepsilon(l_2)) \wedge \ \mathrm{X}\left(\hat{a}\,\mathrm{U}^{\frac{1}{2}}(b \wedge \beta_\varepsilon(l_2))\right)\right.$$

From the program π we can now compose the constructed formulae, such that we obtain a restriction that enforces that any model must mimic the operations, at every point where an according label occurs. Let hence

$$\varphi_\pi := \mathrm{G}\left(\bigwedge_{(l,\mathsf{op},l_1,l_2)\in\pi} l \rightarrow \varphi(\mathsf{op}, l_1, l_2)\right).$$

Enforcing propagation

Now that we can assure that at any label, given a counter value, the according operation is executed correctly we need to establish the connection between each computation step. In other words: We need to ensure that the result of a computation, i.e. the powers of $\hat{a}^n$ and $\hat{b}^m$, is *copied* correctly to the input, in terms of a and b, of the next operation. Recall, the scheme of the computation sequence

$$l_0\ a^{n_0}\ \mathsf{op}_a\ \hat{a}^{n_1}\ b^{m_0}\#\mathsf{op}_b\ \hat{b}^{m_1}\ l_1\ a^{n_1}\ \mathsf{op}'_a\ \hat{a}^{n_2}\ b^{m_1}\#\mathsf{op}'_b\ \hat{b}^{m_2}\ l_2 \ldots \tag{3.2}$$

We assured a correct translation from a^{n_0} to $\hat{a}^{n_1}$ and b^{m_0} to $\hat{b}^{m_1}$. Now, we need to propagate the values further via the translation $\hat{a}^{n_1}$ to a^{n_1} and $\hat{b}^{m_1}$ to b^{m_1}.

The issue we face, is, again that we can actually not express quantitative equivalences of symbols over "distances" in words. We managed in the last section to bridge one or two symbols between letters, e.g. a and $\hat{a}$, that are supposed to appear in the same number. Now for copying the counter values between configurations represented in the word we face variable domains in between, namely that part where the other counter changes (or not) according to an operation.

However, we can make use of an invariant. Consider the crucial frame for copying symbols a:

$$\ldots \mathsf{op}_a \underbrace{\hat{a}^n \; b^m \# \mathsf{op}_b \; \hat{b}^{m'} \; l \; a^n}\; \mathsf{op}'_a \ldots$$

We investigate the sub-term $b^m \# \mathsf{op}_b \hat{b}^{m'} l$ that ends just before an operation from aOP occurs. Consider the case where $\mathsf{op}_b = i_b$, then we have $m' = m + 1$ and the length of the whole term is

$$|b^m| \; + |\#| \; + |i_b| \; + |\hat{b}^{m+1}| \; + |l| = m \; + 1 \; + 1 \quad + (m+1) \; + 1.$$

Thus, if we separate the letters $b, \#, i_b$ and $\hat{b}, l$, both sets of symbols represent exactly one half of the positions. Therefore, if we require for the whole term $\hat{a}^n b^m \# i_b \hat{b}^{m'} l a^{n'}$, that the number of positions for any of $\{\hat{a}, b, \#, i_b\}$ is equal to the number of position for any of $\{\hat{b}, a\} \cup L$, we implicitly require, that $n = n'$ because the rest is equal already. Thus we can formulate the condition

$$\begin{gathered}(\hat{a} \vee b \vee \# \vee i_b) \, \mathrm{U}^{\frac{1}{2}} (i_a \vee d_a \vee s_a) \\ \wedge \\ (\hat{b} \vee a \vee \nu_L) \, \mathrm{U}^{\frac{1}{2}} (i_a \vee d_a \vee s_a)\end{gathered}$$

which reflects our aim in the case of $\mathsf{op}_b = i_b$. Furthermore, we observe, that even in all the other cases, where $\mathsf{op}_b = d_b$ and $\mathsf{op}_b = \varepsilon$, we can count d to the right-hand side and get

$$|b^{m+1} \# d_b \hat{b}^m l| = (m+1) \; + 1 \quad + 1 \; + m \; + 1.$$

Also, for $\mathsf{op} = \varepsilon$, we get

$$|b^m \# \hat{b}^m l| = m + 1 \; + \; m + 1.$$

For better reading we denote

$$A_{\text{left}} := \{\hat{a}, b, \#, i_b\} \quad \text{and} \quad A_{\text{right}} := \{\hat{b}, a\} \cup L$$

and the disjunction of all formulae from a set M by

$$\nu_M := \bigvee_{m \in M} M.$$

Whenever an operator symbol for the first counter occurs we want to ensure that the outcome of that operation is propagated.

Therefore we set

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

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 the op_a sign, we find the symbols

$$B_{\text{left}} := \{\hat{b}, a\} \cup L$$

and to the right

$$B_{\text{right}} := \{\hat{a}, b\}.$$

Next, consider the following three cases for op_a.

$\mathsf{op}_a = i_a$: When counter 1 is increased, we have

$$|\hat{b}^m \; l \; a^n i_a \hat{a}^{n+1} b^m| = m + 1 + n + 1 + n + 1 + m$$

and thus an offset of 1 when counting i_a to the left part. Therefore, again, we cut off the first position and require then that the blue and green symbols are equally many which implies that the number of $\hat{b}$ equals the number of b.

We express the quantitative equality between the two sets of symbols by

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

Note, that this formula can only be true if the operation on counter 1, i.e. the symbol between a and $\hat{a}$, is actually i_a, not d_a or s_a. Also, the first # defines the scope, since any later # comes after a symbol from bOP and those are neither in B_{left} nor in B_{right}. Since the formula requires, that the ratio of symbols from each set must be at least one half, any occurrence of a "foreign" symbol (e.g. from bOP) within the scope would lead to a violation of this property.

$\mathsf{op}_a = d_a$: If the counter is supposed to be decreased the according computation looks like

$$|\hat{b}^m \; l \; a^{n+1} d_a \hat{a}^n b^m| = m + 1 + n + 1 + 1 + n + m$$

and if we count the operator d_a to the right-hand side symbols this time we have an offset of 2. Still, we can express the correct propagation by

$$\psi_{\mathsf{dec}} := \mathrm{XX}\left(\left(\nu_{B_{\text{left}}} \, \mathrm{U}^{\frac{1}{2}} \, \# \right) \; \wedge \; \left((\nu_{B_{\text{right}}} \vee d_a) \, \mathrm{U}^{\frac{1}{2}} \, \# \right) \right)$$

because we can assume that there are at least two symbols on the left (blue) side: one label l and at least one letter a.

$\mathsf{op}_a = s_a$: The last case we have to consider is that of a neutral operation on a. Then, we count

$$|\hat{b}^m \; l \; a^n s_a \hat{a}^n b^m| = m + 1 + n + 1 + n + m$$

and see, that we do not need to cut off anything. The inner part is equally distributed and by requiring that both types occur equally often before the # symbol by

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

we implicitly express the correct propagation.

Combining all these cases, we obtain a formula which expresses that in any case, the propagation is done correctly. The property must always hold right after the the position of the operator symbol for counter 2. Recall, that these symbols for counter two 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.

$$\psi_b = \mathrm{G}\left((\nu_{\mathsf{bOP}} \vee (\# \wedge \neg \mathrm{X}\,\nu_{\mathsf{bOP}}) \rightarrow \mathrm{X}\left(\psi_{\mathsf{inc}} \vee \psi_{\mathsf{dec}} \vee \psi_{\mathsf{skip}}\right)\right)$$

While we could express all cases for ψ_a by a single formula we need to consider the operations individually for ψ_b because of the different offsets.

The final formula

Now that we have φ_π to ensure the correct operations on the counters and ψ_a, ψ_b to ensure the correct propagation of counter 1 and 2, respectively, the last task is to enforce a correct ordering of the symbols, since we assumed that at some points. This is not difficult with the schema of the encoded computation in mind (Equation 3.2):

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

We subsume these formulae by a conjunction ζ thereof. Additionally we initialize the computation according to the initial configuration of $\mathcal{M}$ by a formula φ_I. This reflect basically setting the counter values through a required fixed prefix. Formally, φ_I describes the language

$$\mathcal{L}(\varphi_I) = \{l_0\, a^{n_0}\, \mathsf{op}_a\, \hat{a}^{n_1}\, b^{m_0}\#\}\Sigma^\omega$$

which can be achieved easily with a conjunction of a one formula for each position in the fixed prefix:

$$\varphi_I = l_0 \wedge \left(\bigwedge_{j=1}^{n_0} \mathrm{X}^j(a)\right) \wedge \left(\mathrm{X}^{n_0+1}\,\mathsf{op}_a\right) \wedge \left(\bigwedge_{j=0}^{n_1} \mathrm{X}^{n_0+1+j}\,\hat{a}\right) \ldots$$

Combining all the formulae

- φ_π, assuring correct instruction and computation of counter values in a configuration,
- ψ_a, ψ_b, forcing the correct propagation from configuration to configuration,
- $\mathrm{G}\,\zeta$, ensuring the correct ordering of all symbols globally,
- φ_I, representing the initial configuration,

finally yields the formula

$$\Phi_{\mathcal{M}} = \varphi_\pi \wedge \psi_a \wedge \psi_b \wedge \mathrm{G}\,\zeta \wedge \varphi_I.$$

By construction, $\Phi_{\mathcal{M}}$ describes exactly the computation of $\mathcal{M}$. We conclude, that the formula $\Phi_{\mathcal{M}} \wedge Fl_{\text{final}}$ is satisfiable if and only if $\mathcal{M}$ terminates after finitely many computation steps. Now, by Lemma 3.2 above, we have proven the following result.

Theorem 3.2 (Undecidability of $fLTL$)**.** *The satisfiability problem for $fLTL$ formulae is undecidable.*

4 Game-Representation and Satisfiability for $fLTL$

Even though there is no hope to find an approach to the satisfiability problem for arbitrary $fLTL$ formulae we want to put some more attention to the problem in order to gain a better insight into the formalism and it's fragments. We approach LTL from a game theoretical view, namely by extending focus games as presented in Section 2.3.1. Our aim is to understand the novel logic and we believe that games provide a natural intuition.

Recall the unfolding equality

$$w \models \varphi \,\mathrm{U}\, \psi \quad \Leftrightarrow \quad w \models \psi \vee (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}\, \psi)).$$

In $fLTL$, this equality does only hold for $c = 1$. The reason is, that the "contribution" of a single position in a word can be arbitrary small since the length of the scope is not bounded. As soon as $c < 1$, any number of violating positions can be "covered" by enough satisfying positions. Take e.g. $p \,\mathrm{U}^{\frac{2}{3}}\, q$ and a word starting with $\emptyset^n$. For any n we can append $2n$ positions that satisfy p and thereby fulfilling the required frequentness. Hence a finite prefix can never prove that a word violates this formula, it can at most prove satisfaction.

However, a tableaux-like approach relies on some notion of unfolding. Therefore we consider the standard unfolding and recall the idea of "weakening" obligations. In Figure 4.1 the unfolding of the formula $p \,\mathrm{U}\, q$ is represented as $\wedge$-$\vee$-graph. Let us associate two players E and A with nodes labelled $\vee$ and $\wedge$, respectively. Together with a word w, the graph defines a game that is won by (the existential player) E, owning the $\vee$-nodes, if she can always reach a $\top$-node, regardless of the choices of A on his nodes. This is the case, if w satisfies the formula. Moreover, the formula is satisfiable if and only if E can *choose* a word (i.e. a strategy) such that she wins the according game.

Consider now the formula $p \,\mathrm{U}^{0.8}\, q$, intuitively meaning that a word satisfies the formula already if 80% of the positions before q satisfy p. Regarding the game, this is like allowing E to cheat at some positions of A by pruning the left branch. However, E should only be able to prune sub-trees that result from the left side of the U-operator when it is unfolded. (Figure 4.1, right-hand side)

On the logical side this means that there is an alternative during the unfolding. For the formula $p \,\mathrm{U}^{0.8}\, q$, either p *and* $\mathrm{X}(p \,\mathrm{U}^{0.8}\, q)$ must hold, or *just* $\mathrm{X}(p \,\mathrm{U}^{0.8}\, q)$. Yet,

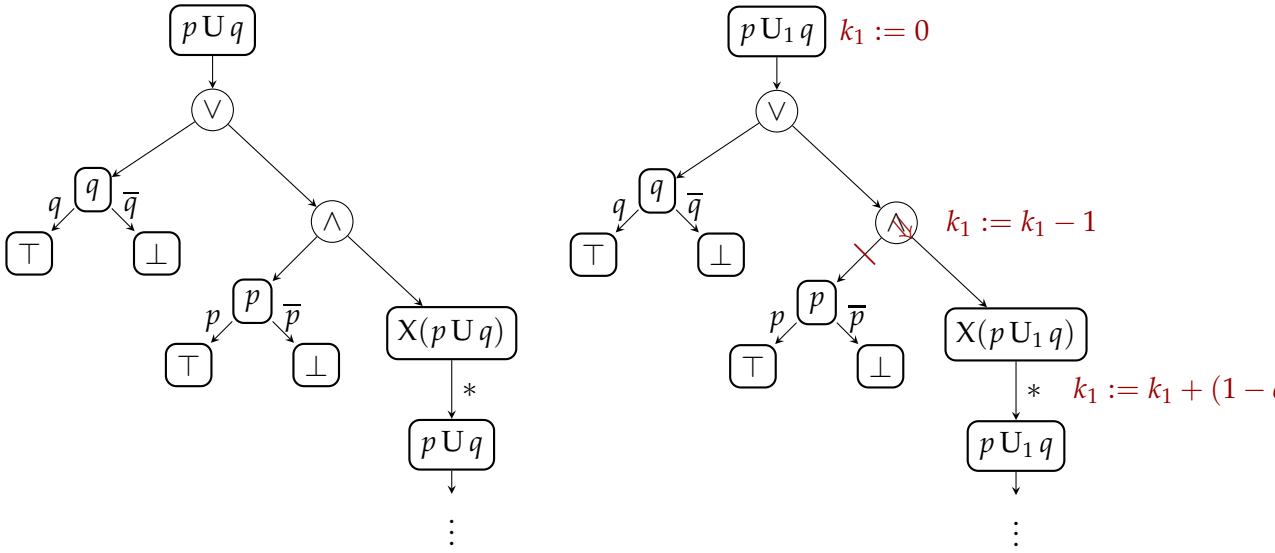


Figure 4.1: A representation of the LTL formula pUq as game graph. We consider it generic for a word w, that determines the choice on the labelled edges.

we need to restrict how often p can be disregarded, i.e. obligate E to not cheat too often. This can be realized using an *account* that keeps the credit E has left or a debt, respectively. We can calculate that way: E gets a regular income for every position passed in the word. Every time E decides to cheat, i.e. to dismiss the left branch from the U^c-operator, the account is additionally debited by 1. Taking $1 - c$ for the income gives E exactly that much credit, that for every $(1 - c)^{-1}$ positions played E earns one credit to invest in cheating. Therefore, whenever the account is non-negative, E has not cheated too often since the required frequentness so far is met. In the example $p\,\mathrm{U}^{0.8}\,q$, p has to hold in 80% of the positions. For every $5 = (1 - \frac{4}{5})^{-1}$ steps played, E can choose one position to prune, and has thus not to assure p holds there.

4.1 Counter semantics for $fLTL$

An account for a sub-formula $\varphi\,\mathrm{U}^c\,\psi$ can store information about the "history" of a position. The unfolding separates such a formula into a local obligation φ and a postponed obligation $\mathrm{X}\,\varphi\,\mathrm{U}\,\psi$, including the eventuality ψ. In the frequentness setting the decision whether the eventuality can hold now is also dependant on some global property, namely if the obligation was satisfied often enough in the past. Previous decisions have to be recorded and that is what the counter is good for. With the notion of credit it tracks exactly the information needed in the future. We can formulate that by saying "satisfy φ and earn credit or do not and loose some".

To formalize this idea we define an alternative, yet sound semantics that allows us

to bias formulae by setting an explicit counter value.

The syntax of $fLTL$ is extended by a counter bias as subscript for the U-operator.

$$\varphi ::= fLTL \mid \varphi \,\mathrm{U}^c_k\, \varphi$$

Recall the definition of the U^c-operator for $fLTL$ (Definition 3.1):

$$w \models \varphi \,\mathrm{U}^c\, \psi \quad \text{iff} \quad \begin{array}{l} \exists_n : w^n \models \psi \text{ and} \\ \#_{\varphi,w}(n) \geq c \cdot n \end{array}$$

We can rewrite $\#_{\varphi,w}(n) \geq c \cdot n$ to $\#_{\varphi,w}(n) - c \cdot n \geq 0$, that reflects our intuition for the counter: For every position (i.e. n times), c is subtracted, and for all the positions that satisfy φ, 1 is added and therefore these positions amount to $1 - c$, what we had in mind. Now, presetting a bias in terms of a counter value is straight forward:

$$w \models \varphi \,\mathrm{U}^c_k\, \psi \quad \text{iff} \quad \begin{array}{l} \exists_n : w^n \models \psi \text{ and} \\ k + \#_{\varphi,w}(n) - c \cdot n \geq 0 \end{array}$$

For $k = 0$, the definitions obviously coincide and we consider U^c as abbreviation for U^c_0 and consequently U for U^1_0.

