FI MU

Faculty of Informatics
Masaryk University

The Stuttering Principle Revisited: On the Expressiveness of Nested X and U Operators in the Logic LTL

by

Antonín Kučera
Jan Strejček

FI MU Report Series FIMU-RS-2002-03

July 2002

The Stuttering Principle Revisited: On the Expressiveness of Nested X and U Operators in the Logic LTL

Antonín Kučera* Jan Strejček†

Faculty of Informatics, Masaryk University,
Botanická 68a, 60200 Brno, Czech Republic,
{tony,strejcek}@fi.muni.cz.

Abstract

It is known that LTL formulae without the 'next' operator are invariant under the so-called *stutter-equivalence* of words. In this paper we extend this principle to general LTL formulae with given nesting depths of the 'next' and 'until' operators. This allows us to prove the semantical strictness of three natural hierarchies of LTL formulae, which are parametrized either by the nesting depth of just one of the two operators, or by both of them. As another interesting corollary we obtain an alternative characterization of LTL languages, which are exactly the regular languages closed under the generalized form of stutter equivalence. We also indicate how to tackle the state-space explosion problem with the help of presented results.

*Supported by the Grant Agency of Czech Republic, grant No. 201/00/1023.

†Supported by the Grant Agency of Czech Republic, grant No. 201/00/0400, and by a grant FRVŠ No. 601/2002.

1 Introduction

Linear temporal logic (LTL) [Pnu77] is a popular formalism for specifying properties of (concurrent) programs. The syntax of LTL is given by the following abstract syntax equation:

$$\varphi \ ::= \ \mathtt{tt} \ \mid \ p \ \mid \ \neg\varphi \ \mid \ \varphi_1 \wedge \varphi_2 \ \mid \ \mathsf{X}\varphi \ \mid \ \varphi_1 \,\mathsf{U}\, \varphi_2$$

Here p ranges over a countable set $\Lambda = \{o, p, q, \dots\}$ of *letters*. We also use $\mathsf{F}\varphi$ to abbreviate $\mathtt{tt}\,\mathsf{U}\,\varphi$, and $\mathsf{G}\varphi$ to abbreviate $\neg\mathsf{F}\neg\varphi$.

In this paper, we are mainly interested in theoretical aspects of LTL (though some remarks on a potential applicability of our results to model-checking with the logic LTL are mentioned in Section 4). To simplify our notation, we define the semantics of LTL in terms of languages over finite words (all of our results carry over to infinite words immediately). An *alphabet* is a finite set $\Sigma \subseteq \Lambda$. Let Σ be an alphabet and φ an LTL formula. Let $w \in \Sigma^*$ be a word over Σ. The length of w is denoted by $|w|$, and the individual letters of w are denoted by $w(0), w(1), \dots, w(n-1)$, where $n = |w|$. Moreover, for every $0 \le i < |w|$ we denote by w_i the i^{th} suffix of w, i.e., the word $w(i) \cdots w(|w|-1)$. Finally, for all $0 \le i < |w|$ and $j \ge 1$ such that $i+j \le |w|$ the symbol $w(i,j)$ denotes the subword of w of length j which starts with $w(i)$.

Remark 1.1. *To simplify our notation, we adopt the following convention: whenever we refer to $w(i)$, w_i, or $w(i,j)$, we implicitly impose the condition that the object* exists. *For example, the condition '$w(4) = p$' should be read 'the length of w is at least* 5 *and $w(4) = p$'.*

The *validity* of φ for $w \in \Sigma^*$ is defined as follows:

$$\begin{array}{lll} w \models \mathtt{tt} & & \\ w \models p & \text{iff} & p = w(0) \\ w \models \neg\varphi & \text{iff} & w \not\models \varphi \end{array}$$

$$
\begin{array}{lll}
w \models \varphi_1 \wedge \varphi_2 & \text{iff} & w \models \varphi_1 \wedge w \models \varphi_2 \\
w \models \mathsf{X}\varphi & \text{iff} & w_1 \models \varphi \\
w \models \varphi_1 \,\mathsf{U}\, \varphi_2 & \text{iff} & \exists i \in \mathbb{N}_0 : w_i \models \varphi_2 \wedge \forall\, 0 \le j < i : w_j \models \varphi_1
\end{array}
$$

For every alphabet Σ, every LTL formula φ defines the language $L^{\Sigma}_{\varphi} = \{w \in \Sigma^* \mid w \models \varphi\}$. From now on we omit the 'Σ' superscript in L^{Σ}_{φ}, because it is always clearly determined by the context.

It is well-known that languages definable by LTL formulae form a proper subclass of regular languages [Tho91]. More precisely, LTL languages are exactly the languages definable in first-order logic [Kam68] and thus exactly the languages recognizable by deterministic counter-free automata [MP71].

Since LTL contains just two modal connectives, a natural question is how they influence the expressive power of LTL. First, let us (inductively) define the *nesting depth* of the X and the U modality in a given LTL formula φ, denoted $X(\varphi)$ and $U(\varphi)$, respectively.

$$
\begin{array}{ll}
U(\mathtt{tt}) = 0 & X(\mathtt{tt}) = 0 \\
U(p) = 0 & X(p) = 0 \\
U(\varphi \wedge \psi) = \max\{U(\varphi), U(\psi)\} & X(\varphi \wedge \psi) = \max\{X(\varphi), X(\psi)\} \\
U(\mathsf{X}\varphi) = U(\varphi) & X(\mathsf{X}\varphi) = X(\varphi) + 1 \\
U(\varphi \,\mathsf{U}\, \psi) = \max\{U(\varphi), U(\psi)\} + 1 & X(\varphi \,\mathsf{U}\, \psi) = \max\{X(\varphi), X(\psi)\}
\end{array}
$$

Now we can introduce three natural hierarchies of LTL formulae. For all $m, n \in \mathbb{N}_0$ we define

$$
\begin{array}{rcl}
\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n) & = & \{\varphi \in \mathrm{LTL} \mid U(\varphi) \le m \ \wedge\ X(\varphi) \le n\} \\
\mathrm{LTL}(\mathsf{U}^m) & = & \bigcup_{i=0}^{\infty} \mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^i) \\
\mathrm{LTL}(\mathsf{X}^n) & = & \bigcup_{i=0}^{\infty} \mathrm{LTL}(\mathsf{U}^i, \mathsf{X}^n)
\end{array}
$$

Hence, the $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$ hierarchy takes into account the nesting depths of both modalities, while the $\mathrm{LTL}(\mathsf{U}^m)$ and $\mathrm{LTL}(\mathsf{X}^n)$ hierarchies 'count' just the nesting depth of U and X, respectively. Our work is motivated by basic questions about the presented hierarchies; in particular, the following problems seem to be among the most natural ones:

Question 1. Are those hierarchies semantically strict? That is, if we increase m or n just by one, do we always obtain a strictly more expressive fragment of LTL?

