

Master's Thesis

V.R. Sathiyararayanan ✉🏠

Chennai Mathematical Institute, India

Abstract

My master's thesis!

2012 ACM Subject Classification Theory of computation → Modal and temporal logics

Keywords and phrases Logic, Modal and temporal logic

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Acknowledgements I want to thank my advisors Dr. Paul Gastin and Dr. Aiswarya Cyriac for helping me through this project

1 Introduction

The separation property, invented by Dov Gabbay, is a strangely influential consequence of the design of popular temporal languages. Simply put, it requires all formulas in the language to be equivalent to a variant made up of formulas purely concerned with one *region* of the flow of time. Surprisingly, this property is linked to expressive completeness: a sufficiently expressive temporal language with the separation property can express any first-order property. In the next few sections, we will detail the separation property over linear time, its consequences on functional completeness, and discuss the generalization described in [3].

2 Preliminaries

Before discussing separation, we define some standard notions. A flow of time is simply a non-empty set T partially ordered by the binary relation $<$. We symbolically refer to these flows by the pair $(T, <)$. Examples include $(\mathbb{N}, <)$ and $(\mathbb{R}, <)$ with their natural ordering, unordered trees with the descendant relation, and Mazurkiewicz traces. We will consider the truth values of propositions (from a fixed set \mathcal{P}) at points on these flows.

The first-order vocabulary over these structures contains the ordering relation $<$ and a collection of *monadic* relations Q_1, Q_2, \dots that match the propositions q_1, q_2, \dots in \mathcal{P} . An assignment h of atoms in a time flow $(T, <)$ assigns to each Q_i a subset of T where the atom q_i is true. Augmented with the assignment, the triplet $(T, <, h)$ is called a *temporal structure*. First-order formulas are evaluated over these structures in the usual way. In this discussion, we pay special attention to first-order formulas with a single free-variable; they quite naturally mirror temporal formulas.

Instead of free variables and quantification, temporal languages employ *connectives* to reason through time. Popular connectives include F , P , G , H , U , and S , known as *future*, *past*, *globally*, *history*, *until* and *since* respectively. In this paper, we will limit our discussion to connectives that are definable by monadic first-order formulas.

Temporal formulas are evaluated at points in time. In a temporal structure $\mathcal{M} = (T, <, h)$, atoms are evaluated as

$$\mathcal{M}, t \models p \iff (T, <, h[x \mapsto t]) \models p(x) \iff t \in h(p)$$

As per the standard notation, the assignment $h[x \mapsto t]$ assigns the time point t to the first-order variable x . For a generic connective \sharp of arity n , let $\varphi_\sharp(t, X_1, \dots, X_n)$ be the



© Jane Open Access and Joan R. Public;
licensed under Creative Commons License CC-BY 4.0

42nd Conference on Very Important Topics (CVIT 2016).

Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:8



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

monadic first-order formula defining it. Here, t is the point in time that the connective is evaluated at, and the X_i are monadic (second-order) variables. These variables expect a single-variable first-order formula, as shown below

$$\mathcal{M}, t \models \sharp(A_1, \dots, A_n) \iff (T, <, h[x \mapsto t]) \models \varphi_{\sharp}(x, \alpha_{A_1}, \dots, \alpha_{A_n})$$

Here, A_i are temporal formulas and α_{A_i} are their first-order translations. Notably, φ_{\sharp} can only quantify over elements in the domain T ; it cannot use second order quantifiers.

We illustrate this behaviour with an example. The connective F is defined by the formula

$$\varphi_F(t, X) \triangleq \exists x. (t < x) \wedge X(x)$$

Hence, we have

$$\mathcal{M}, t \models Fp \iff (T, <, h[x \mapsto t]) \models \exists y. (x < y) \wedge p(y)$$

We similarly define the other main connectives

$$\begin{aligned} \varphi_P(t, X) &\triangleq \exists x. (x < t) \wedge X(x) \\ \varphi_G(t, X) &\triangleq \forall x. (t < x) \wedge X(x) \\ \varphi_H(t, X) &\triangleq \forall x. (x < t) \wedge X(x) \\ \varphi_U(t, X_1, X_2) &\triangleq \exists x. [(t < x) \wedge X_1(x) \wedge \forall y ((t < y < x) \rightarrow X_2(y))] \\ \varphi_S(t, X_1, X_2) &\triangleq \exists x. [(x < t) \wedge X_1(x) \wedge \forall y ((x < y < t) \rightarrow X_2(y))] \end{aligned}$$

Note that, unlike the typical definition of U , φ_U doesn't rely on the present point t . Such an until is referred to in the literature by either the *strict* until (see [2]) or the *strong* until (see [1]). This particular behaviour makes observing separation much easier.