The definition yields, what should happen for an unfolding by considering one single position, i.e. $n = 1$. If that position satisfies φ we get to add $1 - c$ to the counter and otherwise $0 - c$. Then the counter will always indicate a potential "ending" of the unfolding along the word being greater or equal to zero.

Definition 4.1 (Counted unfolding)**.** Let Φ be an $fLTL$ formula. The unfolding on LTL formulae is extended to the counter semantics as follows.

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

We may write more concisely $\langle \psi \vee \rangle^{k \geq 0} (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}^c_{k+1-c}\, \psi))$ when convenient to reflect that the presence of the first part $\psi \vee$ depends on the counter.

So far we argued informally about how the counters should behave. The following Lemma 4.1 confirms that Definition 4.1 is purely syntactic and does not change the semantics of a formula.

Lemma 4.1 (Counted unfolding equivalence)**.** *Let Φ be a (possibly biased) $fLTL$ formula and $w \in \Sigma^\omega$. Then*

$$w \models \Phi \quad \textit{iff} \quad w \models \mathsf{unf}(\Phi).$$

Proof. For $\Phi \neq \varphi \,\mathrm{U}_k^c\, \psi$ the unfolding does not affect the formula and the result follows trivially. Therefore, let $\Phi = \varphi \,\mathrm{U}_k^c\, \psi$.

Case 1: $k \geq 0$**.** By definition we have

$$w \models \varphi \,\mathrm{U}_k^c\, \psi \qquad \Leftrightarrow \qquad \exists_n : \left(\begin{array}{l} w^n \models \psi \text{ and} \\ k + \#_{\varphi,w}(n) - c \cdot n \geq 0 \end{array} \right).$$

Splitting the cases $n = 0$ and $n > 0$ we get for the right-hand side above

$$(w^0 \models \psi \text{ and } k \geq 0) \quad \text{or} \quad \exists_n : \left(\begin{array}{l} w^{n+1} \models \psi \text{ and} \\ k + \#_{\varphi,w}(n+1) - c \cdot (n+1) \geq 0) \end{array} \right). \tag{4.1}$$

The left conjunct reduces to $w \models \psi$ since we assumed $k \geq 0$. On the right-hand side we rewrite $\#_{\varphi,w}(n+1)$ depending on the first position. If φ holds at the first position of w we have $\#_{\varphi,w}(n+1) = \#_{\varphi,w^1}(n) + 1$ and $\#_{\varphi,w}(n+1) = \#_{\varphi,w^1}(n) + 0$ otherwise. Also we have $w^{n+1} = (w^1)^n$. Hence we obtain

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

We can distribute the existential quantifier over the disjunction:

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

By the equalities

$$\begin{aligned} k + 1 + \#_{\varphi,w^1}(n) - c \cdot (n+1) &= (k + 1 - c) + \#_{\varphi,w^1}(n) - c \cdot n, \\ k + 0 + \#_{\varphi,w^1}(n) - c \cdot (n+1) &= (k - c) + \#_{\varphi,w^1}(n) - c \cdot n \end{aligned}$$

we get

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

which reduces by the definition of the U-operator to

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

by the definition of the 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 finally to

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

Case 2: $k < 0$**.** The second case is almost identical. The only difference is that in Equation 4.1 the left-hand side evaluates to false and therefore the term $w \models \psi$ disappears in all following equations. Still, the last equation holds by

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

□

Let us examine what happens to the Release. We continue analogously by defining

$$\varphi \,\mathrm{R}^c_k\, \psi := \neg(\neg\varphi \,\mathrm{U}^{1-c}_k\, \neg\psi)$$

yielding

$$\begin{array}{rlllll} & \neg\Big(& \exists_n : w^n \models \neg\psi & \text{and} & k + \#_{\neg\varphi,w}(n) - n \cdot (1-c) \geq 0 \Big) \\ \Leftrightarrow & & \forall_n : w^n \models \psi & \text{or} & k + \#_{\neg\varphi,w}(n) - n \cdot (1-c) < 0. & \end{array}$$

Again, φ and $\neg\varphi$ are mutually exclusive and therefore $\#_{\neg\varphi,w}(n) = n - \#_{\varphi,w}(n)$ and hence

$$\begin{array}{lll} \Leftrightarrow & \forall_n : w^n \models \psi \quad \text{or} \quad k + (n - \#_{\varphi,w}(n)) - n \cdot (1-c) < 0 \\ \Leftrightarrow & \forall_n : w^n \models \psi \quad \text{or} \quad k - \#_{\varphi,w}(n) + n \cdot c < 0. \end{array} \tag{4.2}$$

Consider a word w that satisfies a formula $\varphi \,\mathrm{U}^c\, \psi$. Then, there is some *witnessing* position n in w, i.e. w^n satisfies ψ and $n = 0$ or the freuqentness

$$f_{\varphi,w}(n) := \frac{\#_{\varphi,w}(n)}{n} \quad (n > 0)$$

of positions satisfying φ on the prefix $a_0 a_1 \ldots a_{n-1}$ of w is at least c. At any position n' before n where the freuqentness of φ is insufficient, i.e. where $f_{\varphi,w}(n') \leq c$ we know that the part $a_{n'} \ldots a_n$ must have a ratio of at least c, since the overall ratio at n is sufficient. Therefore $\varphi \,\mathrm{U}^c\, \psi$ holds at such a position n'.

In general we have the following implications.

Lemma 4.2 (Positional implications). *Let $w \in \Sigma^\omega$ satisfy an fLTL formula $\Phi = \varphi \,\mathrm{U}^c\, \psi$ and n be a witnessing position in w for Φ. Then for all $0 < i < n$ we have*

$$\begin{aligned} w^i \models \varphi \,\mathrm{U}^c\, \psi \quad \textit{and} \quad f_{\varphi,w}(i) \geq c \quad &\Rightarrow \quad w \models \varphi \,\mathrm{U}^c\, \psi, \\ w \models \varphi \,\mathrm{U}^c\, \psi \quad \textit{and} \quad f_{\varphi,w}(i) \leq c \quad &\Rightarrow \quad w^i \models \varphi \,\mathrm{U}^c\, \psi. \end{aligned}$$

In the counter setting we have the related observation, for $k \leq k'$,

$$\begin{aligned} \varphi \,\mathrm{U}^c_k\, \psi &\Rightarrow \varphi \,\mathrm{U}^c_{k'}\, \psi \\ \varphi \,\mathrm{R}^c_{k'}\, \psi &\Rightarrow \varphi \,\mathrm{R}^c_k\, \psi \end{aligned}$$

Proof. **1).** By definition we have

$$w^i \models \varphi \,\mathrm{U}^c\, \psi \quad \text{iff} \quad \exists_j : (w^i)^j \models \psi \text{ and } \#_{\varphi,w^i} \geq c \cdot j.$$

The intervals 0 to $i-1$ and i to $i+j-1$ do not overlap, nor have a position in between. Therefore the number of positions that satisfy φ in the combined interval 0 to $i+j-1$, i.e. before $i+j$, is

$$\#_{\varphi,w}(i+j) = \#_{\varphi,w}(i) + \#_{\varphi,w^i}(j).$$

Recall that w^i starts with $a_i a_{i+1} \ldots$, while a_{i-1} is the last letter of w that is taken into account for $\#_{\varphi,w}(i)$.

With

$$f_{\varphi,w}(i) \geq c \quad \Leftrightarrow \quad \#_{\varphi,w}(i) \geq c \cdot n$$

we obtain

$$\#_{\varphi,w}(i+j) = \#_{\varphi,w}(i) + \#_{\varphi,w^i}(j) \geq c \cdot i + c \cdot j = c \cdot (i+j).$$

Hence, there is an $j' = i+j$ such that $w^{j'} = (w^i)^j \models \psi$ and $\#_{\varphi,w}(j') \geq c \cdot j'$. Thus $w \models \varphi \,\mathrm{U}^c\, \psi$.

2). We have a witness n for $w \models \varphi \,\mathrm{U}^c\, \psi$, i.e. $w^n \models \psi$ and $\#_{\varphi,w}(n) \geq c \cdot n$. Additionally $\#_{\varphi,w}(i) \leq c \cdot i$ with $0 < i < n$. Splitting

$$w = a_0 \ldots a_{i-1} w^i = a_0 \ldots a_{i-1} a_{i+0} \ldots a_{i+(n-i)} w^{n+1},$$

the number of positions before $n = i + n - i$ which satisfy φ can be written as

$$\begin{aligned} \#_{\varphi,w}(n) = \#_{\varphi,w}(i) + \#_{\varphi,w^i}(n-i) &\geq c \cdot n = c \cdot (i + (n-i)) \\ \geq c \cdot i \quad + c \cdot (n-i) & \end{aligned}$$

For $\#_{\varphi,w}(i) \leq c \cdot i$ it hence follows that $\#_{\varphi,w^i}(n-i) \geq c \cdot (n-i)$. Therefore

$$\begin{gathered} \exists_{n'=n-i} : (w^i)^{n'} = w^n \models \psi \text{ and } \#_{\varphi,w^i}(n') \geq c \cdot n' \\ \Leftrightarrow \\ w^i \models \varphi \,\mathrm{U}^c\, \psi. \end{gathered}$$

The implication for the counted formulae follow direct from the definition. For $k \leq k'$ and any model v, w for $\varphi \,\mathrm{U}^c_k\, \psi$ and $\varphi \,\mathrm{R}^c_{k'}\, \psi$, respectively,

$$\begin{aligned} k' + \#_{\varphi,w}(n) - c \cdot n \geq k + \#_{\varphi,w}(n) - c \cdot n \geq 0 \\ k - \#_{\varphi,w}(n) + c \cdot n \leq k' - \#_{\varphi,w}(n) + c \cdot n < 0, \end{aligned}$$

thus the counter condition is actually relaxed and therefore v, w satisfy in particular $\varphi \,\mathrm{U}^c_{k'}\, \psi$ and $\varphi \,\mathrm{R}^c_k\, \psi$, respectively. □

By the explicit semantics (Equation 4.2) we derive that when unfolding a release-formula, the counter must be modified such that c is added every time (term $+n \cdot c$) and an additional one can be subtracted if φ is asserted to hold (term $-\#_{\varphi,w}(n)$). We subsume:

Theorem 4.1 (Release-unfolding)**.** *In duality to the until operator we find*

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

Proof. We rewrite the release in terms of Until:

$$\mathsf{unf}(\varphi \,\mathrm{R}^c_k\, \psi) = \mathsf{unf}(\neg(\neg\varphi \,\mathrm{U}^{1-c}_k\, \neg\psi))$$

By Lemma 4.1 unf does not change the semantics of any formula, in particular not of subformulae.

$$w \models \mathsf{unf}(\neg(\neg\varphi \,\mathrm{U}^{1-c}_k\, \neg\psi)) \quad \Leftrightarrow \quad w \models \neg\mathsf{unf}(\neg\varphi \,\mathrm{U}^{1-c}_k\, \neg\psi))$$

Case 1: $k \leq 0$**.**

$$\begin{aligned} & w \models \neg\mathsf{unf}(\neg\varphi \,\mathrm{U}^{1-c}_k\, \neg\psi) \\ \Leftrightarrow\ & w \models \neg \left(\neg\psi \wedge \left(\begin{array}{l} (\neg\varphi \wedge \mathrm{X}(\neg\varphi \,\mathrm{U}^{1-c}_{k+1-(1-c)}\, \neg\psi)) \\ \vee\ \mathrm{X}(\neg\varphi \,\mathrm{U}^{1-c}_{k-(1-c)}\, \neg\psi) \end{array} \right) \right) \\ \Leftrightarrow\ & w \models \psi \vee \left(\begin{array}{l} (\varphi \vee \mathrm{X}(\varphi \,\mathrm{R}^c_{k+c}\, \psi)) \\ \wedge\quad \mathrm{X}(\varphi \,\mathrm{R}^c_{k+c-1}\, \psi) \end{array} \right) \\ \Leftrightarrow\ & w \models \psi \vee \left(\begin{array}{ll} (\varphi \wedge\ \mathrm{X}(\varphi \,\mathrm{R}^c_{k+c-1}\, \psi)) & \vee \\ (\mathrm{X}(\varphi \,\mathrm{R}^c_{k+c}\, \psi) \wedge\ \mathrm{X}(\varphi \,\mathrm{R}^c_{k+c-1}\, \psi)) & \end{array} \right) \qquad \text{(distribute)} \\ \Leftrightarrow\ & w \models \psi \vee \left(\begin{array}{ll} (\varphi \wedge\ \mathrm{X}(\varphi \,\mathrm{R}^c_{k+c-1}\, \psi)) & \vee \\ \mathrm{X}(\varphi \,\mathrm{R}^c_{k+c}\, \psi) & \end{array} \right) \qquad \text{(from Lemma 4.2)} \end{aligned}$$

Case 2: $k < 0$**.** The second case follows in analogy. The only change is to omit the conjunction/disjunction with ψ in the beginning of every line above, except for the first one. □

Remark. Even though the coincidence between counted and standard unfolding follows by the semantic equivalences

$$\mathsf{unf}(\varphi \,\mathrm{U}_0^1\, \psi) \equiv \varphi \,\mathrm{U}_0^1\, \psi \equiv \varphi \,\mathrm{U}\, \psi \equiv \mathsf{unf}(\varphi \,\mathrm{U}\, \psi)$$

established already we observe directly that

$$\mathsf{unf}(\varphi \,\mathrm{U}_0^1\, \psi) = \psi \vee (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}_{0+1-1}^1\, \psi)) \vee \mathrm{X}(\varphi \,\mathrm{U}_{0-1}^1\, \psi) \equiv \psi \vee (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}_0^1\, \psi))$$

because $\varphi \,\mathrm{U}_k^1\, \psi$ is not satisfiable for any $k < 0$. The definition requires (for $c = 1$) that $k + \#_{\varphi,w}(n) - n \cdot 1 \leq 0$ which cannot be the case since always $\#_{\varphi,w}(n) \leq n$.

In the next section, we need the here developed notion of unfolding to define a tableau-like game graph for $fLTL$ formulae.

4.2 Counter focus games

In Section 2.3.1 we introduced focus games that could be used to decide satisfiability for LTL formulae. The game graph was defined in terms of configurations that contain formulae and rules that are applied automatically or can be chosen by one of the players and lead to a preceding configuration. Paths in the game graph represent the plays of a game and if the existential player E had a winning strategy, we could derive a model and thus conclude satisfiability for the formula.

Using the developed counter semantics for $fLTL$, which yields a notion of unfolding, we now aim at adopting this technique for $fLTL$. Making use of the counter, the unfolding stores information about the past, a bias, for each until operators. That way the eventuality of an until formula can be evaluated with regard to the (relative) quantity of met obligations in the past.

The only syntactic change is the difference in the unfolding function unf and that (sub-)formulae may be equipped with a counter k and a frequency c. Therefore, syntactically, the set of sub-formulae in $fLTL$ is larger, in fact it is infinite since counter values can grow and decrease arbitrarily by application of unf.

Still, we keep the definition of sub-formulae, disregarding counters. We only discriminate these formulae by the constant frequency c such that, e.g. for

$$\varphi = p \,\mathrm{U}_0^{0.8}\, q \vee (p \wedge (p \,\mathrm{U}_{-3}^{0.4}\, q \vee p \,\mathrm{U}_4^{0.8}\, q))$$

the set of sub-formulae $\mathsf{sub}(\varphi)$ includes $p \,\mathrm{U}^{0.8}\, q$ and $p \,\mathrm{U}^{0.4}\, q$ but not their counted versions $p \,\mathrm{U}_0^{0.8}\, q$, $p \,\mathrm{U}_4^{0.8}\, q$ or $p \,\mathrm{U}_{-3}^{0.4}\, q$. Also, due to the different unfolding, there is one more sub-formula for each until.

For the games on $fLTL$ formulae the set of configurations is adjusted accordingly and the major difference is, that it is not finite anymore. Not only plays can last

arbitrary long without revisiting a configuration, also configurations themselves might grow, since they might accumulate the same sub-formula arbitrary often, only with different counters. Here we can add a new rewriting rule, that reduces configurations such that they contain only one instance of a formula at a time and are therefore again bounded by $|\mathsf{sub}(\Phi)|$ for a formula Φ.

Definition 4.2 (Extended Focus Games)**.** Let Φ be an $fLTL$ formula in positive normal form.

The counted focus game on Φ is a graph $G(\Phi) = (V, E)$ where

- $V \subseteq \mathsf{sub}(\Phi) \times \mathbb{Q} \times 2^{\mathsf{sub}(\Phi)\times\mathbb{Q}}$ is the set of *configurations* and
- $E \subseteq V \times V$ is a set of possible *moves*.

We write configurations $C = ([\varphi], \Gamma) \in V$ meaning that φ is the formula in focus and it is thus required that $\varphi \in \Gamma$. The initial configuration remains

$$C_0 = (\mathsf{unf}(\Phi), \{\mathsf{unf}(\Phi)\}).$$

The moves (C, C') are defined according to Figure 4.2. Configurations are directly rewritten to contain only one instance of a counted sub-formula at a time and to break up $\wedge$-formulae that do not have the focus.

Configurations C_1, C_2 are identified if they result in the same configuration after rewriting.

Note that, for the rewriting rules from Figure 4.2, the order of application does not matter. In particular either the rule for breaking up the $\wedge$ operator or those for merging similar until and release formulae can be applied.