Question 2. If we take two classes A, B in the above hierarchies which are syntactically incomparable (for example, we can consider $\mathrm{LTL}(\mathsf{U}^4, \mathsf{X}^3)$ and $\mathrm{LTL}(\mathsf{U}^2, \mathsf{X}^5)$, or $\mathrm{LTL}(\mathsf{U}^3, \mathsf{X}^0)$ and $\mathrm{LTL}(\mathsf{U}^2)$), are they also semantically incomparable? That is, are there formulae $\varphi_A \in A$ and $\varphi_B \in B$ such that φ_A is not expressible in B and φ_B is not expressible in A?

Question 3. In the case of $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$ hierarchy, what is the semantical intersection of $\mathrm{LTL}(\mathsf{U}^{m_1}, \mathsf{X}^{n_1})$ and $\mathrm{LTL}(\mathsf{U}^{m_2}, \mathsf{X}^{n_2})$? That is, what languages are expressible in both fragments?

We provide (positive) answers to Question 1 and Question 2. Here, the results about $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$ hierarchy seem to be particularly interesting. As for Question 3, one is tempted to expect the following answer: The semantical intersection of $\mathrm{LTL}(\mathsf{U}^{m_1}, \mathsf{X}^{n_1})$ and $\mathrm{LTL}(\mathsf{U}^{m_2}, \mathsf{X}^{n_2})$ are exactly the languages expressible in $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$, where $m = \min\{m_1, m_2\}$ and $n = \min\{n_1, n_2\}$. Surprisingly, this answer turns out to be *incorrect*. For all $m \geq 1, n \geq 0$ we give an example of a language L which is definable both in $\mathrm{LTL}(\mathsf{U}^{m+1}, \mathsf{X}^n)$ and $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^{n+1})$, but not in $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$. It shows that the answer to Question 3 is not so easy as one might expect. In fact, Question 3 is left open as an interesting challenge directing our future work.

The results on Question 1 are closely related to the work of Etessami and Wilke [EW00] (see also [Wil99] for an overview of related results). They consider an until hierarchy of LTL formulae which is similar to our $\mathrm{LTL}(\mathsf{U}^m)$ hierarchy. The difference is that they treat the F operator 'explicitly', i.e., their U-depth counts just the nesting of the U-operator and ignores all occurrences of X and F (in our approach, $\mathsf{F}\varphi$ is just an abbreviation for $\mathtt{tt}\,\mathsf{U}\,\varphi$, and hence 'our' U-depth of $\mathsf{F}p$ is one and not zero). They prove the strictness of their until hierarchy in the following way: First, they design an appropriate Ehrenfeucht-Fraïssé game for LTL (the game is played on a pair of words) which in a sense characterizes those pairs of words which can be distinguished by an LTL formulae where the temporal operators are nested only to a certain depth. Then, for every k they construct a formula Fair_k with until depth k and prove that this particular formula cannot be equivalently expressed by any (other) formula with U-depth $k-1$ (here the previous results about the designed EF game are used). Since the formula Fair_k contains just one F operator (and many nested X and U operators), this proof carries over to our LTL(U^m) hierarchy. In fact, [EW00] is in a sense 'stronger' result saying that one additional nesting level of U cannot be 'compensated' by arbitrarily-deep nesting of X and F. On the other hand, the proof does not allow to conclude that, e.g., LTL($\mathsf{U}^3, \mathsf{X}^0$) contains a formula which is not expressible in LTL(U^2) (because Fair_k contains the nested X modalities).

Our method for solving Questions 1 and 2 is different. Instead of designing appropriate Ehrenfeucht-Fraïssé games which could (possibly) characterize the membership to LTL($\mathsf{U}^m, \mathsf{X}^n$), we formulate a general 'stuttering theorem' for LTL($\mathsf{U}^m, \mathsf{X}^n$) languages. Roughly speaking, the theorem says that under certain 'local-periodicity' conditions (which depend on m and n) one can remove a given subword u from a given word w without influencing the (in)validity of LTL($\mathsf{U}^m, \mathsf{X}^n$) formulae (we say that u is (m,n)-*redundant* in w). This result can be seen as a generalization of the well-known form of stutter-invariance admitted by LTL(X^0) formulae (a detailed discussion is postponed to Section 2). Thus, we obtain a simple (but surprisingly powerful) tool allowing to prove that a certain formula φ is *not* definable in LTL($\mathsf{U}^m, \mathsf{X}^n$). The theorem is applied as follows: we choose a suitable alphabet Σ, consider the language L_φ, and find an appropriate $w \in L_\varphi$ and its subword u such that

- u is (m, n)-redundant in w;
- $w' \not\models \varphi$ where w' is obtained from w by deleting the subword u.

If we manage to do that, we can conclude that φ is not expressible in $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$.

We use our stuttering theorem to answer Questions 1 and 2. Proofs are remarkably short (though it took us some time to find appropriate formulae which witness the presented claims). As another interesting corollary we obtain an alternative characterization of LTL languages which are exactly the regular languages closed under the generalized stutter equivalence of words. It is worth noting that some of the known results about LTL (like, e.g., the formula '$\mathsf{G}_2 p$' is not definable in LTL) admit a one-line proof if our general stuttering theorem is applied.

The paper is organized as follows. In Section 2 we formulate and prove the general stuttering theorem for $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$ languages, together with some of its direct corollaries. In Section 3 we answer the Questions 1–3 in the above indicated way. In Section 4 we briefly discuss a potential applicability of our results to the problem of state-space explosion in the context of model-checking with LTL. Finally, in Section 5 we draw our conclusions and identify directions of future research.

2 A general stuttering theorem for $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$

In this section we formulate and prove the promised stuttering theorem for $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$ languages. The definition is slightly technical and therefore we start with some intuition which aims to explain the underlying principles.

It is well-known that LTL(X^0) formulae (i.e., formulae without the X operator) are *stutter invariant*. It means that one can safely delete *redundant letters* from words without influencing the (in)validity of LTL(X^0) formulae (a letter $w(i)$ is redundant in w if $w(i) = w(i{+}1)$). Intuitively, it is not very surprising that this principle can be extended to LTL(X^n) formulae (where $n \in \mathbb{N}_0$). We say that a letter $w(i)$ is n-*redundant* if $w(i) = w(i{+}j)$ for every $1 \leq j \leq n{+}1$. Now we could prove that LTL(X^n) formulae are n-*stutter invariant* in the sense that deleting n-redundant letters from words does not influence (in)validity of LTL(X^n) formulae (we do not provide an explicit proof here because this claim is an immediate consequence of our general stuttering theorem; see also the 'pedagogical' remarks at the end of this section). Hence, LTL(X^n) languages are closed under deleting (as well as 'pumping') of n-redundant letters.