► **Definition 1** (Expressive Completeness). *A temporal language is **first-order expressively complete** over a class of time flows iff there exists a temporal formula A for any first-order formula with one free variable $\varphi(t)$ such that*

$$\mathcal{M}, t \models A \iff \mathcal{M}[x \mapsto t] \models \varphi(x)$$

for any flow \mathcal{M} in the class.

On a related note, a flow of time $(T, <)$ is termed to be expressively complete if there exists an expressively complete temporal language over it.

3 Linear Flows

In [3], Gabbay showed how the temporal language **L** with the strict until U and since S connectives satisfies the separation property over the integer time flow $(\mathbb{Z}, <)$.

To discuss this further, we need the notion of *regions* and *pure formulas*. Informally, the flow of time $(T, <)$ is partitioned into a set of regions. The positions of these regions depends on the position of the time point t where the temporal formula is being evaluated. For linear flows, Gabbay selected three regions:

- The *past* of t , formally defined as $\{x \mid x \in \mathbb{Z} \wedge x < t\}$.
- The *present*, which is simply $\{t\}$.
- The *future* of t , which naturally is $\{x \mid x \in \mathbb{Z} \wedge t < x\}$

Note that these regions are disjoint, and that the union of these regions produces the entire flow. Also, notice that these regions are first-order definable.

Now, we define *pure formulas*. For any flow $(T, <)$, we denote two assignments h and h' to be in *agreement* over a region $R \subset T$ iff for any atom $q \in \mathcal{P}$ and any point $s \in R$,

$$s \in h(q) \iff s \in h'(q)$$

Now, call a temporal formula A *pure* with respect to a region R iff for any two assignments h and h' that agree on R ,

$$(T, <, h), t \models A \iff (T, <, h'), t \models A$$

In other words, A is true on h' iff A is true on h . We use the terms *pure past*, *pure present*, or *pure future* to denote pure formulas in the past, present, and future regions respectively.

Finally, call a formula A **separated** if it is a Boolean combination of pure formulas. Now, we can state the separation property

► **Theorem 2 (Separation Theorem).** *Every temporal formula A in the language of S and U over linear time can be equivalently represented by a separated formula.*

The proof of this theorem is quite involved, and is presented in full detail in [3]. In the next few sections, I'll give a high-level overview of Gabbay et. al.'s scheme. To mirror their notation, I'll write U formulas as $U(p, q)$ instead of $q\mathcal{U}p$.

3.1 Separating S and U over linear time

As a reminder, we restate the definitions of U and S

$$\begin{aligned} \mathcal{M}, t \models U(p, q) &\iff \mathcal{M}, t \models \exists x. (t < x) \wedge p(x) \wedge \forall y (t < y < x \rightarrow q(y)) \\ \mathcal{M}, t \models S(p, q) &\iff \mathcal{M}, t \models \exists x. (x < t) \wedge p(x) \wedge \forall y (x < y < t \rightarrow q(y)) \end{aligned}$$

For convenience, we refer to the left condition (p) in $U(p, q)$ as the *target* condition and the right condition (q) as the *path* condition. Observe that, over linear time, a formula composed only of U s is a pure future formula, a formula composed of S s is a pure past formula. The task, therefore, is to transform formulas with both U s and S es.

Over the integer time flow $(\mathbb{Z}, <)$, these connectives naturally possess the following properties

$$\begin{aligned} U(\alpha \vee \beta, \gamma) &\equiv U(\alpha, \gamma) \vee U(\beta, \gamma) \\ U(\alpha, \beta \wedge \gamma) &\equiv U(\alpha, \beta) \wedge U(\alpha, \gamma) \end{aligned} \tag{1}$$

In addition, their negations can be usefully rewritten as

$$\begin{aligned} \neg U(\alpha, \beta) &\equiv G(\neg\alpha) \vee U(\neg\alpha \wedge \neg\beta, \neg\alpha) \\ \neg S(\alpha, \beta) &\equiv H(\neg\alpha) \vee S(\neg\alpha \wedge \neg\beta, \neg\alpha) \end{aligned}$$

where the semantics of G and H are

$$\begin{aligned} \mathcal{M}, t \models G(\alpha) &\iff \mathcal{M}, t \models \forall t'. t' > t \rightarrow \varphi_\alpha(t') \\ \mathcal{M}, t \models H(\alpha) &\iff \mathcal{M}, t \models \forall t'. t' < t \rightarrow \varphi_\alpha(t') \end{aligned}$$

Here, φ_α is the first-order translation of α .