Merging similar until and release formulae is done *implicitly* in the standard rule set (Figure 2.3) as well as in the extended rule set for identical formulae since they are stored in sets. For example a formula $\varphi \wedge \varphi$ reasonably rewrites $\{\mathsf{unf}(\varphi), \mathsf{unf}(\varphi)\} \to \{\mathsf{unf}(\varphi)\}$. This implicit rewriting ensures in standard focus games the finiteness of the game graph. During the unfolding new formulae are added and if they were all treated individually the state space would be unbounded. In the counter setting the unfolding affects the counters which may increase or decrease arbitrarily and the resulting formulae would be syntactically different. E.g. unfolding a formula $p\,\mathrm{U}_0^{0.8}\,q$ again and again, results - amongst others - in possible sub-formulae $p\,\mathrm{U}_{-0.8}^{0.8}\,q$, $p\,\mathrm{U}_{-1.6}^{0.8}\,q$, $p\,\mathrm{U}_{-2.4}^{0.8}\,q$, and so on. These are all not equal, neither syntactically nor semantically, however, when appearing in conjunction, as they implicitly do in our game configurations, we can discard all but the one with the least counter value. The implications from Lemma 4.2 assure that this does not give advantage nor disadvantage to either of the players since

$$\bigwedge_i \varphi\,\mathrm{U}_{k_i}^{c}\,\psi \equiv \varphi\,\mathrm{U}_{\min_i(k_i)}^{c}\,\psi.$$

Player E

$$\frac{[\varphi_0 \vee \varphi_1], \{\varphi_0 \vee \varphi_1\} \cup \Gamma}{[\mathsf{unf}(\varphi_i)], \{\mathsf{unf}(\varphi_i)\} \cup \Gamma} \qquad \frac{[\psi], \{\varphi_0 \vee \varphi_1\} \cup \Gamma}{[\psi], \{\mathsf{unf}(\varphi_i)\} \cup \Gamma}$$

Player A

$$\frac{[\varphi_0 \wedge \varphi_1], \{\varphi_0 \wedge \varphi_1\} \cup \Gamma}{[\mathsf{unf}(\varphi_i)], \{\mathsf{unf}(\varphi_{1-i})\} \cup \Gamma} \qquad \frac{[\varphi], \{\varphi, \psi\} \cup \Gamma}{[\psi], \{\varphi, \psi\} \cup \Gamma} \text{ (change)}$$

Rewriting rules

$$\frac{[\mathrm{X}(\varphi \,\mathrm{R}^c_{k_i}\, \psi)], \{\mathrm{X}(\varphi \,\mathrm{R}^c_{k_0}\, \psi), \mathrm{X}(\varphi \,\mathrm{R}^c_{k_1}\, \psi)\} \cup \Gamma}{[\mathrm{X}(\varphi \,\mathrm{R}^c_{\max(k_0,k_1)}\, \psi)], \{\mathrm{X}(\varphi \,\mathrm{R}^c_{\max(k_0,k_1)}\, \psi)\} \cup \Gamma} \qquad \frac{[\xi], \{\mathrm{X}(\varphi \,\mathrm{R}^c_{k_0}\, \psi), \mathrm{X}(\varphi \,\mathrm{R}^c_{k_1}\, \psi)\} \cup \Gamma}{[\xi], \{\mathrm{X}(\varphi \,\mathrm{R}^c_{\max(k_0,k_1)}\, \psi)\} \cup \Gamma}$$

$$\frac{[\mathrm{X}(\varphi \,\mathrm{U}^c_{k_i}\, \psi)], \{\mathrm{X}(\varphi \,\mathrm{U}^c_{k_0}\, \psi), \mathrm{X}(\varphi \,\mathrm{U}^c_{k_1}\, \psi)\} \cup \Gamma}{[\mathrm{X}(\varphi \,\mathrm{U}^c_{\min(k_0,k_1)}\, \psi)], \{\mathrm{X}(\varphi \,\mathrm{U}^c_{\min(k_0,k_1)}\, \psi)\} \cup \Gamma} \qquad \frac{[\xi], \{\mathrm{X}(\varphi \,\mathrm{U}^c_{k_0}\, \psi), \mathrm{X}(\varphi \,\mathrm{U}^c_{k_1}\, \psi)\} \cup \Gamma}{[\xi], \{\mathrm{X}(\varphi \,\mathrm{U}^c_{\min(k_0,k_1)}\, \psi)\} \cup \Gamma}$$

$$\frac{[\psi], \{\varphi_0 \wedge \varphi_1\} \cup \Gamma}{[\psi], \{\mathsf{unf}(\varphi_0), \mathsf{unf}(\varphi_1)\} \cup \Gamma}$$

Next

$$\frac{[\mathrm{X}\,\varphi_1], \{\mathrm{X}\,\varphi_1, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}}{[\mathsf{unf}(\varphi_1)], \{\mathsf{unf}(\varphi_1), \ldots, \mathsf{unf}(\varphi_n)\}}$$

Figure 4.2: Moves for counted focus games. Moves are to be read from top to bottom and rewriting rules are applied as long as possible after every move.

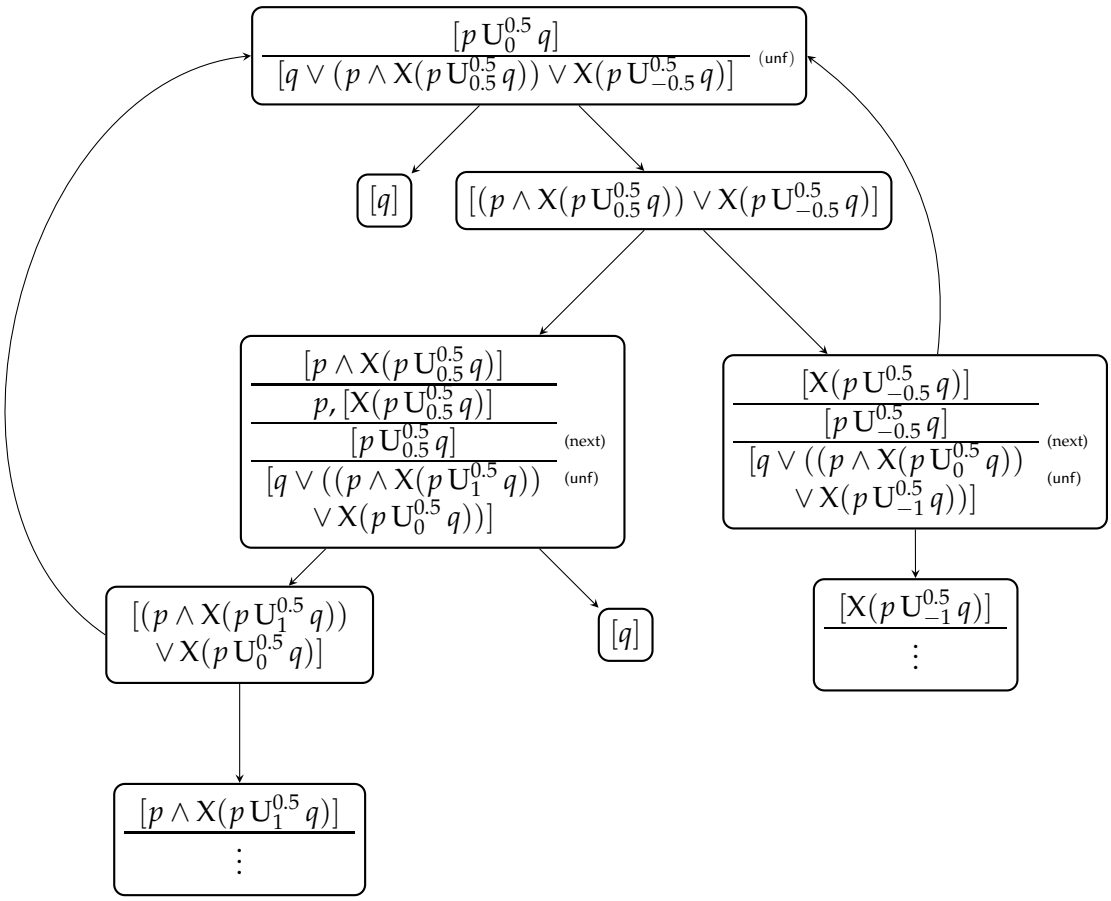


Figure 4.3: Example game graph for $\Phi = p\,\mathrm{U}^{\frac{1}{2}}\,q$. For better readability consecutive moves are condensed where order does not matter, similarly to omitting unneeded brackets. For example, there are actually only two outgoing edges from the initial node at the top and one of them has the two children that are bound directly to the top node. Also, Player A could choose to put the focus of $[p \wedge \mathrm{X}(p\,\mathrm{U}_{0.5}^{0.5}\,q)]$ to p in the next step, however, that would clearly not be optimal and we omitted this case in the figure.

Winning conditions for infinite plays

We can keep Definition 2.6 for winning plays. However, the strategy, to anticipate the outcome of an infinite play on the recurrence of a configuration is not exhaustive anymore. The state space is finite in standard focus games, thus on every potentially infinite play there must some configuration eventually be revisited. With the infinite state space for counter focus games we may not necessarily experience the recurrence of any configuration.

In the Example shown in Figure 4.3, we find the following infinite play on $\mathcal{G}(p\,\mathrm{U}^{0.5}\,q)$.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}[\mathsf{unf}(p\,\mathrm{U}^{0.5}_{0}\,q)] = \\ [q \vee (p \wedge \mathrm{X}(p\,\mathrm{U}^{0.5}_{0.5}\,q)) \vee \mathrm{X}(p\,\mathrm{U}^{0.5}_{-0.5}\,q)]\end{array}}{[(p \wedge \mathrm{X}(p\,\mathrm{U}^{0.5}_{0.5}\,q)) \vee \mathrm{X}(p\,\mathrm{U}^{0.5}_{-0.5}\,q)]}\,(\mathrm{E})}{[p \wedge \mathrm{X}(p\,\mathrm{U}^{0.5}_{0.5}\,q)]}\,(\mathrm{E})}{p, [\mathrm{X}(p\,\mathrm{U}^{0.5}_{0.5}\,q)]}\,(\mathrm{A})}{\begin{array}{c}[\mathsf{unf}(p\,\mathrm{U}^{0.5}_{0.5}\,q)] = \\ [q \vee ((p \wedge \mathrm{X}(p\,\mathrm{U}^{0.5}_{1}\,q)) \vee \mathrm{X}(p\,\mathrm{U}^{0.5}_{0}\,q))]\end{array}}\,(\mathrm{NEXT})}{[(p \wedge \mathrm{X}(p\,\mathrm{U}^{0.5}_{1}\,q)) \vee \mathrm{X}(p\,\mathrm{U}^{0.5}_{0}\,q)]}\,(\mathrm{E})}{[p \wedge \mathrm{X}(p\,\mathrm{U}^{0.5}_{1}\,q)]}\,(\mathrm{E})}{p, [\mathrm{X}(p\,\mathrm{U}^{0.5}_{1}\,q)]}\,(\mathrm{A})}{\begin{array}{c}[\mathsf{unf}(p\,\mathrm{U}^{0.5}_{1}\,q)] \\ \vdots\end{array}}\,(\mathrm{NEXT})}{\begin{array}{c}[\mathsf{unf}(p\,\mathrm{U}^{0.5}_{1.5}\,q)] \\ \vdots\end{array}}\,(\mathrm{NEXT})
$$

Ignoring the counters, the standard winning conditions (Definition 2.6) would classify the play above lost, already after the first application of the next rule. As argued earlier, this is reasonable in the absence of counters because if Player E had a chance to force A to move the focus away from the until formula, this would have been possible already. Taking the counters into account the configurations are not equal anymore, the formulae are biased differently. In the example above the fact that the until counter increases does not help Player E to fulfill the eventuality and the play is supposed to be lost either way.

However, in general we can not conclude directly whether or not Player E will have choices in the future that she did not have already, due to different counter values. Consider a second example $(p\,\mathrm{U}^{0.6}\,q) \wedge (\neg p \wedge \neg q)$. A possible play on that formula is

the following.

$$
\cfrac{
\cfrac{
\cfrac{
\cfrac{
\cfrac{
\cfrac{
\cfrac{
\cfrac{
\cfrac{
\cfrac{
\cfrac{
\cfrac{
\cfrac{
[p\,\mathrm{U}_0^{0.6}\,q \wedge (\overline{p} \wedge \overline{q})]
}{[\mathsf{unf}(p\,\mathrm{U}_0^{0.6}\,q)], \overline{p} \wedge \overline{q})]}\ (\textsc{a})
}{[\mathsf{unf}(p\,\mathrm{U}_0^{0.6}\,q)], \overline{p}, \overline{q}) =}\ (\textsc{rewrite})
}{[(p \wedge \mathrm{X}(p\,\mathrm{U}_{0.4}^{0.6}\,q)) \vee \mathrm{X}(p\,\mathrm{U}_{-0.6}^{0.6}\,q)], \overline{p}, \overline{q}}\ (\textsc{e})
}{[\mathrm{X}(p\,\mathrm{U}_{-0.6}^{0.6}\,q)], \overline{p}, \overline{q}}\ (\textsc{next})
}{[\mathsf{unf}(p\,\mathrm{U}_{-0.6}^{0.6}\,q)] =}
}{[(p \wedge \mathrm{X}(p\,\mathrm{U}_{-0.2}^{0.6}\,q)) \vee \mathrm{X}(p\,\mathrm{U}_{-1.2}^{0.6}\,q)]}\ (\textsc{e})
}{[p \wedge \mathrm{X}(p\,\mathrm{U}_{-0.2}^{0.6}\,q)]}\ (\textsc{a})
}{p, [\mathrm{X}(p\,\mathrm{U}_{-0.2}^{0.6}\,q)]}\ (\textsc{next})\ {}^{*}
}{[\mathsf{unf}(p\,\mathrm{U}_{-0.2}^{0.6}\,q)] =}
}{[(p \wedge \mathrm{X}(p\,\mathrm{U}_{0.2}^{0.6}\,q)) \vee \mathrm{X}(p\,\mathrm{U}_{-0.8}^{0.6}\,q)]}\ (\textsc{e})
}{[p \wedge \mathrm{X}(p\,\mathrm{U}_{0.2}^{0.6}\,q)]}\ (\textsc{a})
}{p, [\mathrm{X}(p\,\mathrm{U}_{0.2}^{0.6}\,q)]}\ (\textsc{next})\ \textsc{revisited}\ {}^{*}
}{[\mathsf{unf}(p\,\mathrm{U}_{0.2}^{0.6}\,q)] =}
}{[q \vee ((p \wedge \mathrm{X}(p\,\mathrm{U}_{0.6}^{0.6}\,q)) \vee \mathrm{X}(p\,\mathrm{U}_{-0.4}^{0.6}\,q))]}\ (\textsc{e})
}{[q]}
$$

Player E should obviously win even though there was repeated position in between and A never changed the focus. In that play, E profited from increasing the counter to a certain amount and was only then able to fulfill the eventuality q. Obviously, disallowing p and q even on the second position, i.e. adding $\wedge\, \mathrm{X}(\neg p \wedge \neg q)$ to the initial formula forces E to play even more "rounds" until she reaches a positive counter value and is able to choose q.

Therefore we can not just disregard the counters but add additional conditions on infinite plays. These need to incorporate that all until formulae have to be fulfilled eventually.

Definition 4.3 (Winning conditions)**.** Let Φ be an $fLTL$ formula and $P = C_0C_1\ldots$ a play on the game $\mathcal{G}(\Phi)$. The play P is won by Player E or A if it is won according to the respective winning conditions of standard focus games in Definition 2.6.

Additionally, Player E wins an infinite play without recurring configuration, if and only if

- Player A changes the focus always eventually or
- a position of the form $[\mathrm{X}(\varphi\,\mathrm{R}_k^c\,\psi)], \Gamma$ is visited always eventually.

Player A wins an infinite play P without recurrence if and only if

- there is a configuration $C_n = [\mathrm{X}(\varphi\,\mathrm{U}_k^c\,\psi)], \Gamma$ such that the change rule is never applied again from that point on.

With the additional conditions focus games with counters also have a unique winner. For finite plays this is already established in standard focus games. On infinite plays, either the focus changes always eventually or it does not. In the first case E wins and A looses consistently. If the focus is eventually never moved again it must reach a formula that re-spawns itself, i.e. an until or release formula and stay on it when being unfolded. Otherwise, at some point, the formula would be broken down to propositions which leaves no choice to A but to move the focus. If it stays on an (possibly unfolded) until formula $\varphi \,\mathrm{U}^c_k\, \psi$ it cannot reach something else but

- $((\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}^c_{k'}\, \psi)) \vee \mathrm{X}(\varphi \,\mathrm{U}^c_{k''}\, \psi))$,
- $(\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}^c_{k'}\, \psi)) \vee \mathrm{X}(\varphi \,\mathrm{U}^c_{k''}\, \psi)$,
- $\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}^c_{k'}\, \psi)$ or
- $\mathrm{X}(\varphi \,\mathrm{U}^c_{k''}\, \psi)$.

In particular the focus can not be on a release formula. Therefore A wins and E does not. The argument is the same for release formulae where E wins and A looses because no until will ever be in focus anymore but the release formula will acquire the focus again and again.