Since the notion of n-redundancy depends just on the X-depth of LTL formulae, one can also ask if there is another 'pumping principle' which depends mainly on the U-depth of LTL formulae; and indeed, there is one. In this case, we do not necessarily pump just individual letters, but whole subwords. To give some basic intuition, let us first consider the formula $\varphi \equiv (o \vee p)\,\mathsf{U}\,q$. Let $w \in \{o,p,q\}^*$ be a word such that $w \models \varphi$. We claim that if w is of the form $w = vuux$, where $v,u,x \in \Sigma^*$, then the word $w' = vux$ also satisfies φ. Our (general) arguments can be easier understood if they are traced down to the following example:

$$\begin{array}{rcl} w & = & \overbrace{ppp}^{v}\overbrace{oppqr}^{u}\overbrace{oppqr}^{u}\overbrace{orp}^{x} \\ w' & = & \underbrace{ppp}_{v}\underbrace{oppqr}_{u}\underbrace{orp}_{x} \end{array}$$

Since $w \models \varphi$, there is w_i such that $w_i \models q$. Now we can distinguish three possibilities.

1. If $w(i)$ is within v, then deleting the first copy of u does not influence the validity of φ (in this case we could in fact delete the whole subword uux).

2. If $w(i)$ is within the second copy of u or within x, then the first copy of u can also be deleted without any problem.

3. If $w(i)$ is within the first copy of u then we can delete the *second* copy of u and the resulting word still satisfies φ.

The previous observation is actually valid for all $\mathrm{LTL}(\mathsf{U}^1, \mathsf{X}^0)$ formulae. Moreover, one could prove (by induction on n) that for every $\varphi \in \mathrm{LTL}(\mathsf{U}^n, \mathsf{X}^0)$ and a word $w = vu^{n+1}x$ such that $w \models \varphi$ we have that $w' = vu^n x$ also models φ. However, we can do even better; there is one subtle point in the inductive argument which becomes apparent only when considering $\mathrm{LTL}(\mathsf{U}^n, \mathsf{X}^0)$ formulae where $n \geq 2$. To illustrate this, let us take $\varphi \equiv (o \,\mathsf{U}\, p) \,\mathsf{U}\, (q \,\mathsf{U}\, r)$ and let w be a word of the form $w = vususux$ where $|s| = 1$. Hence, the subword us is repeated 'basically' twice after its first occurrence, but in the last copy we do not insist on the last letter (the missing 's'). We claim that if $w \models \varphi$, then also the word $w' = vusux$ models φ. Again, the reason can be well illustrated by an example:

$$\begin{aligned} w &= \overbrace{ppp}^{v}\overbrace{oppp}^{u}\overbrace{r}^{s}\overbrace{oppp}^{u}\overbrace{r}^{s}\overbrace{oppp}^{u}\overbrace{ooop}^{x} \\ w' &= \underbrace{ppp}_{v}\underbrace{oppp}_{u}\underbrace{r}_{s}\underbrace{oppp}_{u}\underbrace{ooop}_{x} \end{aligned}$$

Since $w \models \varphi$, there must be some w_i such that $w_i \models q \,\mathsf{U}\, r$. The most interesting situation is when $w(i)$ happens to be within the first copy of us. Actually, the 'worst' possibility is when $w(i)$ is the s (see the example above). As the U-depth of $q \,\mathsf{U}\, r$ is just one, we can rely on our previous observation; since $w_i = susux$, we can surely remove the leading su subword. Thus, $sux \models q \,\mathsf{U}\, r$. In a similar way we can show that $ysux \models o \,\mathsf{U}\, p$ for each suffix y of vu (we know that $ysusux \models o \,\mathsf{U}\, p$ and hence we can again apply our previous observations). Now we can readily confirm that indeed $vusux \models \varphi$.

Increasing U-depth of LTL($\mathsf{U}^m, \mathsf{X}^0$) formulae allows to ignore more and more 'trailing letters'. More precisely, for any LTL($\mathsf{U}^m, \mathsf{X}^0$) formula φ we can 'ignore' the last $m-1$ letters in the repeated pattern.

Our general stuttering theorem for LTL($\mathsf{U}^m, \mathsf{X}^n$) formulae combines both forms of stuttering (i.e., the 'letter stuttering' for the X operator, and the 'subword stuttering' for the U operator). In the next definition, the symbol u^ω (where $|u| \geq 1$) denotes the infinite word obtained by concatenating infinitely many copies of u.

Definition 2.1. *Let Σ be an alphabet and $w \in \Sigma^*$. A subword $w(i,j)$ is (m,n)-*redundant *in w iff the word $w(i+j, m \cdot j + n - m + 1)$ is a prefix of $w(i,j)^\omega$ (i.e., the subword $w(i,j)$ is repeated at least on the next $m \cdot j + n - m + 1$ letters).*

In the context of previous remarks, the above definition admits a good intuitive interpretation; the subword $w(i,j)$ has to be repeated 'basically' m times after its first occurrence (the $m \cdot j$ summand), but we can ignore the last $m-1$ letters. Since there can be n nested X operators, we must 'prolong' the repetition by n letters. Hence, the total number of letters by which we must prolong the repetition is $n - (m-1) = n - m + 1$. Before proving the stuttering theorem, we need to state one auxiliary lemma.

Lemma 2.2. *Let Σ be an alphabet, $m, n \in \mathbb{N}_0$, and $w \in \Sigma^*$. If a subword $w(i,j)$ is*

(i) *(m,n)-redundant then it is also (m',n')-redundant for all $0 \leq n' \leq n$ and $0 \leq m' \leq m$.*

(ii) *$(m,n+1)$-redundant then the subword $w(i+1,j)$ is (m,n)-redundant.*

(iii) *$(m+1,n)$-redundant then the subword $w(i+k,j)$ is (m,n)-redundant for every $0 \leq k < j$.*

Proof. (i) follows immediately as $j > 0$ implies $m' \cdot j + n' - m' + 1 \leq m \cdot j + n - m + 1$. (ii) is also simple—due to the $(m, n+1)$-redundancy of $w(i,j)$ we know that the subword is repeated at least on the next $m \cdot j + n - m + 2$ letters. Hence, the subword $w(i{+}1,j)$ is repeated at least on the next $m \cdot j + n - m + 1$ letters and thus it is (m,n)-redundant. A proof of (iii) is similar; if $w(i,j)$ is repeated on the next $(m{+}1) \cdot j + n - m$ letters, then the subword $w(i{+}k,j)$ (where $0 \leq k < j$) is repeated on the next $(m{+}1) \cdot j + n - m - k = m \cdot j + n - m + j - k$ letters, i.e., $w(i{+}k,j)$ is $(m, n + j - k - 1)$-redundant. The (m,n)-redundancy of $w(i{+}k,j)$ follows from (i) and $k < j$. $\square$