Our strategy involves *pulling-out* U s from inside S and vice versa. We accomplish this by writing all temporal formulas in a standard notation, and then applying a sequence of *elimination* rules. In the next section, we describe these rules.

3.1.1 Eliminations

Let α , β , φ and ψ be boolean combinations of propositional atoms. In the following subsections, we pull out a $U(\varphi, \psi)$ from inside a S under a variety of minimal configurations. In later sections, we show that these configurations suffice.

$$S(\alpha \wedge U(\varphi, \psi), \beta)$$

This formula requires $U(\varphi, \psi)$ to be true at a point t' in the past of t . This in turn implies φ at some point t'' ahead of t' . This naturally breaks down into three cases: $t'' > t$, $t'' = t$, and $t' < t'' < t$. The translation is

$$\begin{aligned} & S(\varphi \wedge \beta \wedge S(\alpha, \psi \wedge \beta), \beta) \\ \vee & (S(\alpha, \psi \wedge \beta) \wedge (\varphi \vee (\psi \wedge U(\varphi, \psi)))) \end{aligned}$$

$$S(\alpha \wedge \neg U(\varphi, \psi), \beta)$$

In this case, we immediately rewrite $\neg U(\varphi, \psi)$ as $G(\neg\alpha) \vee U(\neg\alpha \wedge \neg\beta, \neg\alpha)$. This gives us

$$\begin{aligned} & S(\alpha \wedge \neg U(\varphi, \psi), \beta) \equiv \\ & S(\alpha \wedge G(\neg\varphi), \beta) \\ \vee & S(\alpha \wedge U(\neg\varphi \wedge \neg\psi, \neg\varphi), \beta) \end{aligned}$$

where each individual case can be translated using the ideas used to rewrite $S(\alpha \wedge U(\varphi, \psi), \beta)$.

$$S(\alpha, U(\varphi, \psi))$$

It's instructive to recognize how $S(\alpha, U(\varphi, \psi))$ could be translated. Unlike the previous cases, the Until fragment needs to be true at each point in the path to α . This could involve multiple segments in this path where ψ is true till φ is true. Wonderfully, this is *indistinguishable* from the case where, at each point in the path, either φ or ψ is true. This formula is translated to

$$\begin{aligned} & S(\alpha, \perp) \\ \vee & S(\alpha, \varphi \vee \psi) \wedge [\varphi \vee (\psi \wedge U(\varphi, \psi))] \end{aligned}$$

Here, $S(\alpha, \perp)$ can only be true if α is true at the previous point. Otherwise, we'll need $U(\varphi, \psi)$ to be satisfied at the previous location, hence the $\varphi \vee (\psi \wedge U(\varphi, \psi))$ at the present. At each point t' in the path to α , if $t' + 1 \models \varphi$, $t' \models U(\varphi, \psi)$. Otherwise, $t' + 1 \models \psi$. At this point, we can use an inductive argument, starting from the previous point, to prove the correctness of this translation.

$$S(\alpha, \beta \vee U(\varphi, \psi))$$

The idea is to attempt to enforce $U(\varphi, \psi)$ at each point in the path *iff* we can detect an earlier point in the path which needed to satisfy it. A simple way to detect these points is to look for the moment where $\neg\beta$ was true, and check whether, along the way to that point, $\neg\varphi$ was true at each step. Accordingly, $S(\neg\beta \wedge \neg\alpha, \neg\varphi \wedge \neg\alpha)$ does the trick. Here, the $\neg\alpha$ is to ensure that we specifically look for points in the future of α , the leftmost point in our consideration.

It's important to recognize that we are capable of recognizing such points at each step of the path to α . This means that, if we recognized such a point that's 3 steps away, we recognized it at 2 and 1 step away too. This allows us a simple fix: $S(\neg\beta, \neg\varphi \wedge \neg\alpha) \rightarrow \varphi \vee \psi$.