We observe that the optimal strategy for A from standard focus games is still optimal in the counter setting.

Theorem 4.2. *Player E has a winning strategy for a $\mathcal{G}(\Phi)$ on an fLTL formula Φ if and only if Φ is satisfiable.*

Theorem 4.2 follows by the same construction as used to prove Theorem 2.1. For a none-repeating winning play for E against A's optimal strategy, a word model is constructed in the same manner and does not even need to be extended by repeating some suffix since the play is infinite already.

4.3 ω-abstraction

The approach of using focus games for checking satisfiability relies on the fact that the length of plays is bounded. With counters, even if an U-formula carries the focus that did not change during a "loop" the game might not be lost, as we saw in the example $p \,\mathrm{U}^{0.6}\, q \wedge (\neg p \wedge \neg q)$.

In general, such a loop, i.e. repeating similar configurations several times, may belong to Player E's strategy. The following approach distinguishes "useful" loops from "useless" in many cases which allows us to break down infinite plays to finite ones.

The standard winning condition already uses such an anticipation. The actual play, according to the rules, on a formula that is unsatisfiable because of an unsatisfiable

eventuality, e.g. $\top \mathrm{U} \bot$, would last forever. However, the winning condition makes use of the fact, that both players can be assumed to play optimal. The assumption is that, if E was not able to fulfil an eventuality, and the play came back to an exact same point, the next "round" will not yield something new as E is assumed to have played the best she can already. In other words if she were able to fulfil the formula, she could have done that already. The actual infinite play is cut off as soon as E can not profit from playing any longer. The same argument applies for A, who, once forced to changed the focus or to put it on a release, will not be able to avoid that in a next loop.

These conclusions can be made after observing that a configuration was repeated exactly. The goal in this section is to develop arguments for classifying a play lost or won, if repeated configurations are not exactly equal, but similar.

We saw that a game can require to loop several times through configurations that differ only by their counters in order to be won. However it might happen as well, that even playing arbitrary long does not help E to win. In the unfolding

$$\mathsf{unf}(\top \mathrm{U}^{0.5}_{k} \bot) = \bot \vee (\top \wedge \mathrm{X}(\top \mathrm{U}^{0.5}_{k+1-c} \bot)) \vee \mathrm{X}(\top \mathrm{U}^{0.5}_{k-c} \bot),$$

E can always choose the conjunction $\top \wedge \mathrm{X}(\top \mathrm{U}^{0.5}_{k+1-c} \bot)$ and increase the counter arbitrarily but has no advantage, i.e. even with an arbitrary high counter value she will still not be able to satisfy $\bot$.

In any case, by revisiting the a configuration, but with a higher counter value, E has proven, that she can increase the counter value arbitrarily by just playing long enough. To figure, if she can win the play we can grant an arbitrary high value, which we symbolically may denote ω, and let the play go on. Now, that particular counter can not change anymore. Setting the counter to ω does not enable E to do anything that were not possible by just playing sufficiently long. On the other hand, if E does not quit the loop (nor force a focus change) with that value, she will obviously never be able to fulfil the obligation imposed by an U-formula.

Let $P = C_0 C_1 \ldots$ be a run on a game $\mathcal{G}(\Phi)$ and C_m, C_n configurations such that $m < n$ and $C_m = C_n[\varphi \mathrm{U}^{c}_{k+\varepsilon} \psi \;/\; \varphi \mathrm{U}^{c}_{k} \psi]$. By $C[\varphi'/\varphi]$ we mean that the *present* formula φ, i.e. not any sub-formula, is replaced by φ'. The configurations C_m and C_n are not equal, still we consider them as repetition and consider ε as the *gain* of a partial play from C_m to C_n.

Even though repeating configurations with negative gain may occur we only need to consider positive abstraction, that is setting counters to ω for a positive gain. This abstraction is an over-approximation for until formulae and an under-approximation for release formulae (as follows e.g. also from the implications of Lemma 4.2).

When an until formula is in focus, has not been fulfilled since C_m and did not even gain credit, there is no hope that E can win because having a smaller counter value gives E rather less but never more options in the game. Assuming E played optimal this means E can neither fulfil the until now nor later.

ω-abstraction

$$\frac{[\mathrm{X}(\xi_1 \,\mathrm{U}_k\, \xi_2)], \{\mathrm{X}(\xi_1 \,\mathrm{U}_k\, \xi_2), \mathrm{X}\,\varphi_2, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}}{[\mathrm{X}(\xi_1 \,\mathrm{U}_\omega\, \xi_2)], \{\mathrm{X}(\xi_1 \,\mathrm{U}_\omega\, \xi_2), \mathrm{X}\,\varphi_2, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}}\ (\textsc{next})^\omega$$

$$\frac{[\xi], \{\mathrm{X}(\xi_1 \,\mathrm{U}_k\, \xi_2), \mathrm{X}\,\varphi_2, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}}{[\xi], \{\mathrm{X}(\xi_1 \,\mathrm{U}_\omega\, \xi_2), \mathrm{X}\,\varphi_2, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}}\ (\textsc{next})^\omega$$

$$\frac{[\mathrm{X}(\xi_1 \,\mathrm{R}_k\, \xi_2)], \{\mathrm{X}(\xi_1 \,\mathrm{R}_k\, \xi_2), \mathrm{X}\,\varphi_2, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}}{[\mathrm{X}(\xi_1 \,\mathrm{R}_\omega\, \xi_2)], \{\mathrm{X}(\xi_1 \,\mathrm{R}_\omega\, \xi_2), \mathrm{X}\,\varphi_2, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}}\ (\textsc{next})^\omega$$

$$\frac{[\xi], \{\mathrm{X}(\xi_1 \,\mathrm{R}_k\, \xi_2), \mathrm{X}\,\varphi_2, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}}{[\xi], \{\mathrm{X}(\xi_1 \,\mathrm{R}_\omega\, \xi_2), \mathrm{X}\,\varphi_2, \ldots, \mathrm{X}\,\varphi_n, q_1, \ldots, q_m\}}\ (\textsc{next})^\omega$$

Figure 4.4: Rules for approximating a counter value by ω. Their application can, under certain conditions, yield a conclusive verdict about the winner of an infinite play by folding the play to a finite one.

Dually for a release formula: The focus does not play an important role here, just like in standard focus games, since they do not need to be fulfilled eventually. A release formula in a recurring position of a play was at least not violated and we only need to check if this will not happen in the future either. As long as the respective counter does not increase, Player E can repeat the strategy she was using before, in particular not violating the formula. Only, if the counter value increased during the loop, there is doubt that E will be able to satisfy such a formula in the future, in particular when the counter changes from negative to zero or positive which changes the unfolding and imposes an additional obligation. Player E can avoid to satisfy the sub-formula ψ of $\varphi \,\mathrm{R}^c_k\, \psi$ as long as k is negative. But once k is zero or positive, E has to prove that ψ can be satisfied. Therefore we set $k = \omega$ and let the play go on which leads either to the termination of the game by a purely propositional formula or to another repetition of the configuration. This approximation is reflected by the rules form Figure 4.4 which shall be applied in a repeated position C_n if the gain is $\varepsilon > 0$.

Up to now, we only considered plays where a configurations was repeated and at most one counter value changed. Let us have a look at a play, where two counters change during a loop.

For two or even more until formulae, the same argument applies as for one changing counter, as long as all of them gain credit during a loop. If E needs to play a certain number of rounds until a counter value is sufficiently high, it does not harm, to play

even longer in order to get a higher value for another counter. Thus we can assume an arbitrary high value on any of them. Similar considerations apply for a set of release formulae. For a positive gain on some release formulae, E needs to prove that she can satisfy the obligation for an arbitrary time for all of them.

We can apply the ω-abstraction or conclude directly as long as all counters either did not decrease or increase, respectively whereas, for formulae of the form

$$\Phi = (\varphi_1 \,\mathrm{U}_{k_1}\, \psi_1) \wedge (\varphi_2 \,\mathrm{U}_{k_2}\, \psi_2)$$

and a play $P = C_1 C_2 \dots C_i \dots C_j \dots$ on $\mathcal{G}(\Phi)$ with

$$\begin{aligned} &\mathrm{X}(\varphi_1 \,\mathrm{U}^c_{k_1}\, \psi_1), \mathrm{X}(\varphi_2 \,\mathrm{U}^c_{k_2}\, \psi_2) \in C_i \quad \text{and} \\ &\mathrm{X}(\varphi_1 \,\mathrm{U}^c_{k'_1}\, \psi_1), \mathrm{X}(\varphi_2 \,\mathrm{U}^c_{k'_2}\, \psi_2) \in C_j \end{aligned}$$

where $k'_1 > k_1$ and $k'_2 < k_2$ all approximations are biased. Similarly, for a formula

$$\Psi = (\varphi_1 \,\mathrm{R}_{k_1}\, \psi_1) \wedge (\varphi_2 \,\mathrm{R}_{k_2}\, \psi_2).$$

There are several possibilities to approximate such plays but only some of them might allow a conclusion. In general, we can tell that E can not win a play if she looses even an over-approximation and certainly wins if she can do that in an under-approximation.

Consider the former case where we set k_1 to ω and leave the value of k_2 untouched. This is clearly optimistic for the until formula Φ. If Player E looses this approximation the game is certainly lost, but if she wins the reason could be the advantage of reaching the arbitrary high counter value on counter k_1 without having to trade off counter k_2. If there is an actual value k_1^*, that allows E to satisfy the according until formula, that value might not be reachable without decreasing k_2 too much for winning the play in the end. To exemplify this case, consider the formula

$$(p \,\mathrm{U}^{0.6}_{-1.8}\, q) \wedge (r \,\mathrm{U}^{0.5}_{1.5} \,\mathrm{G}(\neg q)) \wedge \mathrm{G}(\neg r)$$

with an illustration of a play on it shown in Figure 4.5.

Remark. Presetting counters is done for better readability in the examples. We may construct a formula, e.g.

$$\neg q \wedge \neg p \wedge \mathrm{X}\neg p \wedge \mathrm{X}\mathrm{X}\neg p \wedge \mathrm{X}\mathrm{X}\mathrm{X}(\mathrm{G}\neg r) \wedge p \,\mathrm{U}^{0,6}_0\, q \wedge r \,\mathrm{U}^{0.5}_0 (G\neg q),$$

which leads to the configuration

$$(p \,\mathrm{U}^{0.6}_{-1.8}\, q), (r \,\mathrm{U}^{0.5}_{1.5} \,\mathrm{G}(\neg q)), \mathrm{G}(\neg r)$$

after three application of the next-rule. The formula has, formally, a different semantics in terms of its models, however, this is irrelevant for the considerations made here.

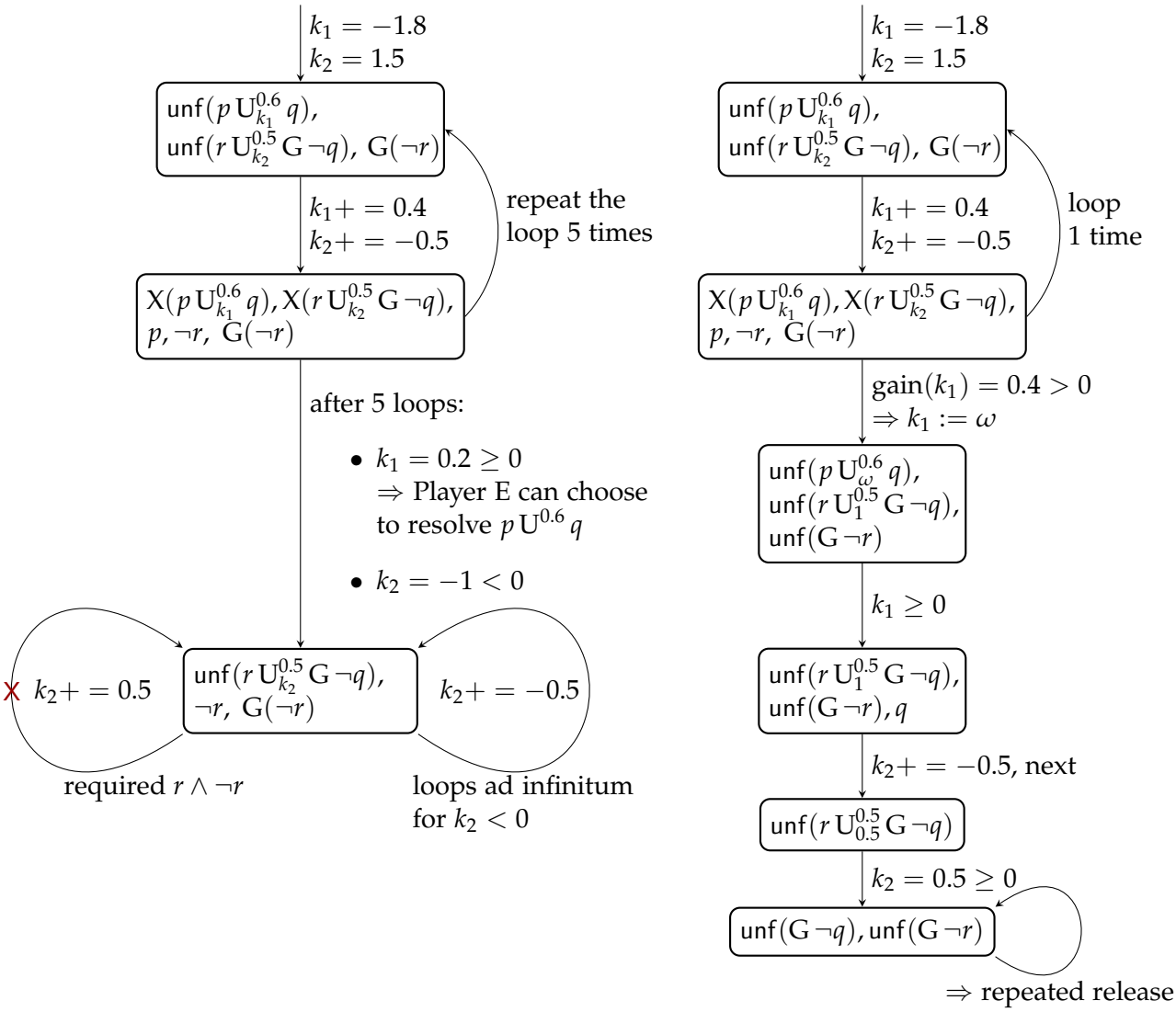


Figure 4.5: The formula $(p\,\mathrm{U}^{0.6}_{-1.8}\,q) \wedge (r\,\mathrm{U}^{0.5}_{1.5}\,\mathrm{G}(\neg q)) \wedge \mathrm{G}(\neg r)$ is not correctly approximated by the ω-rules. A sketch of a play is shown: (l) In order to resolve the first until, E has to play the loop five times but that results in a negative value for k_2. Since $\neg r$ needs to hold globally, E cannot gain credit for the second until any more and can not win the game. The approximation (r) would be done after looping once, allowing E to resolve the first until within two steps. Then k_2 is still positive and can also be resolved.

The assumption of setting the increasing counter to ω is pessimistic for the release formula Ψ because it obligates E to globally satisfy ψ_1. In particular, k_1 could have been negative and became instantly positive by setting it to ω. k_2 did not have the time to decrease far enough which might have been possible if k_1 increased slowly enough. See Figure 4.6 for an example in which E would win whereas the approximation yield a negative and thus wrong anticipation.

As can be seen from these examples, the ω-abstraction is not necessarily conclusive on loops with diverging counters, i.e. where the individual gain on one counter is positive whereas the gain on another counter is negative. The reason is that, in general, E can trade one counter for another which might be leading to a winning play but does not have to.

For a conjunction of until formulae, such as Φ, the apparent alternative of setting $k_2 := -\omega$ always results in a lost play for E, no matter what k_1 is set to. E has no chance to win on an until formula with that value since the counted unfolding only allows to fulfil the eventuality in presence of a none-negative counter value.

Considering the conjunction Ψ of release formulae, any release formula with a negative, infinite counter value can never be violated. In particular, since E did not loose with counters k_1 and k_2 she will not loose the game when setting k_2 to $-\omega$ either. Therefore this approximations yields no further information.

Setting k_1 to ω and k_2 to $-\omega$ also leads to no conclusion. If E wins under that assumption the actual game might have been lost because k_2 was actually always too large to win. If E lost, on the other hand, that does not mean the game needed to be lost, it is possible that k_1 would not increase as quickly, allowing k_2 to decrease to a certain amount that allows k_1 to not increase any more.

The overall observation is so far, that in case of diverging counters, the ω-abstraction does not provide good arguments to anticipate the outcome of an infinite game. That also applies to combinations of until and release formulae.

Only if all changing counters accord, the abstraction can be conclusive: Whenever all changing counters on release formulae decrease and all changing counters on until formulae increase, abstracting them by $-\omega$ and ω, respectively anticipates the outcome of the game correctly. In the dual case, where all R-counters increase and all U-counters decrease the approximation is not even necessary since, following the implication lemma there is no hope that the until formulae can ever be resolved.