Definition 2.3. *Let Σ be an alphabet. For all $m, n \in \mathbb{N}_0$ we define the relation $\prec_{m,n} \subseteq \Sigma^* \times \Sigma^*$ as follows: $w \prec_{m,n} v$ iff v can be obtained from w be deleting some (m,n)-redundant subword. We say that $w, v \in \Sigma^*$ are (m,n)-stutter equivalent iff $w \approx_{m,n} v$, where $\approx_{m,n}$ is the least equivalence on Σ^* containing $\prec_{m,n}$. We say that a language $L \subseteq \Sigma^*$ is* (m,n)-stutter closed *if it is closed under $\approx_{m,n}$.*

Theorem 2.4 (stuttering theorem for $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$). *Let Σ be an alphabet, and let $\varphi \in \mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$ where $m, n \in \mathbb{N}_0$. The language L_φ is (m,n)-stutter closed.*

Proof. Let $\varphi \in \mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$. It suffices to prove that for all $w, v \in \Sigma^*$ such that $w \prec_{m,n} v$ we have that $w \models \varphi \iff v \models \varphi$. We proceed by a simultaneous induction on m and n (we write $(m', n') < (m, n)$ iff $m' \leq m$ and $n' < n$, or $m' < m$ and $n' \leq n$).

Basic step: $m = 0$ and $n = 0$. Let $w, v \in \Sigma^*$ be words such that $w \prec_{0,0} v$. Let $w(i,j)$ be the $(0,0)$-redundant subword of w which has been deleted to obtain v. Since $\mathrm{LTL}(\mathsf{U}^0, \mathsf{X}^0)$ formulae are just 'Boolean combinations' of letters and `tt`, it suffices to show that $w(0) = v(0)$. If $i > 0$ then it is clearly the case. If $i = 0$, then $v(0) = w(j)$ and the $(0,0)$-redundancy of $w(0,j)$ implies that $w(j) = w(0)$.

Induction step: Let $m, n \in \mathbb{N}_0$, and let us assume (I.H.) that the theorem holds for all m', n' such that $(m', n') < (m, n)$. Let $\varphi \in \mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$ and let $w, v \in \Sigma^*$ be words such that $w \prec_{m,n} v$. Let $w(i, j)$ be the (m, n)-redundant subword of w which has been deleted to obtain v. We distinguish four possibilities:

- $\varphi \in \mathrm{LTL}(\mathsf{U}^{m'}, \mathsf{X}^{n'})$ for some $(m', n') < (m, n)$. Since $w(i, j)$ is (m', n')-redundant by Lemma 2.2 (i), we can apply the induction hypothesis.
- $\varphi = \mathsf{X}\psi$. We need to prove that $w_1 \models \psi \iff v_1 \models \psi$. As ψ is an $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^{n-1})$ formula and $(m, n-1) < (m, n)$, the induction hypothesis implies that ψ cannot distinguish between words related by $\prec_{m,n-1}$. Hence, it suffices to show that $w_1 \prec_{m,n-1} v_1$. Let us consider the subword $w(i, j)$. If $i > 0$ then $w_1(i-1, j)$ is (m, n)-redundant and due to Lemma 2.2 (i) it is also $(m, n-1)$-redundant. Furthermore, v_1 can be obtained from w_1 by deleting the subword $w_1(i-1, j)$.

  If $i = 0$ then $w(0, j)$ is (m, n)-redundant. Lemma 2.2 (ii) implies that $w(1, j)$ is $(m, n-1)$-redundant. It means that the subword $w_1(0, j)$ is $(m, n-1)$-redundant. Furthermore, v_1 is obtained from w_1 by deleting $w_1(0, j)$.
- $\varphi = \psi \,\mathsf{U}\, \rho$. As the subformulae ψ, ρ belong to $\mathrm{LTL}(\mathsf{U}^{m-1}, \mathsf{X}^n)$, they cannot (by induction hypotheses) distinguish between words related by $\prec_{m-1,n}$.

  Let $g : \{0, 1, \ldots, |w|-1\} \longrightarrow \{0, 1, \ldots, |v|-1\}$ be a function defined as follows.

  $$g(l) = \begin{cases} l, & l < i + j \\ l - j, & l \geq i + j \end{cases}$$

  We claim that for every $l < i + j$ we have that $w_l \prec_{m-1,n} v_{g(l)}$. Indeed, for $l < i$ it follows from Lemma 2.2 (i), and for $i \leq l <$

$i + j$ it is due to Lemma 2.2 (iii). For every $l \geq i + j$ the words $v_{g(l)}$ and w_l are the same. Hence, for every $l < |w|$ we have that $w_l \models \psi \iff v_{g(l)} \models \psi$ and the same can be proven for ρ (in the same way).

Now we show that if $w \models \psi \mathsf{U} \rho$ then also $v \models \psi \mathsf{U} \rho$. If $w \models \psi \mathsf{U} \rho$, there is $c \geq 0$ such that $w_c \models \rho$ and for every $d < c$ we have that $w_d \models \psi$. Then $v_{g(c)} \models \rho$ (see above) and from the definition of g it follows that for every $d' < g(c)$ there is $d < c$ such that $g(d) = d'$. Thus, for every $d' < g(c)$ we have that $v_{g(d)} = v_{d'} \models \psi$. To sum up, we obtain that $v \models \psi \mathsf{U} \rho$.

Similarly, we also show that if $v \models \psi \mathsf{U} \rho$ then $w \models \psi \mathsf{U} \rho$. If $v \models \psi \mathsf{U} \rho$, there is $c \geq 0$ such that $v_c \models \rho$ and for every $d < c$ we have that $v_d \models \psi$. Let c' be the least number satisfying $g(c') = c$ (there is such a c' as the function g is surjective). Then $w_{c'} \models \rho$ (see above). From the definition of g we get that for every $d' < c'$ it holds that $g(d') < g(c') = c$ (otherwise we would obtain a contradiction with our choice of c'). Thus, $w_{d'} \models \psi$ and hence $w \models \psi \mathsf{U} \rho$.