If φ was true, we will not see this point in our next search. Otherwise, ψ would be true, allowing for the possibility of enforcement in the future.

The overall translation now is

$$\begin{aligned} & S(\alpha, \neg\alpha \wedge (S(\neg\beta \wedge \neg\alpha, \neg\varphi \wedge \neg\alpha) \rightarrow \varphi \vee \psi)) \\ \wedge & S(\neg\beta \wedge \neg\alpha, \neg\varphi \wedge \neg\alpha) \rightarrow (\varphi \vee (\psi \wedge U(\varphi, \psi))) \end{aligned}$$

$$S(\alpha, \beta \vee \neg U(\varphi, \psi))$$

This case is very similar to the previous case. The points we search for must be in danger of satisfying $U(\varphi, \psi)$; hence, we look for $S(\neg\beta \wedge \neg\alpha, \psi \wedge \neg\alpha)$. We fix these points by requiring φ to be false. In the worst-case, we've dragged on the possible *until* to the present, at which point we can extinguish all hope. This gives us the overall translation:

$$\begin{aligned} & S(\alpha, \neg\alpha \wedge (S(\neg\beta \wedge \neg\alpha, \psi \wedge \neg\alpha) \rightarrow \neg\varphi)) \\ \wedge & S(\neg\beta \wedge \neg\alpha, \psi \wedge \neg\alpha) \rightarrow ((\neg\psi \wedge \neg\varphi) \vee (\neg U(\varphi, \psi))) \end{aligned}$$

$$S(\alpha \wedge U(\varphi, \psi), \beta \vee U(\varphi, \psi))$$

This is a neat combination of $S(\alpha \wedge U(\varphi, \psi), \beta)$ and $S(\alpha, \beta \vee U(\varphi, \psi))$. The translation is simple. *I believe Gabbay made a typo in this particular example. [4] mentions this.*

$$\begin{aligned} & S(\alpha, \psi) \wedge (\varphi \vee (\psi \wedge U(\varphi, \psi))) \\ \vee & S(\varphi \wedge S(\alpha, \psi), S(\neg\beta, \neg\varphi) \rightarrow \varphi \vee \psi) \\ \wedge & S(\neg\beta, \neg\varphi) \rightarrow (\varphi \vee (\psi \wedge U(\varphi, \psi))) \end{aligned}$$

3.1.2 Putting it all together

The eliminations presented in the previous section lend credence to the idea of separation. Amazingly, Gabbay presents a neat induction scheme that builds on these rules to separate *any* temporal formula in the language. In this section, we present an overview of his arguments (presented in more detail in [3]).

► **Lemma 3.** *Let φ and ψ be pure-present formulas and α and β be formulas such that the only appearance of a U in either of them is $U(\varphi, \psi)$, and that U isn't nested inside a S . Then $S(\alpha, \beta)$ can be written as a syntactically separated formula where the only appearance of U is $U(\varphi, \psi)$.*

Proof. We start by writing α and β in their conjunctive and disjunctive normal forms respectively. During this transformation, we treat all top-level instances of U and S in them as atomic propositions. This gives us

$$\begin{aligned} \alpha &\equiv \bigvee_i (\alpha_{i,1} \wedge \alpha_{i,2} \wedge \cdots \wedge \alpha_{i,m_i}) \\ \beta &\equiv \bigwedge_j (\beta_{j,1} \vee \beta_{j,2} \vee \cdots \vee \beta_{j,n_j}) \end{aligned}$$

Here, the literals $\alpha_{i,k}$ and $\beta_{j,k}$ are composed of propositional atoms, S formulas, and $U(\varphi, \psi)$.

We use the above and equation (1) to write $S(\alpha, \beta)$ as

$$\begin{aligned}
S(\alpha, \beta) &\mapsto S\left(\bigvee_i (\alpha_{i,1} \wedge \cdots \wedge \alpha_{i,m_i}), \beta\right) \\
&\mapsto \bigvee_i S(\alpha_{i,1} \wedge \cdots \wedge \alpha_{i,m_i}, \beta) \\
&\mapsto \bigvee_i S\left(\alpha_{i,1} \wedge \cdots \wedge \alpha_{i,m_i}, \bigwedge_j (\beta_{j,1} \vee \cdots \vee \beta_{j,n_i})\right) \\
&\mapsto \bigvee_i \bigwedge_j S(\alpha_{i,1} \wedge \cdots \wedge \alpha_{i,m_i}, \beta_{j,1} \vee \cdots \vee \beta_{j,n_i})
\end{aligned}$$

In the resulting formula, the target of each top-level S is a conjunction of literals, and the path condition is a disjunction of literals. Notably, if $U(\varphi, \psi)$ doesn't appear in the target and the path of a top-level S formula, that subformula is a pure-past formula.