If R- and U-counters decrease, the only thing we can check is whether the E profits from setting all R-counters to $-\omega$. Then, only if all U-counters increase a second approximation by ω is justified. Given this leads to resolving all untils, we can conclude that E has a winning strategy. In all other cases we stay inconclusive. Also, the case of a conjunction of R- and U-formulae with increasing counters k_r and k_u, respectively, is similar to the case of having two until formulae with diverging counters. Only if E is winning by setting k_r to ω and leaving k_u, and then a further approximation setting

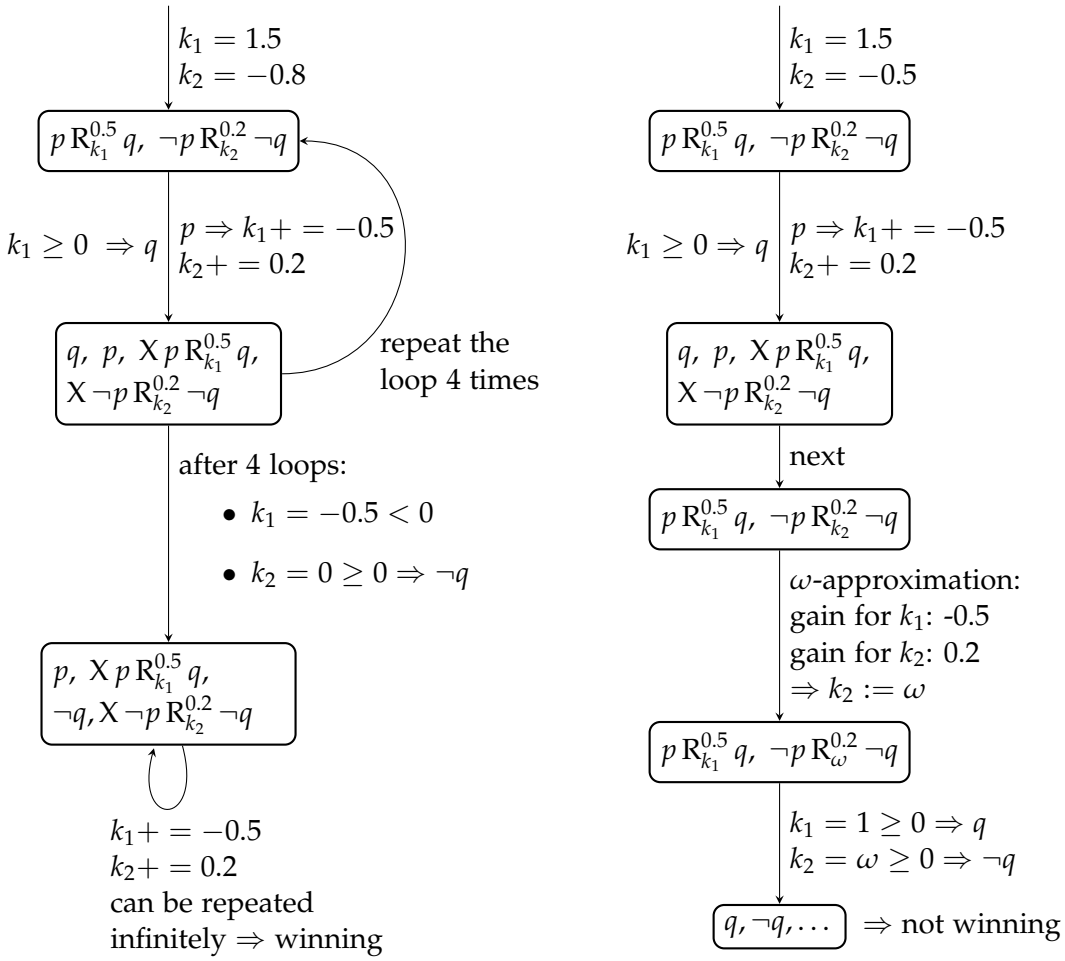


Figure 4.6: The figure sketches a play on the formula $p\,\mathrm{R}^{0.5}_{1.5}\,q \wedge \neg p\,\mathrm{R}^{0.2}_{-0.8}\,\neg q$ which would be won by E (l) but is approximated too pessimistic by the ω-rules (r).

k_u to ω is equally successful we can conclude that E is indeed winning. Setting only k_u to ω is optimistic, hence if E looses she has no chance to win on the actual game either. As in previous cases, we do not gain information if E is winning.

The last option is to set both, k_r and k_u to ω which is inherently inconclusive, just as setting ω and $-\omega$ in the case of two release formulae, one increasing and one decreasing, respectively.

The apparent difficulties arising from the combinatorics with several changing counters motivates the definition of an $fLTL$ fragment of formulae with at most one *single* counter. Within this fragment we can assure, the ω-approximation can always be applied conclusively and is hence decidable. Unfortunately this fragment is too weak to keep up Theorem 3.1 where we used a conjunction of several until formulae with annotated frequencies.

4.4 Systems of linear inequalities

We describe the idea of reducing the existence of a winning strategy for Player E to the solution of some equation system. Examples are given in order to explain the construction. Secondly the drawbacks of this approach are outlined and motivate the definition of a fragment of $fLTL$ that assures applicability.

The ω-approximation failed on the example

$$(p \,\mathrm{U}^{0.6}_{-1.8}\, q) \wedge (r \,\mathrm{U}^{0.5}_{1.5}\, \mathrm{G}(\neg q)) \wedge \mathrm{G}(\neg r),$$

as shown in Figure 4.5, because it directly considers the limit. In order to overcome this problem we consider a more sensitive approach to reason about loops, i.e. circles in the game graph.

We will start describing the idea of reducing the existence of a winning strategy for Player E to the solution of some equation system. Examples are given in order to explain the construction. Secondly the drawbacks of this approach are outlined and motivate the definition of a fragment of $fLTL$ that assures applicability. Based on this fragment, the approach is formalized at the end of this section.

In contrast to setting a counter value to an arbitrary high value because of a possible loop, we want to calculate whether there is a number n such that traversing the loop for n times leads to a qualitative change of the situation, i.e. it gives E either new options or imposes new restrictions.

In the example, traversing the loop more than three times took Player E's option to choose $\mathrm{G}\,\neg q$ in the unfolding of the second until formula, i.e. to resolve it, since this leads to a negative counter value. On the other hand, four times is not enough for the other counter value to get positive. Therefore E has no chance to satisfy the eventuality q *before* the second counter gets negative.

Let us denote the counter of the two until formulae by k_1 and k_2, respectively. The loop causes k_2 to decrease by $-c_2$, where $c_2 = 0.5$, $c_1 = 0.6$ are the frequencies of the respective until operators. Assuming for the moment, that the loop is the only possible way of playing the game, k_2 is decreased by c_2 once after the first until was resolved and additionally as often as the loop was traversed before. Hence, we need to ensure that

$$k_2 - c_2 + n \cdot (-c_2) \geq 0.$$

The loop also effected an increase of k_1 by $1 - c_1$ and, in order to resolve the first until, it is required that

$$k_1 + n \cdot (1 - c_1) \geq 0.$$

Finding a natural solution, i.e. a number $n \in \mathbb{N}$, such that both inequalities hold, yields directly a winning play for E. Furthermore, if there is no solution, E cannot win by only playing the considered loops. For $k_1 = -1.8$, $k_2 = 1.5$ we observe directly that there is no natural solution. If we are able to reason about all loops of a game graph in such a manner, we can conclude if the formula is satisfiable or not.

Consider a game $\mathcal{G}(\Phi)$. We introduce a folded representation of the game graph where explicit counter values are substituted by unique variables. Instead of calculating the unfolding $\mathsf{unf}(\varphi \,\mathrm{U}_k^c\, \psi)$ for $c \neq 1$ directly, we add three edges to represent the choice of Player E. The variable k does not change and we always have a subsequent configuration where the formula is resolved, i.e. where E chose ψ. But we do add annotations to the edges: There is a constraint on the edge of the form $k \leq 0$ whenever a formula is resolved. When E chooses one of the other two options we label the edge with the respective modification of k: For $\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}_k^c\, \psi)$ we use a label such as $k+ = 1 - c$ and for only $\mathrm{X}(\varphi \,\mathrm{U}_k^c\, \psi)$ the label is $k+ = -c$. See Figure 4.7 for a simple example.

As can be seen in the example, choices of Player A can blow up the graph artificially, since A can potentially choose to focus e.g. on p in the example but would again have to change the focus immediately.

To analyse the game graph it makes sense to reduce it as far as possible without loosing important information. We can reduce $\wedge$-nodes by incorporating an optimal strategy for A in the game graph. That restricts A to decisions according to that strategy which, however, does not make him weaker. Unfortunately, the optimal strategy we considered in Theorems 2.1 and 4.2 is not *positional*, i.e. the decision at a certain point in a play may depend on more then just the last configuration. Imposing a non-positional strategy on a game is possible but results in general in an exponentially blown-up game graph, since it is, from an automata-theoretical point of view, the elimination of $\wedge$-nodes and hence related to a subset construction. In [DL05], Dax and Lange give a construction for a pure positional strategy using multiple foci in each configuration, which basically stores the queue maintained by A in every configuration.

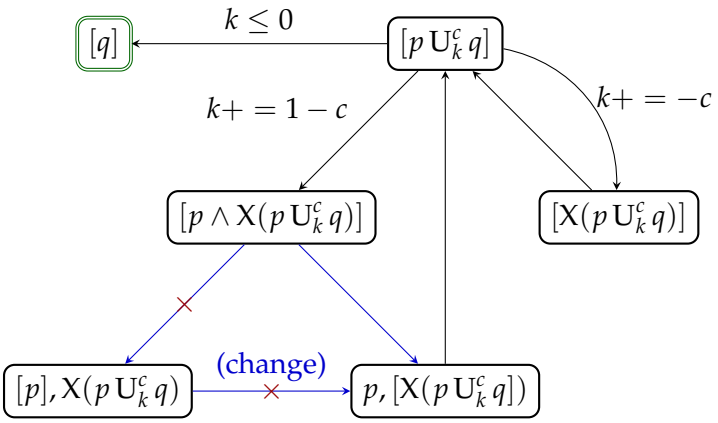


Figure 4.7: Folded representation of a game $\mathcal{G}(p\,\mathrm{U}^c\,q)$ where counter changes and constrains are represented by labels on edges. Choices of Player A are printed in blue.

For the following considerations it suffices to only use the positional parts of the strategy, removing some of A's choices. By the optimality of the strategy described, e.g. in the proof for Theorem 2.1, we can restrict Player A to obey the following rule: He is supposed to always choose the right-hand side of an $\wedge$-formula to inherit the focus, unless it does not contain an until operator while the left-hand side does. In particular, A then keeps the focus on an U-operator as long as possible because the formula reappears with an additional X on the right-hand side in the unfolding. Considering the example from above, this leads to the game graph in Figure 4.7 where the red crosses indicate pruned components of the graph.

In the next step we can identify *winning nodes* and *lost nodes*. These are the nodes, that directly yield a winner for a play reaching that node. That is the case if they either contain a propositional contradiction, i.e. $\bot$ or both, q and $\overline{q}$ for $q \in AP$, or are a consistent subset of AP.

Additionally we can decide the winner for all nodes that do not contain a formula with frequency annotated operator. These configurations represent pure LTL formulae for which we can compute a winner as described in section 2.3.1. Winning nodes play a role in the analysis of the graph while lost nodes can be cut off.

Figure 4.7 shows a game graph for the formula $p\,\mathrm{U}^c\,q$. There is a restricted edge from the initial to the final node ($[q]$) and any path must obey that restriction. Along the path modifications are applied to to the initial counter value. The combination of all of these modifications before the restricted edge is crossed, must yield a value that meets the condition. If that is not possible, the final node is not reachable and the game can therefore not be won.

Any path to the restricted edge is composed of the *direct path* (i.e. without duplicate

nodes) and a number of traversals for each loop possible along that direct path. In this example the direct path consists only of the node ($[p\,\mathrm{U}^{c}_{k}\,q]$) and does thus not influence the value of k itself. Along that path we have the possibility to traverse two different loops which add $1-c$ and $-c$ to k, respectively. Therefore, satisfiability for that formula can be described by the inequality

$$k + n_1 \cdot (1-c) + n_2 \cdot (-c) \geq 0.$$

For any initial value of k, we can find a natural solution and hence a path that forms a winning play in the game for E.

In general, there can be several winning nodes, and several direct paths to such a node. Consider the formula $p_1\,\mathrm{U}^{c_1}_{k_1}\,q_1 \wedge p_2\,\mathrm{U}^{c_2}_{k_2}\,q_2$ with two until sub-formulae. The folded graph is shown partially in Figure 4.8. Note that there are no nodes in which Player A can make a choice because of the restrictions introduced above.

Even though the satisfiability is obvious here let us demonstrate how an equation system can be constructed. For every winning node, consider the direct paths starting at the initial one.

For the node N_2 the direct path ϱ_1 consists of only three nodes:

$$\varrho_1 = (p_1\,\mathrm{U}^{c_1}_{k_1}\,q_1, p_2\,\mathrm{U}^{c_2}_{k_2}\,q_2) \xrightarrow{k_1 \geq 0} (q_1, p_2\,\mathrm{U}^{c_2}_{k_2}\,q_2) \xrightarrow{k_2 \geq 0} (q_1, q_2)$$

There are eight loops from N_0 to itself. Unfolding the first until formula gives E two choices. In order to get back to node N_0, the second formula must also not be resolved and E has, again the two choices of gaining or investing credit. The other four loops are basically the same but with the opposite order, unfolding the second until formula first. Therefore we have the combinations of gaining or investing credit per formula. Formally we want to consider the labels that indicate a modification of some counter as mapping. The weight $\mathsf{weight}_k(\varrho)$ of a path ϱ with respect to a counter k is then the concatenation of the mappings along that path, that modify k. For example the loop

$$l_1 = N_0 \xrightarrow{k_1 += 1-c_1} N_8 \xrightarrow{k_2 += -c_2} N_1 0 \to N_0$$

is labelled by the mapping

$$\mathsf{weight}_{k_1}(l_1) : x \mapsto x + 1 - c_1 \text{ for } k_1 \text{ and}$$
$$\mathsf{weight}_{k_2}(l_1) : x \mapsto x + 1 - c_2 \text{ for } k_2.$$

For the seven other loops we obtain similar mappings for each counter. The second node on the path can not be reached from itself. Therefore it imposes no further inequalities. Since each of the mappings here only add some *constant value* to the counter value we add them all up, weighted by the number of traversals of the respective loop and obtain the following conditions for winning the game via N_2.

$$k_1 + n_1 \cdot \mathsf{weight}_{k_1}(l_1) + \ldots + n_8 \cdot \mathsf{weight}_{k_1}(l_8) \geq 0$$
$$k_2 + n_1 \cdot \mathsf{weight}_{k_2}(l_1) + \ldots + n_8 \cdot \mathsf{weight}_{k_2}(l_8) \geq 0$$

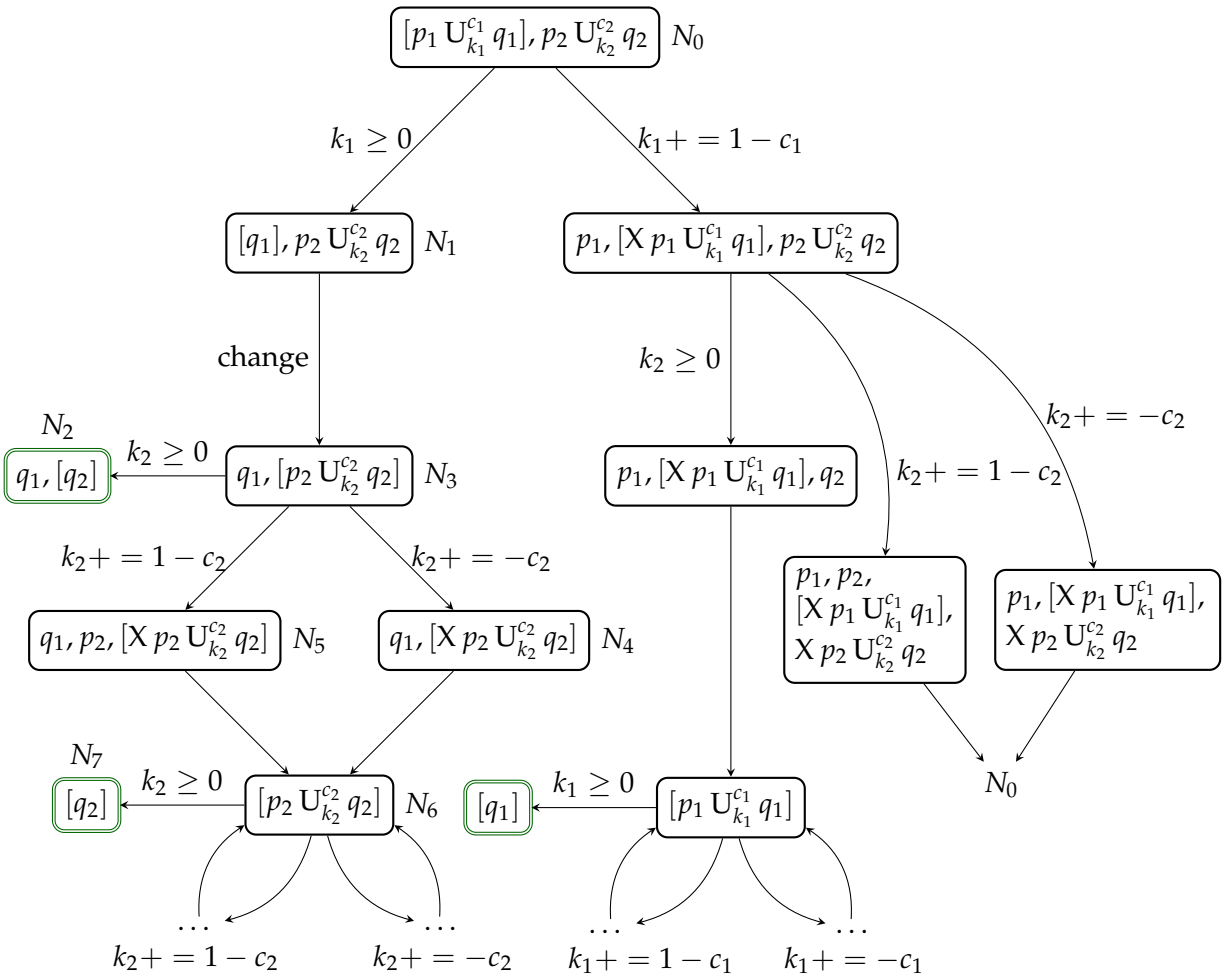


Figure 4.8: Folded graph of the formula $p_1 \,\mathrm{U}^{c_1}_{k_1}\, q_1 \wedge p_2 \,\mathrm{U}^{c_2}_{k_2}\, q_2$. Analogous parts have been omitted.