- φ is a 'Boolean combination' of formulae of the previous cases. Formally, this case is handled by an 'embedded' induction on the structure of φ. The basic step (when φ is *not* of the form $\neg\psi$ or $\psi_1 \wedge \psi_2$) is covered by the previous cases. The induction step ($\varphi \equiv \neg\psi$ or $\varphi \equiv \psi_1 \wedge \psi_2$ where we assume that our theorem holds for ψ, ψ_1, ψ_2) follows immediately. $\square$

A natural question suggested by Theorem 2.4 is whether it also holds vice versa, i.e., if every regular (m, n)-stutter closed language is definable by an LTL$(\mathsf{U}^m, \mathsf{X}^n)$ formula. We must reject this hypotheses, as shown by the following counterexample:

Example 2.5. *Let* $\Sigma = \{p, q, r\}$ *and* $\varphi = \neg r \wedge ((p \,\mathsf{U}\, q) \,\mathsf{U}\, r)$. *It is easy to see that the language* $L_\varphi = \{p, q\}^* qr\Sigma^*$ *is* $(1, 0)$*-stutter closed. We prove that* L_φ *is not definable in* $\mathrm{LTL}(\mathsf{U}^1, \mathsf{X}^0)$. *To do that, it suffices to show that for any* $\psi \in \mathrm{LTL}(\mathsf{U}^1, \mathsf{X}^0)$ *we have that* $pqr \models \psi \iff pqpr \models \psi$ *(observe that* $pqr \in L_\varphi$ *and* $pqpr \notin L_\varphi$*). There are three cases.*

- $\psi \in \mathrm{LTL}(\mathsf{U}^0, \mathsf{X}^0)$. *Since the validity of* $\psi \in \mathrm{LTL}(\mathsf{U}^0, \mathsf{X}^0)$ *formulae depends only on the first letter of a given word, we are done.*

- $\psi = \psi_1 \,\mathsf{U}\, \psi_2$, *where* $\psi_1, \psi_2 \in \mathrm{LTL}(\mathsf{U}^0, \mathsf{X}^0)$. *Let* $S_1 = \{o \in \Sigma \mid o \models \psi_1\}$ *and* $S_2 = \{o \in \Sigma \mid o \models \psi_2\}$. *For every* $w \in \Sigma^*$ *we have that* $w \models \psi_1 \,\mathsf{U}\, \psi_2$ *iff there exists* $j \in \mathbb{N}_0$ *such that* $w(j) \in S_2$ *and for all* $0 \leq i < j$ *we have that* $w(i) \in S_1$. *One can easily check that the words* pqr, $pqpr$ *cannot be distinguished by the just specified condition for any* $S_1, S_2 \subseteq \Sigma$.

- ψ *is a 'Boolean combination' of formulae of the previous cases. Here we argue by a (straightforward) structural induction.* $\square$

Nevertheless, we can easily prove the following alternative characterization of LTL languages:

Corollary 2.6. *A regular language* L *over* Σ *is definable in LTL iff* L *is* (m, n)*-stutter closed for some* $m, n \in \mathbb{N}_0$.

Proof. The '$\Longrightarrow$' direction follows immediately from Theorem 2.4. The other direction is a simple consequence of the fact that a regular language L is expressible in LTL iff the minimal deterministic automaton recognizing L is counter-free [Kam68, MP71]—one can easily argue that if the minimal deterministic automaton for L is *not* counter-free, then L cannot be (m, n)-stutter closed for any $m, n \in \mathbb{N}_0$. (Indeed, if the minimal automaton is not counter-free then there are $u, w, v \in \Sigma^*$, $|w| \geq 1$, and $k \geq 2$ such that for each $i \in \mathbb{N}_0$ we have that $uw^{k \cdot i}v \in L$ and $uw^{k \cdot i - 1}v \notin L$. Now suppose that L is (m, n)-stutter closed for some $m, n \in \mathbb{N}_0$ and let $d = (m+n+2)$.

Since the first copy of w in $uw^{k \cdot d}v \in L$ is (m, n)-redundant, we have a contradiction with $uw^{k \cdot d-1}v \notin L$.) $\square$

In the context of Corollary 2.6, the evidence provided by Example 2.5 cannot be seen as fully satisfactory—since every regular language L which is (m, n)-stutter closed for some $m, n \in \mathbb{N}_0$ is definable in LTL, there surely exist $m', n' \in \mathbb{N}_0$ such that L is definable in $\mathrm{LTL}(\mathsf{U}^{m'}, \mathsf{X}^{n'})$. Due to Example 2.5 we know that the relationship among m, n and m', n' is not purely $m' = m$ and $n' = n$. Maybe the actual relationship is just slightly more complicated; and maybe there is no direct connection at all. Example 2.5 does not contradict any of the two hypothesis.

It is worth noting that even special cases of Theorem 2.4 can bring interesting consequences. For example, we already mentioned the well-known form of stuttering admitted by $\mathrm{LTL}(\mathsf{X}^0)$ formulae which can be generalized to $\mathrm{LTL}(\mathsf{X}^n)$ formulae. Formally, for each $w \in \Sigma^*$ and $n \in \mathbb{N}_0$ we define the n-*canonical form* of w, denoted $[w]_n$, which is the word obtained from w by deleting all n-redundant letters (see above). Two words $w, v \in \Sigma^*$ are n-*stutter equivalent* iff $[w]_n = [v]_n$. Observe that 0-stutter equivalence is exactly the well-known stutter equivalence of $\mathrm{LTL}(\mathsf{X}^0)$, and that n-stutter equivalence is subsumed by $\approx_{m,n}$ for each $m \in \mathbb{N}_0$ (indeed, it suffices to realize that $w(i, 1)$ is an (m, n)-redundant subword iff $w(i)$ is an n-redundant letter). Hence, a direct corollary to Theorem 2.4 is

Corollary 2.7. *Let* $\varphi \in \mathrm{LTL}(\mathsf{X}^n)$ *where* $n \in \mathbb{N}_0$. *The language* L_φ *is closed under* n*-stutter equivalence.*

Of course, a direct proof of this corollary is a bit simpler than the proof of Theorem 2.4. However, it already brings interesting consequences.

Corollary 2.8. *The property* $\mathsf{G}_2 p$ *(which says 'at every even position is* p*') is not expressible in LTL.*

Proof. Suppose the converse. Let $\Sigma = \{p, q\}$. As $\mathsf{G}_2 p$ is expressible in LTL, there is $n \in \mathbb{N}_0$ and a formula $\varphi \in \mathrm{LTL}(\mathsf{X}^n)$ which is equivalent to G_2p. Since L_φ contains the word $p^{2n+2}q$ and the first occurrence of p is n-redundant, we obtain $p^{2n+1}q \in L_\varphi$ which is a contradiction. $\square$

Another application for Corollary 2.7 will be given in Section 3. So, a direct proof of Corollary 2.7 might be of some use even in a basic course on LTL, because it is not much longer than a proof for 0-stutter equivalence (which is often included) and it brings interesting consequences 'for free'.