Hence, we focus our attention on the top-level S formulas containing $U(\varphi, \psi)$. In one such formula, let α' be the conjunction of all literals in the target that aren't $U(\varphi, \psi)$ or its negation. Similarly, let β' be the disjunction of all literals in the path that aren't $U(\varphi, \psi)$ or its negation. This lets us write that formula as one of the following

$$\begin{aligned}
&S(\alpha' \wedge \pm U(\varphi, \psi), \beta') \\
&S(\alpha', \beta' \vee \pm U(\varphi, \psi)) \\
&S(\alpha' \wedge \pm U(\varphi, \psi), \beta' \vee \pm U(\varphi, \psi))
\end{aligned}$$

Clearly, the eliminations we explored in the previous section can separate this formula! Additionally, note that the only U formula in the RHS of the eliminations is $U(\varphi, \psi)$, satisfying the condition specified in the beginning of the lemma.

Applying these elimination rules to each top-level S containing a $U(\varphi, \psi)$ produces a separated formula equivalent to $S(\varphi, \psi)$. This completes the proof. \blacktriangleleft

The second step of the induction scheme is to consider cases where $U(\varphi, \psi)$ is nested under multiple levels of S .

► **Lemma 4.** *Let φ and ψ be pure-present formulas, and let γ be a formula such that the only appearance of a U in α is $U(\varphi, \psi)$. Then, γ can be written as a syntactically separated formula where the only appearance of a U is $U(\varphi, \psi)$.*

Proof. We show this lemma by inducting on the pair (n_1, n_2) , where n_1 is the maximum number of nested S s above a $U(\varphi, \psi)$ and n_2 is the number of $U(\varphi, \psi)$ nested inside n_1 S s.

Base case. Here, $n_1 = 0$, and γ is already separated.

Induction step. Pick the most deeply nested subformula $S(\alpha, \beta)$ of γ such that all instances of $U(\varphi, \psi)$ in α and β are not nested inside a S . Applying lemma 3 to $S(\alpha, \beta)$ strictly reduces (n_1, n_2) , allowing us to use the induction hypothesis. Remember, lemma 3 only generates formulas where then only appearance of U is $U(\varphi, \psi)$, which is required to use the induction hypothesis.

This completes the proof. \blacktriangleleft

The next step generalizes this approach to different (basic) until subformulas.

► **Lemma 5.** *Let $\varphi_1, \varphi_2, \dots, \varphi_n$ and $\psi_1, \psi_2, \dots, \psi_n$ be pure present formulas and γ be a formula such that all appearances of U in γ are of the form $U(\varphi_i, \psi_i)$ for some $i \in \{1, 2, \dots, n\}$. Then, γ can be written as a syntactically separated formula.*

Proof. Predictably, we induct on n .

Base case. This is $n = 1$, identical to lemma 4.

Induction case. Introduce new propositional atoms p_1, p_2, \dots, p_{n-1} . For each $i \in \{1, \dots, n-1\}$, replace each occurrence of $U(\varphi_i, \psi_i)$ in γ with p_i to produce γ' . We can apply lemma 4 to γ' to produce its separated equivalent, γ'' . Replace each instance of p_i in γ'' with $U(\varphi_i, \psi_i)$ to produce γ''' . Finally, apply the induction hypothesis on γ''' to separate γ . This proves the lemma.

► **Remark.** It isn't difficult to see that we cannot use lemma 4 if we introduce a single atom p_n to represent $U(\varphi_n, \psi_n)$. Introducing more atoms is essential to the overall induction structure.

◀

We can now finally consider the case of nested U s.

► **Lemma 6.** *Let γ be a formula that doesn't contain S s nested inside a U . Then, γ can be separated.*

Proof. We cleverly induct on the maximum nesting depths of U s under a S . Let n be the maximum U -nesting depth of γ .

Base case. This is $n = 1$, which is lemma 5.

Induction step. Suppose there are m subformulas rooted at a U that aren't under a U and are under an S . Introduce $2m$ atoms $\{p_1, \dots, p_{2m}\}$ and replace the target and path conditions of these m subformulas with these atoms. This produces a new formula γ' that is amenable to lemma 5. Applying the lemma produces a separated formula γ'' that uses the atoms $\{p_1, \dots, p_{2m}\}$. These atoms may appear under a S in the separated formula γ'' . Now, replace each of these atoms by the target/path condition they substituted earlier. This produces γ''' , a formula with the maximum U -nesting depth under a S strictly $< n$. Applying the induction hypothesis on γ''' proves this lemma.

► **Remark.** We don't need to consider the value m in our induction hypothesis, as required in the proof of lemma 4.

◀

Before we finally prove the separation theorem, notice that, since U and S are duals of each other, the eliminations in section 3.1.1 and lemmas 3, 4, 5 and 6 hold when the U and S are swapped.

► **Theorem 7 (Separation Theorem).** *Any formula γ that uses S and U can be separated.*

Proof. We induct over the *junction depth* of the input formula. Define this depth as follows.

► **Definition 8 (Junction Depth).** *The junction depth of a temporal formula γ is the length of the longest sequence of subformulas $\alpha_1, \alpha_2, \dots, \alpha_n$ of γ such that*

1. *The root of all α_i is either a U or a S .*
2. *α_{i+1} is a subformula of α_i .*
3. *If α_i is rooted by a U (or a S), then α_{i+1} is rooted by a S (or a U , respectively).*
4. *There is no subformula β of γ such that*
 - a. *β is a strict subformula of α_i .*
 - b. *α_{i+1} is a strict subformula of β .*
 - c. *β and α_{i+1} are rooted by the same connective.*

Note that condition 4 isn't necessary to compute the junction depth. However, I will use it in my proof.

As an illustration, observe that the junction depth of the formula $U(a, S(U(c, d), U(e, f)))$ is 3, and there are two possible sequences:

- $U(a, S(U(c, d), U(e, f))), S(U(c, d), U(e, f)), U(c, d).$
- $U(a, S(U(c, d), U(e, f))), S(U(c, d), U(e, f)), U(e, f).$

Let the junction depth of γ be n .

Base case 1: $n = 1$. The formula is already separated.

Base case 2: $n = 2$. In this case, apply lemma 6 to separate γ .

Induction step: $n \geq 3$. Let there be m sequences of subformulas that witness the junction depth n . Form a set A of all subformulas at position 3 of these m sequences; the size of A can be less than m . Note that condition 4 makes these subformulas maximal; i.e., no formula in A is a subformula of another. This maximality allows us to substitute each formula in A with a newly introduced atom from the set $\{p_1, \dots, p_{|A|}\}$.

Call the resulting formula γ' . It isn't difficult to argue that this formula has a junction depth of strictly $< n$, allowing us to apply the induction hypothesis. This produces a separated formula γ'' with $|A|$ new atoms. Substitute the subformulas in A at the corresponding atoms in γ'' to produce γ''' .

Now, all subformulas in A have a junction depth of $n - 2$. If all of these appear in the pure-present segment of γ'' , the new junction depth of γ''' grows to at-most $n - 2$. Similarly, if one of these substitutions occurs inside a pure-past / pure-future segment of γ'' , the junction depth grows to at-most $n - 1$. This allows us to apply the induction hypothesis again, producing the fully separated formula γ''' .

This proves the separation theorem over linear time. ◀

References

- 1 Michael Benedikt and Clemens Ley. Limiting until in ordered tree query languages. *ACM Trans. Comput. Logic*, 17(2), mar 2016. doi:10.1145/2856104.
- 2 Volker Diekert and Paul Gastin. Pure future local temporal logics are expressively complete for mazurkiewicz traces. In Martín Farach-Colton, editor, *LATIN 2004: Theoretical Informatics*, pages 232–241, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- 3 Dov M. Gabbay, Ian Hodkinson, and Mark Reynolds. *Temporal Logic (Vol. 1): Mathematical Foundations and Computational Aspects*. Oxford University Press, Inc., USA, 1994.
- 4 Maarten Marx. Conditional xpath, the first order complete xpath dialect. In *Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04, page 13–22, New York, NY, USA, 2004. Association for Computing Machinery. doi:10.1145/1055558.1055562.