There are two direct paths to node N_7, one via N_5 and another via N_4. The first one adds to k_2 the value $1 - c_2$:

$$([p_1 \,\mathrm{U}^{c_1}_{k_1}\, q_1], p_2 \,\mathrm{U}^{c_2}_{k_2}\, q_2) \xrightarrow{k_1 \geq 0} (q_1, [p_2 \,\mathrm{U}^{c_2}_{k_2}\, q_2]) \xrightarrow{1-c_2}$$
$$(q_1, p_2, [\mathrm{X}(p_2 \,\mathrm{U}^{c_2}_{k_2}\, q_2)]) \xrightarrow{(next)} ([p_2 \,\mathrm{U}^{c_2}_{k_2}\, q_2]) \xrightarrow{k_2 \geq 0} ([q_2])$$

The second path to N_7 changes k_2 by $-c_2$:

$$([p_1 \,\mathrm{U}^{c_1}_{k_1}\, q_1], p_2 \,\mathrm{U}^{c_2}_{k_2}\, q_2) \xrightarrow{k_1 \geq 0} (q_1, [p_2 \,\mathrm{U}^{c_2}_{k_2}\, q_2]) \xrightarrow{-c_2}$$
$$(q_1, [\mathrm{X}(p_2 \,\mathrm{U}^{c_2}_{k_2}\, q_2)]) \xrightarrow{(next)} ([p_2 \,\mathrm{U}^{c_2}_{k_2}\, q_2]) \xrightarrow{k_2 \geq 0} ([q_2])$$

On these paths we find the loops which we already considered earlier from and to N_0 and additionally two loops starting at node N_6. One loop has a label $k_2{+}{=}\, 1 - c_2$, the other one $k_2{+}{=}\, -c_2$. We thus have two alternative systems of inequalities, one for every direct path to N_7, where any natural solution also yields at least one winning play.

For the first path to N_7 with label $k_2{+}{=}\, 1 - c_2$ we have

$$k_1 + n_1 \cdot \mathsf{weight}_{k_1}(l_1) + \ldots + n_8 \cdot \mathsf{weight}_{k_1}(l_8) \geq 0$$

$$\begin{aligned} k_2 + (1 - c_2) + n_1 \cdot \mathsf{weight}_{k_2}(l_1) + \ldots + n_8 \cdot \mathsf{weight}_{k_2}(l_8) \\ + n_9 \cdot \mathsf{weight}_{k_2}(l_9) + n_{10} \cdot \mathsf{weight}_{k_2}(l_{10}) \geq 0 \end{aligned}$$

and for the second path with label $k_2{+}{=}\, -c_2$

$$k_1 + n_1 \cdot \mathsf{weight}_{k_1}(l_1) + \ldots + n_8 \cdot \mathsf{weight}_{k_1}(l_8) \geq 0$$

$$\begin{aligned} k_2 + (-c_2) + n_1 \cdot \mathsf{weight}_{k_2}(l_1) + \ldots + n_8 \cdot \mathsf{weight}_{k_2}(l_8) \\ + n_9 \cdot \mathsf{weight}_{k_2}(l_9) + n_{10} \cdot \mathsf{weight}_{k_2}(l_{10}) \geq 0. \end{aligned}$$

We proceed by constructing these systems for every direct path to a winning node.

Reconsider the example

$$\Phi = (p \,\mathrm{U}^{0.6}_{-1.8}\, q) \wedge (r \,\mathrm{U}^{0.5}_{-1.5}\, \mathrm{G}(\neg q)) \wedge \mathrm{G}(\neg r).$$

from Section 4.3. While the ω-approximation failed (c.f. Figure 4.5) on that example we can construct a system of inequalities and verify that it has no solution. The folded game graph is shown in Figure 4.9.

We observe a single winning node N_6, reachable on two direct paths

$$\varrho_1 = N_0 \xrightarrow{-c_2} N_1 \xrightarrow{k_1 \geq 0} N_2 \to N_5 \xrightarrow{k_2 \geq 0} N_6,$$
$$\varrho_2 = N_0 \xrightarrow{k_1 \geq 0} N_3 \xrightarrow{(change)} N_4 \xrightarrow{-c_2} N_2 \to N_5 \xrightarrow{k_2 \geq 0} N_6.$$

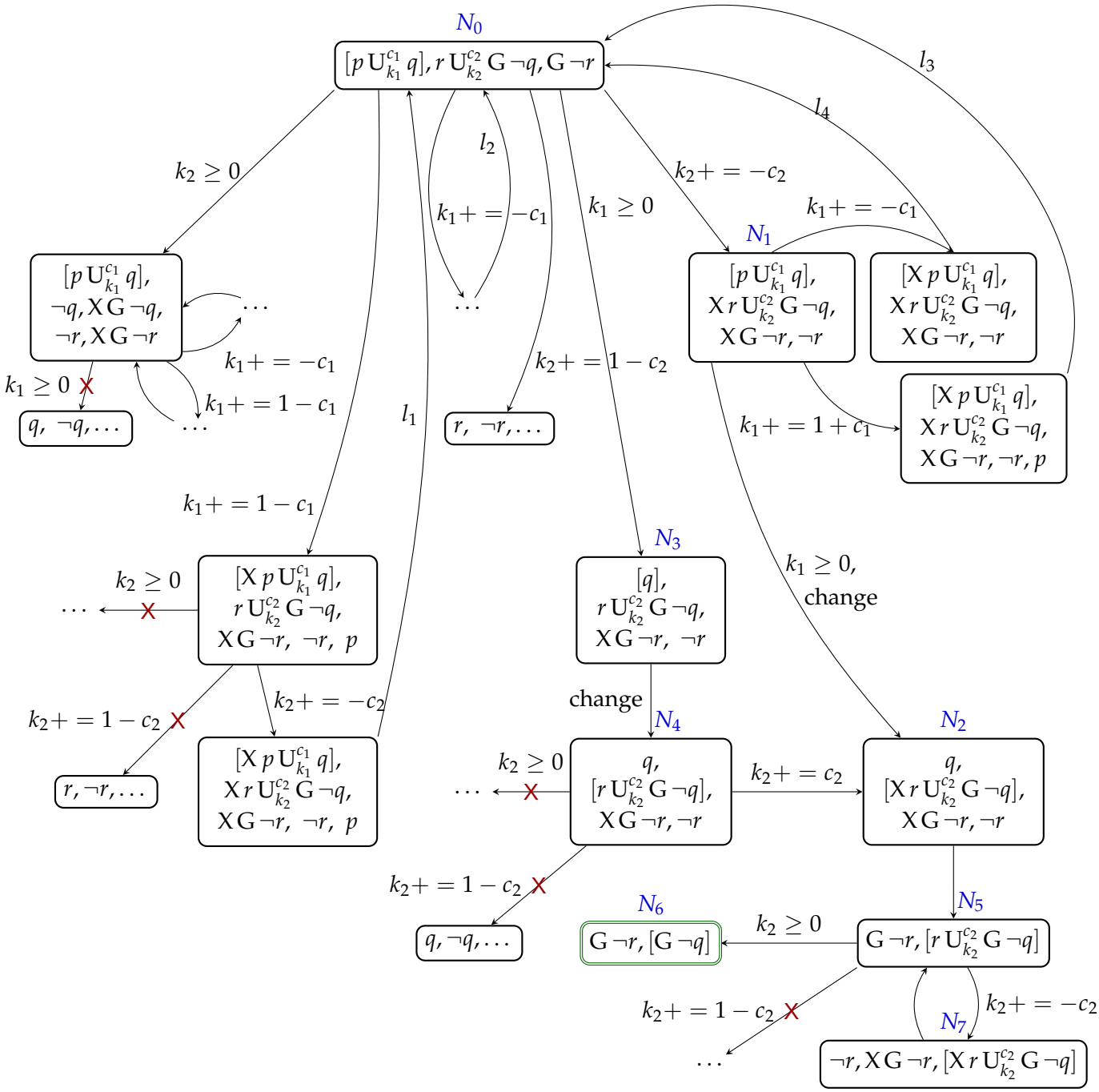


Figure 4.9: A folded version of the game graph on the formula $(p\,\mathrm{U}^{0.6}_{-1.8}\,q) \wedge (r\,\mathrm{U}^{0.5}_{-1.5}\,\mathrm{G}(\neg q)) \wedge \mathrm{G}(\neg r)$. Crosses on edges indicate that they lead to nodes where Player E would loose. The loop l_2 is very similar to l_1 and has been omitted.

Node N_0 is reachable from itself by four different paths l_1 to l_4, labelled by the mappings

$$\begin{aligned}\mathsf{weight}_{k_1}(l_1), \mathsf{weight}_{k_1}(l_3) &: k \mapsto (k + 1 - c_1)\\ \mathsf{weight}_{k_1}(l_2), \mathsf{weight}_{k_1}(l_4) &: k \mapsto (k - c_1)\end{aligned}$$

for counter k_1 of the first until formula $p \,\mathrm{U}_{k_1}^{c_1}\, q$ and

$$\left.\begin{matrix}\mathsf{weight}_{k_2}(l_1), \mathsf{weight}_{k_2}(l_3),\\ \mathsf{weight}_{k_2}(l_2), \mathsf{weight}_{k_2}(l_4)\end{matrix}\right\} : \quad k \mapsto (k - c_2)$$

For N_5 there is another loop

$$l_5 = N_5 \xrightarrow{k_2 += -c_2} N_7 \to N_5$$

with $\mathsf{weight}_{k_2}(l_5) : k \mapsto k - c_2$ and no effect on k_1.

There are two extra loops on node N_1, however they coincide with the ones form N_0 and are thus already considered.

Any valid path including the edge (N_0, N_3) has to obey the restriction that the counter value k_1 must be greater than or equal to zero at that point. The weight of the path that before that edge, applied to the initial value of k_1, must meet this condition. Since the weights of the loops are additive, we can decompose any valid path in the weight of the direct path to the node before the restricted edge and the traversals of the loops.

We let ϱ_1' denote the path ϱ_1 up to the first restricted edge and ϱ_1'' the path up to the second restricted edge. Thus, ϱ_1' consists only of the node N_0 and ϱ_1'' ends at N_5.

The weight of any path across the edge (N_5, N_6), applied to the initial value k_2 must also obey the according restriction, i.e. must be greater or equal to zero. These weights can equally be decomposed into the weight of the direct path to N_5, with respect to k_2, and the single loops. Therefore the following equation system evaluates the conditions for any path to N_6 that extends ϱ_1 by traversing loops $l_1, \ldots, l_5$ for $n_1, \ldots, n_5$ times, respectively.

$$\begin{aligned}k_1 + \mathsf{weight}_{k_1}(\varrho_1') + \left(\sum_{i=1}^{4} n_i \cdot \mathsf{weight}_{k_1}(l_i)\right) &\geq 0\\ k_2 + \mathsf{weight}_{k_2}(\varrho_1'') + \left(\sum_{i=1}^{5} n_i \cdot \mathsf{weight}_{k_2}(l_i)\right) &\geq 0\end{aligned}$$

For the initial values $k_1 = -1.8$ and $k_2 = 1.5$ from the example, there is no natural solution. Hence, such a path cannot exist. For the second path the equation system can be constructed in the same manenr and is, for this example, equal to the one for

ϱ_1. The paths ϱ_1 and ϱ_2 are the only paths that could possibly be extended to a valid path from the initial node to the (only) winning node. But since they do not allow to meet the conditions, there is no path and we can conclude that N_6 and the according configuration in the game $\mathcal{G}(\Phi)$ is not reachable at all. The conclusion is that there is no possibility for Player E to trade the counters for each other in such a way that at some point she can resolve both until formulae.

4.4.1 Restrictions

In the examples presented so far the order, in which loops were traversed, did not matter. It was only necessary to consider the number of times n of traversals for each single loop l. This was due to the fact that the modifications that could be done to the counter values were additive and therefore we could assume that

$$\mathsf{weight}(l^n) = n \cdot \mathsf{weight}(l)$$

where l^n means the concatenation of l for n times, which is always a path in the graph since l starts and ends in the same node.

Unfortunately the game graph might not always be as simple as in the examples above. The approach has, in general, two major drawbacks: The occurrence of restrictions on loops and none-additive weight functions.

Constraints on loops. To realize the problem with constrained loops, consider the formula

$$(p \,\mathrm{R}^{0.3}_{k_1}\, \neg q) \wedge q \,\mathrm{U}^{0.6}_{k_2}\, r$$

for initial values $k_1 = 0$, $k_2 = -1$ and the folded game graph shown in Figure 4.10. Release formulae introduce constraints on loops in the folded graph which are not covered by the equation system constructed. It can happen that the constructed system has a natural solution but which is not realizable. That is, all the plays that follow from that solution by traversing loops the respective number of times are not valid but violate the constraints. In the example, we find the only winning node N_1 in the graph and six loops starting on node N_0. Constructing the equation system analogously to the previous examples leads to only one inequality

$$k_2 + \left(\sum_{i=1}^{6} n_i \cdot \mathsf{weight}_{k_2}(l_i) \right) \geq 0.$$

In loops l_1 to l_4, k_2 is only affected by the label of edge (N_0, N_5), which decreases that counter. The other loops l_1 and l_2 increase the counter by $1 - c_2$ and we find a solution to the equation system with $n_1 = n_2 = \ldots = n_5 = 0$ and $n_6 = 3$ since $-1 + 3 \cdot (1 - 0.6) = 0.2 \geq 0$.

The loop

$$l_6 = N_0 \xrightarrow{1-c_2} N_2 \xrightarrow[+c_1]{k_1<0} N_3 \to N_0$$

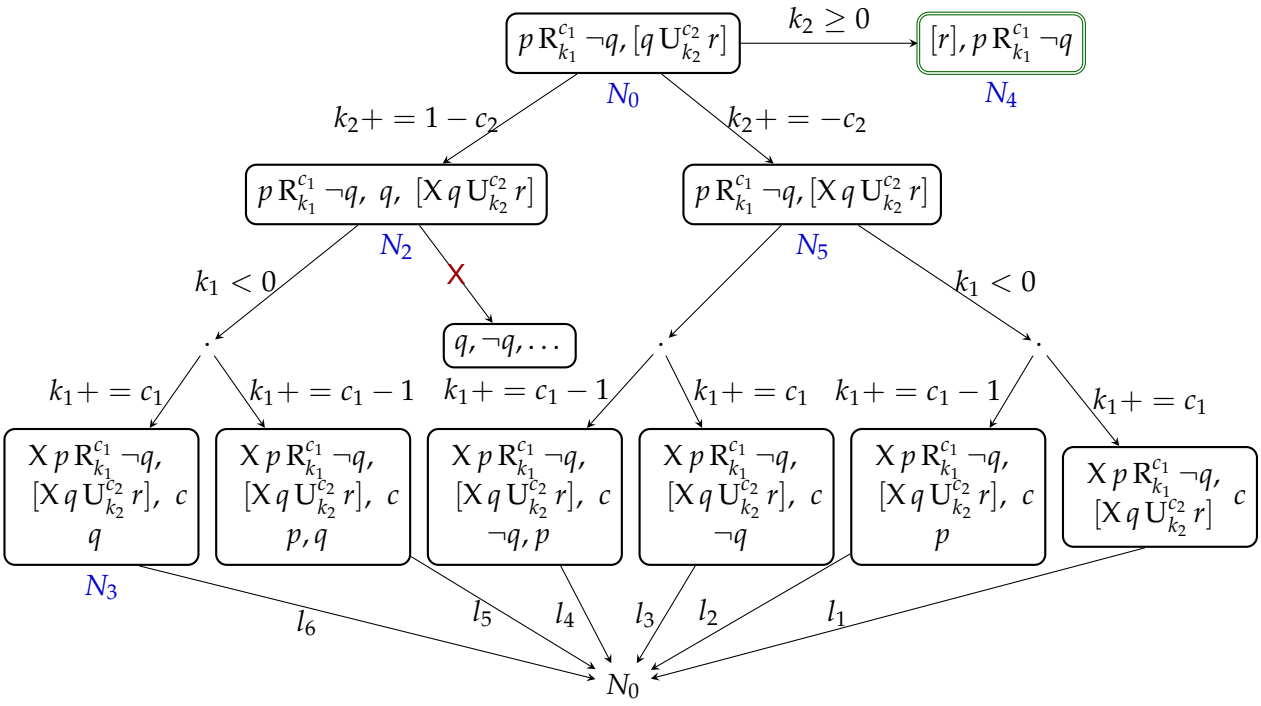


Figure 4.10: Folded game graph of the formula $(p\,\mathrm{R}_{k_1}^{0.3}\,\neg q) \wedge q\,\mathrm{U}_{k_2}^{0.6}\,r$. Node N_4 is marked as winning, since the according focus game would be won by Player E. There is no until formula that needs to be fulfilled and the obligation $\neg q$ can always be satisfied.

indeed gives raise to k_2 but also to k_1. The solution suggests to traverse loop l_6 for $n_6 = 3$ times in order to satisfy the constraint $k_2 \geq 0$ and win the game, which is not possible since E is not allowed to choose the edge (N_2, N_3) because k_1 is not negative. The solution is therefore not realizable by any actual valid play.