3 Answers for Questions 1, 2, and 3

Now we are ready to provide answers to Questions 1, 2, and 3 which were stated in Section 1 (though the Question 3 will be left open in fact). We start with a simple observation.

Lemma 3.1. *For each* $n \geq 1$ *there is a formula* $\varphi \in \mathrm{LTL}(\mathsf{U}^0, \mathsf{X}^n)$ *which cannot be expressed in* $\mathrm{LTL}(\mathsf{X}^{n-1})$.

Proof. Let $\Sigma = \{p\}$ and $n \geq 1$. Consider the formula $\varphi \equiv \overbrace{\mathsf{X}\mathsf{X}\cdots\mathsf{X}}^{n}p$. We show that L_φ is not closed under $(n-1)$-stutter equivalence (which suffices due to Corollary 2.7). It is easy; realize that $p^{n+1} \in L_\varphi$ and the first occurrence of p in this word is $(n-1)$-redundant. Since $p^n \notin L_\varphi$, we are done. $\square$

A 'dual' fact is proven below (it is already non-trivial).

Lemma 3.2. *For each* $m \geq 1$ *there is a formula* $\varphi \in \mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^0)$ *which cannot be expressed in* $\mathrm{LTL}(\mathsf{U}^{m-1})$.

Proof. Let $m \geq 1$ and let $\Sigma = \{q, p_1, \ldots, p_m\}$. We define a formula $\varphi \in \mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^0)$ as follows:

$$\varphi = \mathsf{F}(p_1 \wedge \mathsf{F}(p_2 \wedge \cdots \wedge \mathsf{F}(p_{m-1} \wedge \mathsf{F}p_m)\ldots))$$

Let us fix an arbitrary $n \in \mathbb{N}_0$, and define a word $w \in \Sigma^*$ by

$$w = (q^{n+1}\, p_m\, p_{m-1} \dots p_1)^m\, q^{n+1}$$

Clearly $w \models \varphi$ and the subword $w(0, n{+}1{+}m)$ is $(m{-}1, n)$-redundant. As the word w' obtained from w by removing $w(0, n{+}1{+}m)$ does not model φ, the language L_φ is not $(m{-}1, n)$-stutter closed. As it holds for every $n \in \mathbb{N}_0$, the formula φ is not expressible in $\mathrm{LTL}(\mathsf{U}^{m-1})$. □

The last technical lemma which is needed to formulate answers to Questions 1 and 2 follows.

Lemma 3.3. *For all* $m, n \in \mathbb{N}_0$ *there is a formula* $\varphi \in \mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$ *which is expressible neither in* $\mathrm{LTL}(\mathsf{U}^{m-1}, \mathsf{X}^n)$ *(assuming* $m \geq 1$*), nor in* $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^{n-1})$ *(assuming* $n \geq 1$*).*

Proof. If $m = 0$ or $n = 0$, we can apply Lemma 3.1 or Lemma 3.2, respectively. Now let $m, n \geq 1$, and let $\Sigma = \{p_1, \dots, p_k\}$ where $k = \max\{m, n{+}1\}$. We define formulae ψ and φ as follows:

$$\psi = \begin{cases} p_m \wedge \mathsf{X}^n p_{m-n} & \text{if } m > n \\ p_m \wedge \mathsf{X}^n p_{m+1} & \text{if } m \leq n \end{cases}$$

$$\varphi = \begin{cases} \mathsf{F}\psi & \text{if } m = 1 \\ \mathsf{F}(p_1 \wedge \mathsf{F}(p_2 \wedge \mathsf{F}(p_3 \wedge \dots \wedge \mathsf{F}(p_{m-1} \wedge \mathsf{F}\psi) \dots))) & \text{if } m > 1 \end{cases}$$

where X^l abbreviates $\overbrace{\mathsf{XX}\dots\mathsf{X}}^{l}$. The formula φ belongs to $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$. Let us consider the word w defined by

$$w = \begin{cases} (p_m\, p_{m-1} \dots p_1)^m p_m\, p_{m-1} \dots p_{m-n+1} & \text{if } m > n \\ (p_{n+1}\, p_n \dots p_1)^{m+1} & \text{if } m = n \\ (p_{n+1}\, p_n \dots p_1)^{m+1} p_{n+1}\, p_n \dots p_{m+2} & \text{if } m < n \end{cases}$$

It is easy to check that $w \in L_\varphi$ and the subword $w(0, k)$ (where $k = \max\{m, n{+}1\}$) is $(m, n{-}1)$-redundant as well as $(m{-}1, n)$-redundant. As the word w' obtained from w by removing $w(0,k)$ does not satisfy φ, the language L_φ is neither $(m, n{-}1)$-stutter closed, nor $(m{-}1, n)$-stutter closed. $\square$

The knowledge presented in the three lemmata above allows to conclude the following:

Corollary 3.4 (Answer to Question 1). *The* LTL$(\mathsf{U}^m, \mathsf{X}^n)$, LTL$(\mathsf{U}^m)$, *and* LTL$(\mathsf{X}^n)$ *hierarchies are strict.*

Corollary 3.5 (Answer to Question 2). *Let* A *and* B *be classes of* LTL$(\mathsf{U}^m, \mathsf{X}^n)$, LTL$(\mathsf{U}^m)$, *or* LTL$(\mathsf{X}^n)$ *hierarchy (not necessarily of the same one) such that* A *is syntactically not included in* B. *Then there is a formula* $\varphi \in A$ *which cannot be expressed in* B.

Although we cannot provide a full answer to Question 3, we can at least reject the aforementioned 'natural' hypotheses (see Section 1).

Lemma 3.6 (About Question 3). *For all* $m, n \in \mathbb{N}_0$ *there is a language definable in* LTL$(\mathsf{U}^{m+2}, \mathsf{X}^n)$ *as well as in* LTL$(\mathsf{U}^{m+1}, \mathsf{X}^{n+1})$ *which is not definable in* LTL$(\mathsf{U}^{m+1}, \mathsf{X}^n)$.

Proof. We start with the case when $m = n = 0$. Let $\Sigma = \{p, q\}$, and let $\psi_1 = \mathsf{F}(q \wedge (q \,\mathsf{U}\, \neg q))$ and $\psi_2 = \mathsf{F}(q \wedge \mathsf{X}\neg q)$. Note that $\psi_1 \in$ LTL$(\mathsf{U}^2, \mathsf{X}^0)$ and $\psi_2 \in$ LTL$(\mathsf{U}^1, \mathsf{X}^1)$. Moreover, ψ_1 and ψ_2 are equivalent as they define the same language $L = \Sigma^* q (\Sigma \smallsetminus \{q\}) \Sigma^*$. This language is not definable in LTL$(\mathsf{U}^1, \mathsf{X}^0)$ as it is not $(1,0)$-stutter closed; for example, the word $w = pqpq \in L$ contains a $(1,0)$-redundant subword $w(0,2)$ but $w_2 = pq \notin L$.