Minimum, maximum and reset operations. An additional issue arises from none additive operations on counters. These can not be summed up in a simple equation. Apart from simply adding a constant to a counter it may be necessary to compute a minimum or a maximum between two counters or reset some counter to the initial value.

As incorporated in the rewriting rules (Figure 4.2), for every two counted formulae that differ only in their counter value, one implies the other according to the implication lemma (Lemma 4.2). In case of a counted until, the smaller value determines whether the conjunction is satisfied or not, for a release formula the maximum of both values is determining.

Two equal formulae that are merged according to the maximum or minimum rule can occur because of two reasons: They are either equal sub-formulae at *different* positions in the initial formula, e.g. in a play on $(p\,\mathrm{U}^{c}_{k}\,q) \wedge (r \vee (p\,\mathrm{U}^{c}_{k'}\,q))$ the until sub-formulae may individually appear in the configuration after the choice of Player E. The second possibility is that they originate from the *same* sub-formula that recurred during a play. Recurrence of sub-formulae happens, e.g. if until formulae are nested as in the formula $(p\,\mathrm{U}^{c_1}_{k_1}\,q)\,\mathrm{U}^{c_2}_{k_2}\,r$. The outer until "reproduces" the inner formula $p\,\mathrm{U}^{c_2}_{k_1}\,q$ through its unfolding and the inner formula can reside in a configuration even after a next-step, also because of its unfolding. Then the formula is still present when it repeatedly occurs, as shown in the following play, starting on counter values $\hat{k}_1 = \hat{k}_2 = 0$.

$$
\begin{array}{c}
[\mathsf{unf}((p\,\mathrm{U}^{0.8}_{0}\,q)\,\mathrm{U}^{0.4}_{0}\,r)] = \\
[r \vee ((p\,\mathrm{U}^{0.8}_{0}\,q \wedge \mathrm{X}(p\,\mathrm{U}^{0.8}_{0}\,q\,\mathrm{U}^{0.4}_{0.6}\,r)) \vee \mathrm{X}(p\,\mathrm{U}^{0.8}_{0}\,q\,\mathrm{U}^{0.4}_{-0.4}\,r))] \\
\hline
\mathsf{unf}(p\,\mathrm{U}^{0.8}_{0}\,q), \qquad [\mathrm{X}((p\,\mathrm{U}^{0.8}_{0}\,q)\,\mathrm{U}^{0.4}_{0.6}\,r)] \\
\hline
\mathrm{X}(p\,\mathrm{U}^{0.8}_{-0.8}\,q), \qquad [\mathrm{X}((p\,\mathrm{U}^{0.8}_{0}\,q)\,\mathrm{U}^{0.4}_{0.6}\,r)] \\
\hline
\mathsf{unf}(p\,\mathrm{U}^{0.8}_{-0.8}\,q), \qquad [\mathsf{unf}((p\,\mathrm{U}^{0.8}_{0}\,q)\,\mathrm{U}^{0.4}_{0.6}\,r)] \\
\hline
\mathsf{unf}(p\,\mathrm{U}^{0.8}_{-0.8}\,q), \quad \mathsf{unf}(p\,\mathrm{U}^{0.8}_{0}\,q), \quad [\mathsf{unf}((p\,\mathrm{U}^{0.8}_{0}\,q)\,\mathrm{U}^{0.4}_{0.6}\,r)] \\
\hline
\mathsf{unf}(p\,\mathrm{U}^{0.8}_{-0.8}\,q), \qquad [\mathsf{unf}((p\,\mathrm{U}^{0.8}_{0}\,q)\,\mathrm{U}^{0.4}_{0.6}\,r)] \\
\vdots
\end{array}
\quad (\textsc{min})
$$

In the folded game graph shown in Figure 4.11 this appears as path from N_0 to N_2.

Observe some particularities that result from representing counters only by variables. Apart from only increasing and decreasing them we need more operations to represent the game graph. Whenever a formula recurs, e.g. in node N_2 or node N_5, we need to reflect that the counter value must change according to the actual value *at that point*. We need to distinguish between the two instances of the formula. In the game this is done syntactically because the counter values are explicitly annotated

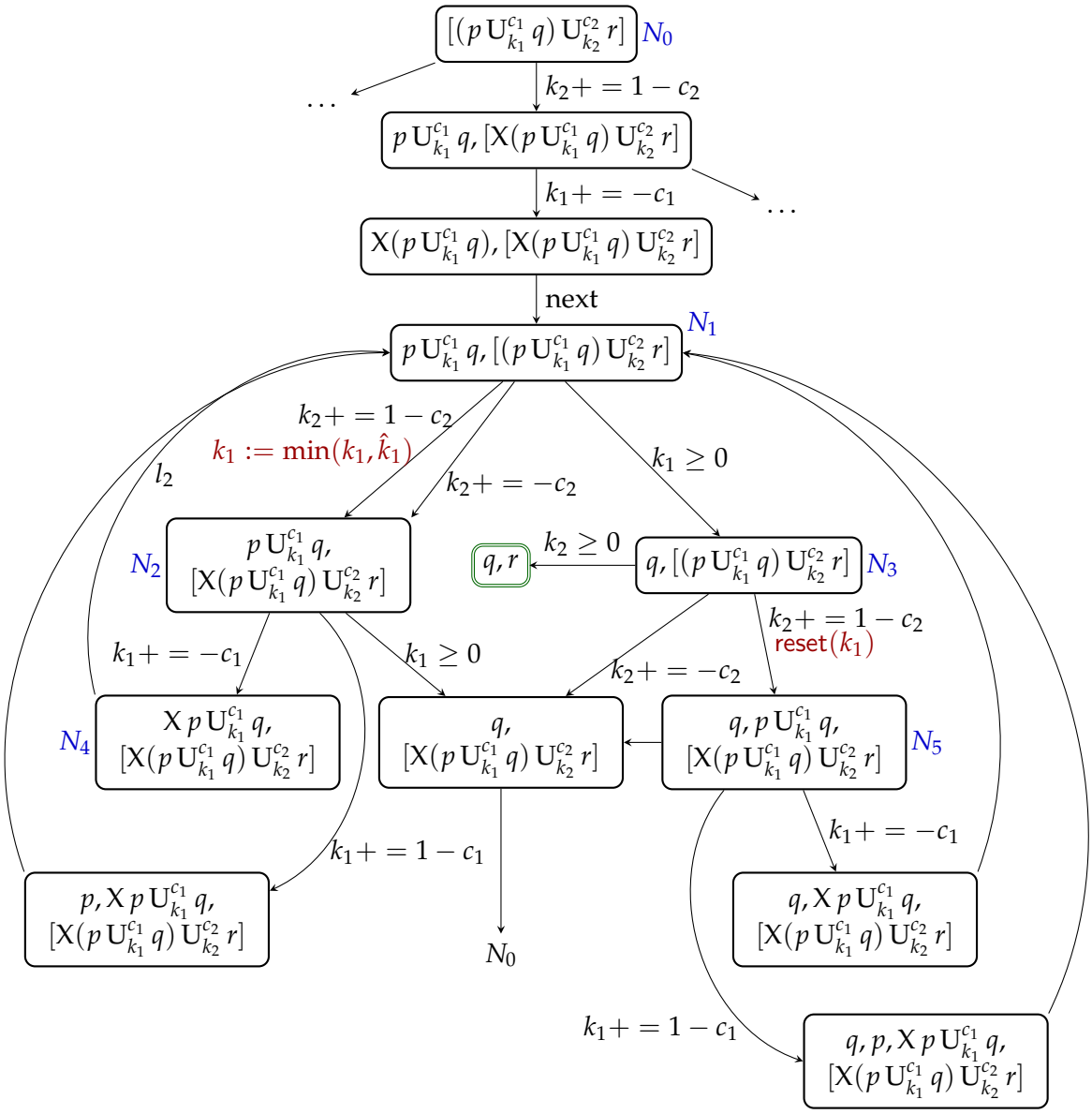


Figure 4.11: Detail of the folded graph for the formula $(p\,\mathrm{U}^{c_1}_{\hat{k}_1}\,q)\,\mathrm{U}^{c_2}_{\hat{k}_2}\,r$. Initial *values* are denoted $\hat{k}_i$, place holder *variables* by k_i.

to the respective operators. But since we replaced the actual values by variables in the folded graph we need to explicitly check, whether a formula is already present in a configuration, every time a rule is applied. The formulae are then merged which requires to calculate the minimum of the current counter value, as it results from the traversed path, and the new instance as it occurs in the initial formula.

Traversing a loop such as

$$(p\,\mathrm{U}^{c_1}_{k_1}\,q) \xrightarrow{k_1+=-c_1} (\mathrm{X}(p\,\mathrm{U}^{c_1}_{k_1}\,q)) \xrightarrow{(next)} (p\,\mathrm{U}^{c_1}_{k_1}\,q)$$

for n times can be reflected by simply adding up the weight n times but we can not reflect the effect of a loop where formulae may recur that way. For example, in order to reflect the effect of a path traversing the loop

$$\begin{aligned}
l_2 = (p\,\mathrm{U}^{c_1}_{k_1}\,q,\ (p\,\mathrm{U}^{c_1}_{k_1}\,q)\,\mathrm{U}^{c_2}_{k_2}\,r) &\qquad (= N_1)\\
\xrightarrow[k_1:=\min(k_1,\hat{k}_1)]{k_2+=1-c_2} (p\,\mathrm{U}^{c_1}_{k_1}\,q,\ \mathrm{X}((p\,\mathrm{U}^{c_1}_{k_1}\,q)\,\mathrm{U}^{c_2}_{k_2}\,r)) &\qquad (= N_2)\\
\xrightarrow{k_1+=-c_1} (\mathrm{X}(p\,\mathrm{U}^{c_1}_{k_1}\,q),\ \mathrm{X}((p\,\mathrm{U}^{c_1}_{k_1}\,q)\,\mathrm{U}^{c_2}_{k_2}\,r)) &\qquad (= N_4)\\
\xrightarrow{(next)} (p\,\mathrm{U}^{c_1}_{k_1}\,q,\ (p\,\mathrm{U}^{c_1}_{k_1}\,q)\,\mathrm{U}^{c_2}_{k_2}\,r) &\qquad (= N_1)
\end{aligned}$$

several times we need to actually nest the weight functions

$$\begin{aligned}
\mathsf{weight}_{k_1}(l_2) &= \min(\hat{k}_1,\hat{k}_1) - c_1 = \hat{k}_1 - c_1\\
\mathsf{weight}_{k_1}(l_2^2) &= \mathsf{weight}_{k_1}(l_2) \circ \mathsf{weight}_{k_1}(l_2)\\
&= \min(\hat{k}_1 - c_1,\hat{k}_1) - c_1\\
\mathsf{weight}_{k_1}(l_2^3) &= \mathsf{weight}_{k_1}(l_2) \circ \mathsf{weight}_{k_1}(l_2) \circ \mathsf{weight}_{k_1}(l_2)\\
&= \min(\min(\hat{k}_1 - c_1,\hat{k}_1) - c_1,\hat{k}_1) - c_2\\
&\vdots
\end{aligned}$$

where the nesting increases for every single traversal of the loop. $\hat{k}_1$ shall denote the initial value, as stated in the formula, in contrast to the place holder variable k_1. We loose the convenient notion of a linear system of inequalities when nesting counted until operators.

Furthermore, another issue becomes apparent at the node $N_3 = (q,\ [(p\,\mathrm{U}^{c_1}_{k_1}\,q)\,\mathrm{U}^{c_2}_{k_2}\,r])$ where Player E might choose to increase the counter k_2. This leads to the node

$$(q,\ p\,\mathrm{U}^{c_1}_{k_1}\,q,\ [\mathrm{X}((p\,\mathrm{U}^{c_1}_{k_1}\,q)\,\mathrm{U}^{c_2}_{k_2}\,r)]).$$

Even though there is no counter k_1 present before we still need to recognize the "new" instance since whatever path lead to this configuration, the weights along it *do not affect* the value of this counter. k_1 is reset to the value that is specified in the initial formula, e.g. zero.

4.4.2 An expressive decidable fragment of $fLTL$

In order to pursue the idea of reducing the satisfiability problem to a set of linear equation systems we need to avoid the complex cases we just considered since they impose non-linear operations on the counter variables that can not be covered by a standard equation system.

Therefore, based on the observations, we suggest a fragment of $fLTL$ where the use of counted operators is restricted. In order to analyse the a formula using inequality systems, the positive normal form of any formula should not contain

- release sub-formulae of the form $\varphi \, \mathrm{R}^c \, \psi$ for $c \neq 0$ or
- nesting of counted until formulae, i.e. sub-formulae $(\varphi \, \mathrm{U}^{c_1} \, \psi) \, \mathrm{U}^{c_2} \, \eta$ for $c_1, c_2 \neq 0$.

Let therefore the $fLTL$ fragment of *simple frequentness properties*, $sfLTL$, consist of formulae of the form

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

where LTL denotes a standard LTL formula.

We can generalize the example in Figure 4.8 consisting of two until formulae to a conjunction of arbitrary many of these formulae. For those we can assure that neither constraints nor minimum, maximum or reset operations occur on loops in the folded game graph. Therefore the inequality system constructed has a natural solution if and only if Player E has e winning strategy for the according game.

Folded game graph for $sfLTL$

The formal definition of the folded game graph is very similar to the extended game graph. Operations and constraints on counters that were imposed implicitly by the unfolding operation are made explicit. Configurations do not contain explicit counter values anymore, but variables. The operations and constraints along the edges are related to these placeholders. For example, consider the configuration $([\mathsf{unf}(p \, \mathrm{U}^{0.8}_{-0.1} \, q)], \{\mathsf{unf}(p \, \mathrm{U}^{0.8}_{-0.1} \, q)\})$. By means of the unfolding operation E has not the possibility to resolve the formula by choosing q since that does not appear in the unfolding due to the negative counter value. In the folded graph, the unfolding operator needs to be changed in order to handle counter variables instead of values. Instead of the value -0.1 there is a placeholder variable k. Also there is an edge that E can choose to move to the configuration $([q], \{q\})$, however, this edge is labelled with the constraint $(k \geq 0)$.

For $c < 1$ and a counter variable (*not* a value) k, we let

$$\begin{aligned} \mathsf{unf}(\varphi \, \mathrm{U}^c_k \, \psi) := \langle \psi \rangle^{k \geq 0} \vee (\langle \varphi \wedge \mathrm{X}(\varphi \, \mathrm{U}^c_k \, \psi) \rangle^{k += 1-c} \\ \vee \langle \mathrm{X}(\varphi \, \mathrm{U}^c_k \, \psi) \rangle^{k += -c}). \end{aligned} \tag{4.3}$$

Then, additional rules

$$\frac{[\langle\varphi\rangle^l], \{\langle\varphi\rangle^l\} \cup \Gamma}{[\varphi], \{\varphi\} \cup \Gamma}\, l \qquad\qquad \frac{[\eta], \{\langle\varphi\rangle^l\} \cup \Gamma}{[\eta], \{\varphi\} \cup \Gamma}\, l \tag{4.4}$$

move the labels, i.e. operations and restrictions, from formulae to the edges in the game graph.

Definition 4.4 (Folded game graph)**.** Let Φ be an $sfLTL$ formula and Φ' the same formula, except that all until formulae $\mu_i = \varphi_i \,\mathrm{U}^{c_i}\, \psi_i$, that carry frequencies $c_i < 1$, are uniquely annotated by a placeholder variable k_i.

The *folded game graph* is an edge-labelled graph $\mathcal{F}(\Phi) = (V, E, \lambda)$. The set of nodes is $V \subseteq \mathsf{sub}(\Phi') \times 2^{\mathsf{sub}(\Phi')}$ and $\lambda : E \to 2^{\Lambda}$ is a mapping of edges to a set of labels.

The labels in Λ are either constraints of the form $k_i \geq 0$, or an operation on a counter variable $k_i += 1 - c_i$ or $k_i += -c_i$, representing functions $k_i \mapsto k_i + 1 - c_i$, $k_i \mapsto k_i - c$, respectively.