The above example can be generalized to arbitrary m, n (using the designed formulae ψ_1, ψ_2). For given m, n we define formulae $\varphi_1 \in$ LTL$(\mathsf{U}^{m+2}, \mathsf{X}^n)$ and $\varphi_2 \in$ LTL$(\mathsf{U}^{m+1}, \mathsf{X}^{n+1})$, both defining the same language L over $\Sigma = \{q, p_1, \ldots, p_{m+1}\}$, and we give an example of a word $w \in L$ with an $(m+1, n)$-redundant subword such that w without this subword is not from L. We distinguish three cases.

- $m = n > 0$. For $i \in \{1,2\}$ we define

$$\varphi_i = \overbrace{\mathsf{XF}(p \wedge \mathsf{XF}(p \wedge \mathsf{XF}(p \wedge \cdots \wedge \mathsf{XF}(p\wedge}^{m\text{-times}} \psi_i) \dots)))$$

  The word $w = (pq)^{m+2} \in L$, $w(0,2)$ is $(m+1,n)$-redundant, and $w_2 = (pq)^{m+1} \notin L$.

- $m > n$. For $i \in \{1,2\}$ we define

$$\xi_i = \overbrace{\mathsf{F}(p_1 \wedge \mathsf{F}(p_2 \wedge \cdots \wedge \mathsf{F}(p_{m-n}\wedge}^{(m-n)\text{-times}} \psi_i) \dots)))$$

$$\varphi_i = \overbrace{\mathsf{XF}(q \wedge \mathsf{XF}(q \wedge \cdots \wedge \mathsf{XF}(q\wedge}^{n\text{-times}} \xi_i) \dots)))$$

  The word $w = (qp_{m-n}p_{m-n-1} \dots p_1)^{m+1}q \in L$, $w(0, m-n+1)$ is $(m+1,n)$-redundant, and $w_{m-n+1} \notin L$.

- $m < n$. For $i \in \{1,2\}$ we define

$$\varphi_i = \overbrace{\mathsf{F}(p_1 \wedge \mathsf{F}(p_2 \wedge \cdots \wedge \mathsf{F}(p_m\wedge}^{m\text{-times}} \overbrace{\mathsf{XX} \dots \mathsf{X}}^{n} \psi_i) \dots))$$

  The word $w = (q^{n-m}p_{m+1}p_m \dots p_1)^{m+2}q^{n-m} \in L$, $w(0,n+1)$ is $(m+1,n)$-redundant, and $w_{n+1} \notin L$. $\square$

In fact, the previous lemma says that if we take two classes LTL$(\mathsf{U}^{m_1}, \mathsf{X}^{n_1})$ and LTL$(\mathsf{U}^{m_2}, \mathsf{X}^{n_2})$ which are syntactically incomparable and where $m_1, m_2 \geq 1$, then their semantical intersection is strictly greater than LTL$(\mathsf{U}^m, \mathsf{X}^n)$ where $m = \min\{m_1, m_2\}$ and $n = \min\{n_1, n_2\}$. Moreover, it also says that if we try to minimize the nesting depths of X and U in a given formula φ (preserving the meaning of φ), there is generally no 'best' way how to do that.

4 A note on model-checking with LTL

The aim of this section is to identify another (potential) application of Theorem 2.4 in the area of model-checking with the logic LTL. We show that the theorem can be used as a 'theoretical basis' for advanced state-space reduction techniques which might further improve the efficiency of LTL model-checking algorithms. The actual development of such techniques is a complicated problem beyond the scope of this paper; nevertheless, we can explain the basic principle, demonstate its potential power, and explicitly discuss the missing parts which must be completed to obtain a working implementation. The chosen level of presentation is semi-formal, and the content is primarily directed to a 'practically-oriented' reader.

The model-checking approach to formal verification (with the logic LTL) works according to the following abstract scheme:

- The verified system is formally described in a suitable modeling language whose underlying semantics associates a well-defined Kripke structure to the constructed model.
- Desired properties of the system are defined as a formula in the logic LTL. More precisely, one defines the properties which should be satisfied by all possible *runs* of the system, which formally correspond to certain maximal paths in the associated Kripke structure.
- It is shown that all runs satisfy the constructed LTL formula.

A principal difficulty is that the size of the associated Kripke structure is usually very large (this is known as the problem of state-space explosion). There are various strategies how to deal with this problem. For example, one can reduce the number of states by abstracting the code and/or the data of the system, use various 'compositional' techniques, or use restricted formalisms (like, e.g., pushdown automata) which allow for a kind of 'symbolic' model-checking where the explicit construction of the associated Kripke structure is not required. One of the most successful methods is *partial order reduction* (see, e.g., [CGP99]) which works for the LTL(X^0) fragment of LTL. It has been argued by Lamport [Lam83] that LTL(X^0) provides a sufficient expressive power for specifying correctness properties of software systems; one should avoid the use of the X operator because it imposes very strict requirements on 'scheduling' of transitions between states which can be hard to implement. Partial order reduction conveniently uses the *stutter invariance* of LTL(X^0) formulae in the sense of Corollary 2.7. Roughly speaking, the idea is as follows: if we are to decide the validity of a given LTL(X^0) formula for a given Kripke structure, we do not necessarily need to examine *all* runs; we can safely ignore those runs which are 0-stutter equivalent to already checked ones. To see how it works in practice, consider the following parallel programme consisting of two threads `A` and `B`.

```
x = 0;          procedure A()               procedure B()
cobegin           begin                       begin
  A; B;             for i=1 to 5 do             z = 2;
coend                 begin                     x = x + 7;
                        x = x + 1;              z = 2 * z;
                        x = x - 1;              z = z - 1;
                      end                     end
                  end
```

The underlying Kripke structure (see Fig. 1) models all possible interleavings between `A` and `B`. The states carry the information about variables and about the position of control in the two threads. The transitions correspond to individual instructions. In Fig. 1, we explicitly indicated the value of `x` in each state; the $\swarrow$ direction corresponds to instructions of `A`, and the $\searrow$ direction corresponds to instructions of `B`. Now imagine that we want to