The graph is constructed in analogy to the rules in Figure 2.3 for focus games with the exception, that the focus is automatically moved

- to the right formula of a conjunction $[\varphi \wedge \psi]$ if ψ contains an until sub-formula or φ does not, and
- the left formula if φ contains an until sub-formula while ψ does not.

For counted until formulae μ_i, the unfolding operation from Equation 4.3 is used. Annotations from that unfolding are handled by the additional rules from Equation 4.4.

A node $([\varphi], \Gamma) \in V$ is called *winning*, if and only if Γ contains no counted sub-formulae μ_i with $c_i < 1$ and the according standard focus game

$$\mathcal{G}(\bigwedge_{\varphi \in \Gamma} \varphi)$$

is won by Player E.

Definition 4.5 (Paths in folded game graphs)**.** A *path* in the folded game graph $\mathcal{F}(\Phi)$ is a finite sequence

$$\varrho = v_0 \xrightarrow{\lambda(v_0, v_1)} v_1 \xrightarrow{\lambda(v_1, v_2)} v_2 \ldots \xrightarrow{\lambda(v_{n-1}, v_n)} v_n$$

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 *loop* if $v_0 = v_n$.

Let

$$\mathsf{op}_i^k(\varrho) = \begin{cases} l_i & \text{if } l_i = \lambda(v_i, v_{i+1}) \text{ is an operation on } k \\ \mathsf{id} & \text{otherwise} \end{cases}$$

be the operations on a certain variable k along the path ϱ. The *weight* of a path ϱ, with respect to a counter variable k, is the concatenation of the operations along the path ϱ.

$$\mathsf{weight}_k(\varrho) := \bigcirc_{i=0}^{n-1} \mathsf{op}_i^k(\varrho)$$

We call a path *valid* for some initial setting, if the path can actually be traversed without violating restrictions. Let $k_1, \ldots, k_m$ be the counter variables that occur on path ϱ and $\hat{k}_1, \ldots, \hat{k}_m$ initial values for the respective variables. The path ϱ is called *valid* for this preset of values if for every conditional label $\lambda(v_i, v_{i+1}) = (k_j \geq 0)$ of an edge (v_i, v_{i+1}) on ϱ, the weight of the prefix path $\varrho_{(i)}$ ending on the i-th node on ϱ, applied to the initial values, satisfies the condition

$$\mathsf{weight}_{k_j}(\varrho_{(i)})(\hat{k}_j) \geq 0.$$

Equation construction. Let $\mathcal{F}$ be a folded game graph. The equations system $\mathsf{Eqn}(\mathcal{F})$ is constructed as follows.

For each winning node v_f we consider all *simple* paths

$$\varrho = v_0 \xrightarrow{\lambda(v_0,v_1)} v_1 \xrightarrow{\lambda(v_1,v_2)} v_2 \ldots \xrightarrow{\lambda(v_{n-1},v_n)} v_n$$

that start at the initial node and end in $v_n = v_f$.

Now, enumerate all loops $l_1, l_2, \ldots, l_r$, that start on some node on ϱ and are not composed by others, and consider variables $n_1, \ldots, n_r$, respectively. For every v_i on ϱ, such that the edge (v_i, v_{i+1}) is labelled with a restriction $k_j \geq 0$, construct an inequality

$$k_j + \mathsf{weight}_{k_j}(\varrho_{(i)}) + \left(\sum_{l_m \text{ before } v_i} n_m \cdot \mathsf{weight}_{k_j}(l_m) \right) \geq 0$$

These equalities form a system $\mathsf{Eqn}(\varrho)$ and that way we obtain a set

$$\mathsf{Eqn}(\mathcal{F}(\Phi)) = \{\mathsf{Eqn}(\varrho) \mid \varrho \text{ is a simple path to a final node in } \mathcal{F}(\Phi\}$$

of equation systems and there is a system $\mathsf{Eqn}(\varrho) \in \mathsf{Eqn}(\mathcal{F}(\Phi))$ that has a natural solution, if and only if the $sfLTL$ formula Φ is satisfiable.

Theorem 4.3. *Let Φ be an $sfLTL$ formula and $\mathcal{F}(\Phi)$ the folded game graph of Φ. Player E has a winning strategy for $\mathcal{G}(\Phi)$ if and only if there is an equation system $\mathsf{Eqn}(\varrho) \in \mathsf{Eqn}(\mathcal{F}(\Phi))$ that has a natural solution.*

Proof. **(⇒).** Assume there is an equation system $\mathsf{Eqn}(\varrho) \in \mathsf{Eqn}(\mathcal{F}(\Phi))$ for some path ϱ to a final node in $\mathcal{F}(\Phi)$, that has natural solution $(n_1, \ldots, n_r)$ when setting counters k_i to their initial value as given in Φ (all 0 for pure $fLTL$ formulae).

The path ϱ', that follows ϱ, but additionally traverses loops $l_1, \ldots, l_r$ for $n_1, \ldots, n_r$ times, is valid for the initial values. That follows from the solved equation system which is built from the constraints on the edges in the path ϱ. Note that there are no additional constraints in ϱ' since they cannot occur on loops. The valid path ϱ' yields directly a winning play P for Player E in $\mathcal{G}(\Phi)$ against A's optimal strategy. If ϱ' leads to a final node in $\mathcal{F}(\Phi)$ that is a leaf, E wins the play P by reaching a consistent configuration consisting only of atomic propositions. Otherwise the final node is counter free, focuses on a release formula and is reachable from itself. Thus E wins the play P, extended by one traversal of the loop on the last configuration.

($\Leftarrow$). Assume E wins against A, while A is using his optimal strategy. Let P be the according play in $\mathcal{G}(\Phi)$. For P, we find an according path ϱ' in $\mathcal{F}(\Phi)$.

The nodes on ϱ' do eventually not contain a counter anymore, since all until formulae must be resolved eventually if E wins. A resolved until formula can only recur if it is nested in some other release or until formula, which is only possible without counters within the fragment $sfLTL$. We cut off ϱ' at that point, knowing, that the play from that point on is a winning play in a standard focus game.

All paths in $\mathcal{G}(\Phi)$ translate to a valid path in $\mathcal{F}(\Phi)$ since E can only make valid choices in the game. Cutting out all loops from ϱ' yields simple path ϱ, that ends in a winning node. Thus $\mathsf{Eqn}(\mathcal{F}(\Phi))$ contains the system $\mathsf{Eqn}(\varrho)$. The loops $l_1, \ldots, l_r$ on ϱ are traversed for $n_1, \ldots, n_r$ times, respectively, on ϱ'. Since ϱ' is valid, i.e. all constraints are met, starting on initial values for the counters k_i, $n_1, \ldots, n_r$ form a natural solution for $\mathsf{Eqn}(\varrho)$. □

We observe, that the construction from Theorem 3.1 resides in this fragment which can thus expresses none-context-free properties. These separate $sfLTL$ e.g. from monadic second order logic.

To end this section, we summarize the considerations regarding the decidability problem of $fLTL$ by the following theorem.

Theorem 4.4. *Within the fragment $sfLTL$ of $fLTL$ the satisfiability problem is decidable, yet $sfLTL$ formulae can express none-context-free properties and are strictly more expressive then LTL.*

5 Conclusion

The demand for a formalism that allows for specifying properties along with a relative frequency leads us to an exciting and challenging theory. Within the well established framework of temporal logic, the idea of frequentness gains shape in terms of an intuitive, novel extension to LTL. The semantical influence of the syntactical generalization deployed by $fLTL$ is worth thorough investigation and our foundational examination provides an outline on the use and capabilities. We observe that $fLTL$ is expressive, we obtain Turing completeness through simulation of Minsky machines, yet the logic retains the ability to express non context-free languages even in a decidable fragment.

The methods developed, especially the notion of counter semantics for $fLTL$ appear very interesting and encourage further study. The notion of counters was developed in first place to transfer the tableaux-like construction of focus games to the frequency setting. An extension of LTL with counters has, to our knowledge, not been studied so far. Frequentness properties may even be seen as a special application of LTL with counters.

We can imagine a generalization of the counter semantics such that, instead of adding constant values to the counter during the unfolding we allow some restricted or arbitrary operator that may depend on the evaluation of the obligation at the current point, e.g.

$$\varphi \,\mathrm{U}^f_k\, \psi \equiv (\psi \wedge \mathsf{cond}(k)) \vee \mathrm{X}\, \varphi \,\mathrm{U}^f_{k'}\, \psi$$

where $k' = f(k, [\![w \models \varphi]\!])$ and $\mathsf{cond} : \mathsf{Dom}(k) \to \mathbb{B}$ evaluates some release condition on k.

While we considered the notion of the until operator that appeared most natural and unrestricted, there are alternatives to the semantics defined in Section 3.1.

The single attempt policy. Consider Definition 3.1 with the following change for the until-operator.

$$\begin{aligned} w \models \varphi \,\mathrm{U}^c\, \psi \quad \text{iff} \quad & \exists_j : w^j \models \psi \text{ and} \\ & \#_{\varphi,w}(j) \geq c \cdot j \textbf{ and} \\ & \forall_{k<j} : w^k \models \neg\psi. \end{aligned}$$

The property $\varphi \,\mathrm{U}^c\, \psi$ is stricter now: ψ must hold eventually, and whenever it holds *first*, the number of positions fulfilling φ must be sufficiently high. In turn, strength-

ening the until that way directly weakens the release since we have

$$\begin{aligned} \neg\Big(\quad & \exists_j : w^j \models \neg\psi \quad \text{and} \quad \forall_{k<j} : w^k \models \neg\psi \ \text{ and} \\ & \qquad\qquad\qquad\quad |\{i \mid 0 \le i < j,\ w^i \models \neg\varphi\}| \ge (1-c)\cdot j\Big) \\ \Leftrightarrow \qquad & \forall_j : w^j \models \psi \qquad \text{or} \quad \exists_{k<j} : w^k \models \neg\psi \ \text{ or} \\ & \qquad\qquad\qquad\quad |\{i \mid 0 \le i < j,\ w^i \models \varphi\}| > c\cdot j. \end{aligned}$$

In other words, only for the first position that does not satisfy ψ, if any, there must be enough preceding positions satisfying φ. This appears to be somehow closer to the traditional notion of release from the point of view that there is the possibility to "fulfill" the formula at some point. On the other hand, the event of "releasing" ψ does require its violation. Take, for example, the formulae $\Phi = p\,\mathrm{R}\,q$ and $\Phi' = p\,\mathrm{R}^{\frac{1}{4}}\,q$. In a word starting with

$$\{q\}\ \{q\}\ \{q\}\ \{p,q\}\ \{q\}\ \{q\}\ \{\}\ \ldots$$

the occurrence of p at the fourth position releases q independently of q for the formula Φ. On the contrary considering Φ', q is not released just because it does not "demand" it.

The observations in terms of expressivity and decidability seem to remain for this variation which may indicate a certain robustness of the definition.

No dept policy. Instead of evaluating the ratio of fulfilled obligations only at the moment when the eventuality is satisfied, we could impose a global restriction, e.g. that the ratio is *never* allowed to fall beyond the threshold c.

$$\begin{aligned} w \models \varphi\,\mathrm{U}^c\,\psi \quad \text{iff} \quad & \exists_j : w^j \models \psi \text{ and} \\ & \forall_{i<j} : \#_{\varphi,w}(i) \ge c\cdot i \end{aligned}$$

In terms of counters that simply means, that a counter can never get negativeor in terms of games, that the existential player is allowed to ignore the obligation only if it was satisfied at sufficiently many positions before.

Note, that all these definition are still consistent with the standard semantics of LTL. The operator U remains the special case of U^c for $c = 1$. However, the until operator is, in a sense, stronger and thus closer to the standard version. Recall that a weaker until operator dually results in a stronger release and vice versa. The semantics above allow, just as in LTL, an eventual resolution of a release formula.

While the single attempt policy seem to retain the major results for $fLTL$, this variation appears less expressive.

We can give an extension to first-order logic that covers $fLTL$ completely, the converse is, however, uncertain. A predicate $\mathcal{H}_{\ge c}(\psi(z), x, y)$ defines a scope, the interval between positions given by variables $x, y \in V$. Within that scope, the ratio between positions that satisfy a formula φ, possibly with one free variable, and the length of

the scope is compared to a given fraction $c \in [0,1]$. Formally $(w,\sigma) \models \mathcal{H}_{\geq c}(\psi(z),x,y)$ if and only if

$$|\{k \mid x \leq k < y \wedge (w,\sigma[k/z]) \models \psi\}| \geq c \cdot (j-i).$$

The extended U operator can then be characterized in terms of first-order logic together with that predicate:

$$w \models \varphi \,\mathrm{U}^c\, \psi \qquad \text{iff} \qquad (w, x \mapsto 0) \models \exists_j : \hat{\psi}(j) \wedge \mathcal{H}_{\geq c}(\hat{\varphi}(z),x,j)$$

$\hat{\varphi}$ and $\hat{\psi}$ denote the formulae φ and ψ, translated inductively to first-order formulae in at most one free variable.

It remains open, whether $fLTL$ can express arbitrary *regular* properties and it is subject to further study if the notion of frequency can be lifted to a related, yet more expressive formalisms such as *regular LTL* by Leucker and Sánchez [LS07] or fixed-point logics such as the modal μ-calculus [Koz83].

Acknowledgements

I would like to thank Martin Leucker for all his advice and supervision. The work and our discussions were very encouraging and strongly confirmed my objectives and aim towards further work and studies.

Also, I gladly acknowledge the influential contribution of Benedikt Bollig to the formulation of constitutive ideas in this report.

Ich danke vor allem auch meinen Eltern, die mir ein ungezwungenes Studium ermöglicht haben und stets alle Freiheiten gaben, ohne mich jemals alleine zu lassen.

References

[Bur74] Rod M. Burstall. Program proving as hand simulation with a little induction. In *IFIP Congress*, pages 308–312, 1974.

[dA97] Luca de Alfaro. *Formal Verification of Probabilistic Systems*. Ph.d. dissertation, Stanford University, 1997. Technical report STAN-CS-TR-98-1601.

[DG08] Volker Diekert and Paul Gastin. First-order definable languages. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2008.

[DL05] Christian Dax and Martin Lange. Game over: The foci approach to ltl satisfiability and model checking. *Electr. Notes Theor. Comput. Sci.*, 119(1):33–49, 2005.

[GO01] Paul Gastin and Denis Oddoux. Fast ltl to büchi automata translation. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *CAV*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2001.

[GPSS80] Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal analysis of fairness. In *POPL*, pages 163–173, 1980.

[GPVW95] Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In Piotr Dembinski and Marek Sredniawa, editors, *PSTV*, volume 38 of *IFIP Conference Proceedings*, pages 3–18. Chapman & Hall, 1995.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[HKNP06] Andrew Hinton, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Prism: A tool for automatic verification of probabilistic systems. In Holger Hermanns and Jens Palsberg, editors, *TACAS*, volume 3920 of *Lecture Notes in Computer Science*, pages 441–444. Springer, 2006.

[HMO10] Jochen Hoenicke, Roland Meyer, and Ernst-Rüdiger Olderog. Kleene, rabin, and scott are available. In Paul Gastin and François Laroussinie, editors, *CONCUR*, volume 6269 of *Lecture Notes in Computer Science*, pages 462–477. Springer, 2010.

[Kam68] Hans W. Kamp. *Tense Logic and the Theory of Linear Order*. Phd thesis, Computer Science Department, University of California at Los Angeles, USA, 1968.

[Koz83] Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[Kri59] Saul Kripke. A completeness theorem in modal logic. *J. Symb. Log.*, 24(1):1–14, 1959.

[KV01] Orna Kupferman and Moshe Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

[LS01] Martin Lange and Colin Stirling. Focus games for satisfiability and completeness of temporal logic. In *LICS*, pages 357–365, 2001.

[LS07] Martin Leucker and César Sánchez. Regular linear temporal logic. In Cliff B. Jones, Zhiming Liu, and Jim Woodcock, editors, *ICTAC*, volume 4711 of *Lecture Notes in Computer Science*, pages 291–305. Springer, 2007.

[Min67] Marvin L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.

[Pri57] Arthur N. Prior. *Time and Modality*. Oxford University Press, 1957.

[Pri67] Arthur N. Prior. *Past, Present and Future*. Oxford University Press, 1967.

[SLSR07] Usa Sammapun, Insup Lee, Oleg Sokolsky, and John Regehr. Statistical runtime checking of probabilistic properties. In Oleg Sokolsky and Serdar Tasiran, editors, *RV*, volume 4839 of *Lecture Notes in Computer Science*, pages 164–175. Springer, 2007.

[Smu68] Raymond M Smullyan. *First-order logic [by] Raymond M. Smullyan*. Ergebnisse der Mathematik und ihrer Grenzgebiete ; Bd. 43. Springer-Verlag, Berlin, New York [etc.], 1968. Bibliography: p. [156].

[Tho97] Wolfgang Thomas. Languages, automata, and logic. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of formal languages, vol. 3: beyond words*, pages 389–455. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[VW86] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *LICS*, pages 332–344. IEEE Computer Society, 1986.

[VW94] Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Inf. Comput.*, 115(1):1–37, 1994.

[Wol85] Pierre Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, (110–111):119–136, 1985.

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Normann Decker)