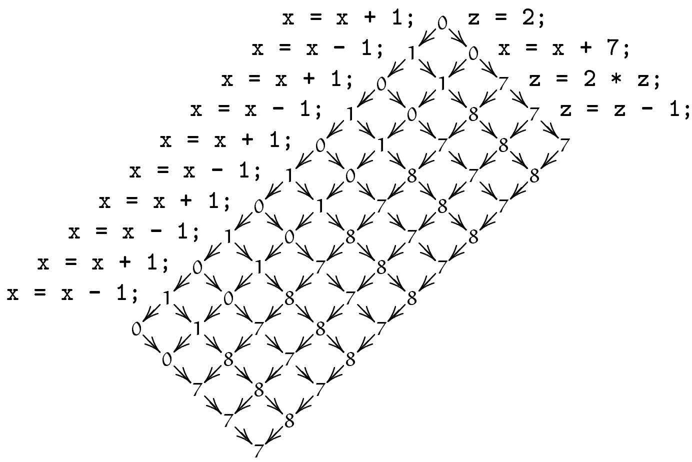

Figure 1: The associated Kripke structure.

verify that x is always strictly less than 8 at every run (which is not true). It can be formally expressed by a formula $\mathsf{G}(\mathtt{x} < 8)$ where the predicate $\mathtt{x} < 8$ should be seen as a *letter* (in the sense of LTL semantics given in Section 1). Hence, to every run we can associate a word over the alphabet $\{\mathtt{x} < 8, \neg(\mathtt{x} < 8)\}$ and interpret our formula in the standard way. Since the values of all variables except for x are irrelevant, the instructions which do not modify the value of x always generate 0-redundant letters (while, for example, the instruction `x = x + 1` sometimes generates a redundant letter and sometimes not). Hence, many of the runs in Fig. 1 are in fact 0-stutter equivalent and hence one can safely 'ignore' many of them. Technically, a set of runs can be ignored by ignoring certain out-going transitions in certain states; and since we ignore some transitions, it can also happen that some states are not visited at all—and thus we could in principle avoid their construction, keeping the Kripke structure smaller. The question is how to recognize those superfluous transitions and states. It does not make much sense to construct the whole Kripke structure and then try to reduce it; what we need is a method which can be applied *on-the-fly* while constructing the Kripke structure. *Partial-order reduction* (as described in [CGP99]) can do the job fairly well—if we apply it to the structure of Fig. 1 and the

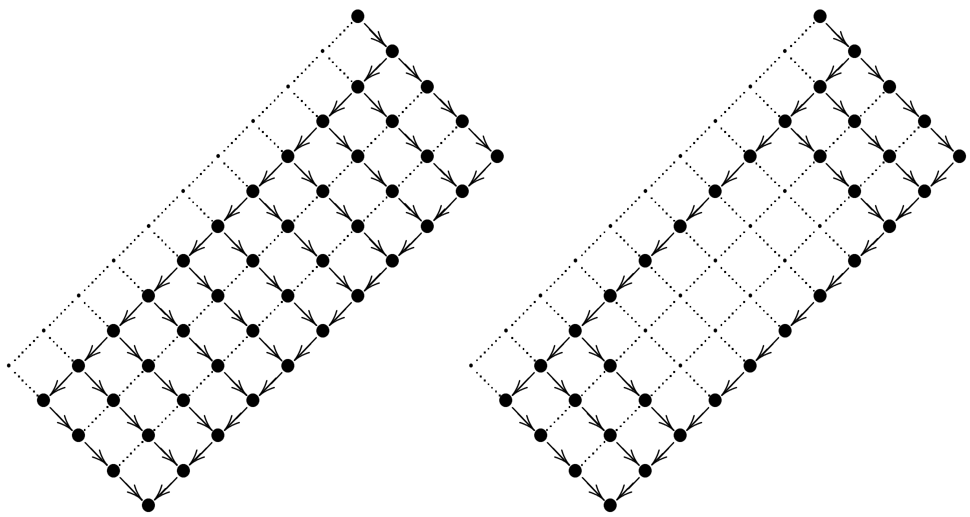

Figure 2: The reduced Kripke structures.

formula $\mathsf{G}(\mathtt{x} < 8)$, we obtain a 'pruned' structure of Fig. 2 (left)[1]. Now we come to the actual point of this section—since $\mathsf{G}(\mathtt{x} < 8)$ is an LTL$(\mathsf{U}^1, \mathsf{X}^0)$ formula, we can also apply the principle of $(1, 0)$-stuttering which allows to 'ignore' even more runs in the Kripke structure of Fig. 1 (many of them are $(1, 0)$-stutter equivalent). One of possible results is shown in Fig. 2 (right)[2]; it clearly demonstrates the potential power of the new method. However, it is not clear if the method admits an on-the-fly implementation, which means that we cannot fully advocate its practical usability at the moment. This question is left open as another challenge.

To sum up, we believe that (m, n)-stuttering might be (potentially) used as the underlying principle for optimized model-checking in a similar fashion as 0-stuttering was used in the case of partial-order reduction. However, it can only be proven by designing a working and efficient on-the-fly reduction method, which is a non-trivial research problem on its own.

[1]The instructions which modify the variable $\mathtt{x}$ are treated as 'dangerous', i.e., as if they never produced redundant letters.

[2]To give a 'fair' comparison with partial-order reduction, the instructions which modify the variable $\mathtt{x}$ are again treated as 'dangerous'.

5 Conclusions

The main technical contribution of this paper is the general stuttering theorem presented in Section 2. With its help we were able to construct (short) proofs of other results. In particular, we gave an alternative characterization of LTL languages (which are exactly regular (m, n)-stutter closed languages), proved the strictness of the three hierarchies of LTL formulae introduced in Section 1, and we also showed several related facts about the relationship among the classes in the three hierarchies.

Some problems are left open. For example, the exact characterization of the semantical intersection of $LTL(U^{m_1}, X^{n_1})$ and $LTL(U^{m_2}, X^{n_2})$ classes (in the case when they are syntactically incomparable) surely deserves further attention. Moreover, we would be also interested if the potential applicability of Theorem 2.4 to model-checking (as indicated in Section 4) can really result in a practically usable state-space reduction method.

References

[CGP99] E.M. Clark, O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, 1999.

[EW00] K. Etessami and T. Wilke. An until hierarchy and other applications of an Ehrenfeucht-Fraïssé game for temporal logic. *Information and Computation*, 160:88–108, 2000.

[Kam68] H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, UCLA, 1968.

[Lam83] L. Lamport. What good is temporal logic? In *Proceedings of IFIP Congress on Information Processing*, pages 657–667, 1983.

[MP71] R. McNaughton and S. Papert. *Counter-Free Automata*. The MIT Press, 1971.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, 1977.

[Tho91] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, B:135–192, 1991.

[Wil99] T. Wilke. Classifying discrete temporal properties. In *Proceedings of STACS'99*, volume 1563 of *LNCS*, pages 32–46. Springer, 1999.

Publications in the FI MU Report Series are in general accessible via WWW and anonymous FTP:

```
http://www.fi.muni.cz/informatics/reports/
ftp  ftp.fi.muni.cz (cd pub/reports)
```

Copies may be also obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic