Guarded Negation Transitive Closure Logic

Diego Figueira
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, France

Santiago Figueira
University of Buenos Aires, Argentina
CONICET, Buenos Aires, Argentina

Yoshiki Nakamura
Chiba University, Japan

Abstract

We study the guarded negation fragment of transitive closure logic (*GNTC*). We show that the satisfiability problem for GNTC is 2ExpTime-complete, by establishing the following reductions: (i) a polynomial-time reduction from the satisfiability problem for GNTC to the satisfiability problem for the unary negation fragment *UNTC* of GNTC, and (ii) a direct exponential-time reduction from the satisfiability problem for UNTC to the non-emptiness problem for 2-way alternating parity tree automata. Furthermore, we show that the model checking problem for GNTC is $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-complete in combined complexity. Our result implies $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-completeness for both UNTC and $\mathrm{UNFO}^{\mathrm{reg}}$, which were left open in previous works.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases Transitive closure logic, Guarded negation, Unary negation, Satisfiability, Model checking

Digital Object Identifier 10.4230/LIPIcs.LICS.2026.43

Related Version *Full Version*: `https://arxiv.org/abs/2501.15303`

Funding This work was supported by JSPS KAKENHI Grant Numbers JP21K13828, JP25K14985; by ANR grants INTENDED (ANR-19-CHIA-0014) and EXPAND (ANR-25-CE23-1215); and by French-Argentinian IRP SINFIN.

Acknowledgements We thank anonymous reviewers for their helpful and insightful comments.

1 Introduction

A successful approach to obtaining decidable fragments of (fixpoint) first-order logics is to add syntactic restrictions on the use of quantification or negation. The "guardedness approach" leads to restricting its use via adding atoms acting as guards, protecting the syntax from any dangerous use of quantification or negation that may lead to undecidability. The restriction on quantification leads to "guarded FO" (or *GFO*) [3], where quantification can only be used in the form $\forall \bar{x}\ \alpha(\bar{x}\bar{y}\bar{z}) \to \varphi(\bar{x}\bar{y})$ or $\exists \bar{x}\ \alpha(\bar{x}\bar{y}\bar{z}) \wedge \varphi(\bar{x}\bar{y})$, where α is a (positive) atom. The restriction on negation results in the strictly more expressive fragment of "guarded negation FO" (or *GNFO*) [13], where the negation is restricted to be of the form $\alpha(\bar{x}\bar{y}) \wedge \neg\varphi(\bar{x})$ and no universal quantifiers are allowed. Both sorts of restrictions (GFO and GNFO) enjoy desirable properties, in particular they capture modal logics, and retain some of their features such as decidability of satisfiability and finite model property.

Adding *recursion* to these logics is a natural next step since many important inductively-defined concepts such as reachability cannot be expressed in FO. In this regard, the guardedness restrictions can also be extended to least fixpoint operators while preserving decidability. The extension of GNFO named *guarded negation fixpoint logic* (*GNFP*) [13], is then the fragment of (first-order) least fixpoint logic (*LFP*) obtained by also requiring that fixpoint

variables be guarded (in addition to the guarded negation restriction). GNFP is a highly expressive logic, in particular extending the modal μ-calculus with backward modalities [48]. GNFP still enjoys a decidable, 2ExpTime-complete, satisfiability problem [13, Theorem 4.4] – just like GNFO – while it no longer has the finite model property.

However, a limitation of GNFP is that it cannot express the transitive closure of formulas [6, Proposition 2].[1] The ability to use transitivity to "navigate" the structure is a desirable feature in many scenarios, and arguably the most basic form of recursion. But expressing the transitive closure $[\varphi]^*_{xy}(x, y)$ of a binary GNFO formula φ (testing that there is a path from x to y in the graph induced by the interpretation of φ) would require the use of *unguarded parameters* inside fixpoint operators, which is forbidden in GNFP.

In view of this, [6] studies guarded negation fixpoint logic with unguarded parameters (*GNFP-UP*), which extends GNFP while preserving decidability, and in particular captures transitive closure formulas. Yet, the expressive power that GNFP-UP adds to GNFP comes at a high computational price, as the complexity of the satisfiability problem increases from 2ExpTime to non-elementary [6, Theorem 20]. The non-elementary procedure has also matching lower bounds; however, such bounds make use of formulas that go well beyond the expressive power of transitive closure formulas.

This begs the question of whether the addition of transitive closure formulas to GNFO already shows an inherent non-elementary behaviour, or if it can be rather better behaved. To investigate this, we consider *GNTC*, the extension of GNFO with transitive-closure formulas,[2] which is incomparable with GNFP in terms of expressive power, but contained in GNFP-UP. Concretely, we address the following question: *Is the satisfiability of GNTC elementary?* We answer this question affirmatively.

Contributions. We show that the satisfiability problem for GNTC is 2ExpTime-complete, matching GNFO, whose lower bound holds even on fixed binary signatures.

We first show that the satisfiability problem for GNTC is equivalent – modulo polynomial-time reductions – to the restriction on its *unary-negation* fragment, or UNTC (Theorem 5). Concretely, *UNTC* is defined by restricting negations to be of the form $\neg\varphi(x)$, and transitive closure to be applied only to formulas with two free variables. This reduction preserves satisfiability and finite-satisfiability.

Once this reduction is in place, the 2ExpTime upper bound for GNTC follows from the (unpublished) result of [20, Theorem 8.5] establishing a 2ExpTime bound for the satisfiability of UNTC. The proof of [20] – heavily relying on prior work [29] – consists of a long chain of non-trivial reductions to the non-emptiness of 2-way alternating parity tree automata (2APTA) from which it is hard to derive any intuition. Here we propose an alternative proof for the satisfiability of UNTC based on a *direct reduction* to the non-emptiness of 2APTA, from which the tight complexity upper bound follows. That is, we show an exponential-time reduction from the satisfiability problem for UNTC to the non-emptiness problem for 2APTA, and since the latter is in ExpTime [48, §4], the 2ExpTime bound follows (Corollary 24). While our proof is not simple, it is arguably simpler than the previous sequence of reductions. It also has the advantage of being direct, shorter, and self-contained. Further, we show how our reduction can be intuitively interpreted via elementary derivation rules of a "local" (model/satisfiability) checker running on tree decompositions.

[1] The proof of this fact can be found in [7, Appendix A.1].

[2] In GNTC, transitive closure formulas do not contain *unguarded parameters*. For the extension of GNTC with unguarded parameters in transitive closure formulas (*GNTC-UP*), the satisfiability problem remains decidable (since GNTC-UP is contained in GNFP-UP) but it is non-elementary. This is because the containment problem for (existential) positive first-order logic with unary transitive closure (called PFO-TC1) is already non-elementary [9, Theorem 4] (see also [10]). For the same reason, UNTC with unguarded parameters in transitive closure formulas (*UNTC-UP*) is also decidable but non-elementary.

The proof we exhibit also gives an alternative direct proof of known results on the satisfiability for extensions of Propositional Dynamic Logics – known as ICPDL and UCPDL$^+$ [21, 29] –, since they admit trivial polynomial translations into UNTC [20, Proposition 8.3].

Finally, we also study the model checking problem, *i.e.*, the problem of, given a (finite) structure $\mathbb{A}$ and a GNTC sentence φ, whether $\mathbb{A} \models \varphi$. We show that, in combined complexity, this problem sits in the same complexity class as GNFO, namely $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-complete[3] (Theorem 26). This is in contrast with the complexity of GNFP, which is hard for P^{NP}, even when restricted to its unary negation fragment [47, Theorem 5.5]. Our results also establish $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-completeness of model checking for UNFO$^{\mathrm{reg}}$ and UNTC, which were left open in previous works [20, Open Problem 2 and Footnote 16 of version v1].

Related work. GNTC is a syntactic fragment of (first-order) *transitive closure logic* (*TC*). The satisfiability problem for full TC is highly undecidable [26, Corollary 6.7] (see also [28, 31]). Apart from the already mentioned work on the highly expressive GNFP-UP [6] which captures GNTC and implies its decidability (with a non-elementary bound), we are not aware of other decidable logics capturing GNTC.

Recent ISO standards for querying property graphs, SQL/PGQ [1] and GQL [2], are at least as expressive as FO and can express transitive closure [25]. In particular, a large fragment of GQL ("restrictor-free") is known to be in TC [22, Lemma 1]. Further, SQL/PGQ (with view creation) [45, Corollary 6.3], as well as the earlier language GraphLog [15, Theorem 3.3], precisely capture the expressive power of TC for graph database queries.

Another relevant line of work consists in considering extensions of Propositional Dynamic Logics, or *PDL* [23] (also known as $\mathcal{ALC}_{\mathrm{reg}}$). The logic *ICPDL* adds intersection and converse to PDL modalities ("programs") and [29, 34] show its satisfiability to be 2ExpTime-complete. The work [21] pushes this even further by adding also conjunctive queries over PDL programs. As shown in [20], a slight extension of the logic of [21], coined *UCPDL*$^+$, is expressive-equivalent to UNTC (while in principle exponentially less succinct).

UNTC (or, equivalently, UCPDL$^+$) is also known to capture several other logics such as *UNFO*^{reg} [32], which is the result of adding regular expressions over binary relations to UNFO, CQPDL [6, p. 4], the *positive calculus of relations with transitive closure* [42, 37, 40], as well as several graph database query languages, including Conjunctive Regular Path Queries (CRPQs) and Regular Queries [43] (see [20, §1.1 & Figure 1] for more examples).

An important relevant related result is that of [20, Theorem 8.5], showing a 2ExpTime upper bound for UNTC satisfiability. One major disadvantage of the proof of [20, Theorem 8.5] is that it is based on a long chain of reductions: an exponential translation from UNTC to UCPDL$^+$, which in turn has an exponential-time reduction to the satisfiability of ICPDL, which is shown in [29, Theorem 4.8] via an exponential-time reduction to two models of automata (in particular, a word automaton that runs on paths labeled by states of a tree automaton), which can then be reduced to the non-emptiness of 2-way alternating parity tree automata (2APTA). Moreover, it is necessary to carefully analyze the reductions to derive an optimal complexity bound.[4] Combining these non-trivial reductions UNTC $\rightsquigarrow$

[3] The class of decision problems solvable in polynomial time with $\mathcal{O}(\log^2(n))$ calls to an NP oracle.

[4] It requires showing: that the exponential translation UNTC $\rightsquigarrow$ UCPDL$^+$ [20, Proposition 8.1] produces formulas of polynomial so-called "conjunctive width"; that while the reduction UCPDL$^+$ $\rightsquigarrow$ ICPDL [20, Lemma 7.11] is exponential (which would lead to a 3ExpTime satisfiability procedure for UNTC) it nevertheless produces formulas of polynomial so-called "intersection-width"; and that while the reduction ICPDL $\rightsquigarrow$ 2APTA is exponential, it yields automata with a number of states which is only exponential in the "intersection-width" [29, Lemma 3.7]. Thus, the combined reduction of [20, Theorem 8.5] gives a doubly-exponential procedure by [29, Lemma 3.7].

UCPDL$^+$ $\rightsquigarrow$ ICPDL $\rightsquigarrow$ 2APTA we obtain a correct but rather long and intricate proof, from which it is difficult to gain intuition. The present work can be seen as a simplification and streamlining of this complex chain of reductions.

Finally, the recently introduced Guarded Fragment with Regular Guards (RGF) [5] extends the expressive power of GFO and ICPDL by allowing to use ICPDL-programs as guards. RGF is incomparable with GNTC (or UNTC) in terms of expressive power.

Organization. After some preliminary definitions in Section 2, we show the reduction from GNTC to UNTC in Section 3. The following Sections 4–7 are dedicated to showing decidability for the satisfiability of UNTC, starting with a high-level description of the proof in Section 4. Finally, we study the model checking problem in Section 8 and conclude with Section 9. All missing proofs can be found in the full version.

2 Preliminaries

We write $\mathbb{Z}$, $\mathbb{N}$, and $\mathbb{N}_+$ for the sets of integers, non-negative integers and positive integers, respectively. For $l, r \in \mathbb{Z}$, we write $[l..r]$ for the set $\{i \in \mathbb{Z} \mid l \leq i \leq r\}$. In particular, for $n \in \mathbb{N}$, we write $[n]$ for $[1..n]$. For a set X, we write $\#X$ for the *cardinality* of X and $\mathcal{P}(X)$ for the set of subsets of X. We use $\sqcup$ to denote that the set union $\cup$ is disjoint.

For a set X of *letters*, we write X^* for the set of *words* over X. We write ε for the *empty word* and ww' for the *concatenation* of words w and w'. We write $w \leq_{\text{pref}} w'$ if w is a prefix of w'. For a sequence $\bar{a} = a_1 \dots a_n$, the *length* $\|\bar{a}\|$ is n, and we write $\bar{a}[i]$ for a_i. For a set X of letters, an X-labeled (rooted) *tree* is a partial function $T \colon \mathbb{N}_+^* \rightharpoonup X$ such that its *domain* $\mathrm{dom}(T)$ is prefix-closed and non-empty. We say that T is *binary* if $\mathrm{dom}(T) \subseteq \{1,2\}^*$. For $w, w' \in \mathrm{dom}(T)$ and $d \in \mathbb{Z}$, we say that w' is in the *direction* d from w if $w = w'$ when $d = 0$, $wd \leq_{\text{pref}} w'$ when $d > 0$, and $w \not\leq_{\text{pref}} w'$ when $w = w''(\text{-}d)$ for some $w'' \in \mathbb{N}_+^*$. We write $\mathrm{dom}_{w,d}(T)$ for the set of all elements in the direction d from w. For $w \in \mathrm{dom}(T)$ and $d \in \mathbb{Z}$, we define $w \diamond d \in \mathrm{dom}(T)$ (*cf.*, [29, p. 285]) as the node adjacent to w and in the direction d from w if it exists, and as undefined otherwise; see Figure 1 for an illustration.[5]

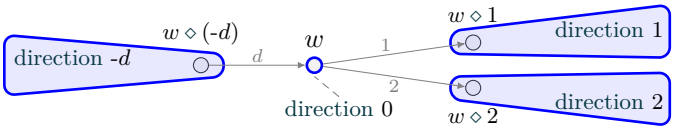


Figure 1 Illustration of directions (from w) and the operator $\diamond$.

Let $\mathcal{R}$ be an infinite set of *relation names*. We write $ar(R)$ for the *arity* of a relation name $R \in \mathcal{R}$. A *relational structure* $\mathbb{A}$ (henceforth *structure*) consists of a non-empty *countable*[6] *domain* $\mathrm{dom}(\mathbb{A})$ of *elements* and, for each relation name $R \in \mathcal{R}$, a relation $R^{\mathbb{A}} \subseteq \mathrm{dom}(\mathbb{A})^{ar(R)}$. We write STR_κ for the class of all structures of cardinality at most κ, and let $\mathsf{STR} \mathrel{\hat{=}} \mathsf{STR}_{\aleph_0}$ where $\aleph_0$ is the countably infinite cardinal and let $\mathsf{STR}_{\text{fin}} \mathrel{\hat{=}} \bigcup_{n \in \mathbb{N}_+} \mathsf{STR}_n$.

We use $\bar{\mathbb{A}}, \bar{\mathbb{B}}, \dots$ to denote $\mathsf{STR}_{\text{fin}}$-labeled binary trees. An element $g \in \mathrm{dom}(\bar{\mathbb{B}})$ is called a *bag*. Thus, $\bar{\mathbb{B}}(g)$ is the finite structure labeled at g, for any bag g of $\bar{\mathbb{B}}$. For a structure $\mathbb{A}$, a *tree decomposition* of $\mathbb{A}$ is an $\mathsf{STR}_{\text{fin}}$-labeled binary tree[7] $\bar{\mathbb{B}}$ such that

[5] For economy of space, trees depicted in this manuscript grow from left to right.
[6] Both GNTC and UNTC are semantic fragments of least fixpoint logic (LFP), which has the *downward Löwenheim–Skolem property* [24, 27]. Thus, countable structures suffice for satisfiability.
[7] $\mathsf{STR}_{\text{fin}}$-labeled *binary* trees suffice to enumerate countable structures of finite treewidth, *cf.*, *e.g.*, [6, §4].

- $\mathrm{dom}(\mathbb{A}) = \bigcup_{g \in \mathrm{dom}(\bar{\mathbb{B}})} \mathrm{dom}(\bar{\mathbb{B}}(g))$,
- $R^{\mathbb{A}} = \bigcup_{g \in \mathrm{dom}(\bar{\mathbb{B}})} R^{\bar{\mathbb{B}}(g)}$ for each $R \in \mathcal{R}$,
- $\mathrm{dom}(\bar{\mathbb{B}}(g)) \cap \mathrm{dom}(\bar{\mathbb{B}}(g'')) \subseteq \mathrm{dom}(\bar{\mathbb{B}}(g'))$ for all $g, g', g'' \in \mathrm{dom}(\bar{\mathbb{B}})$ such that g' is on the path between g and g''.

The *width* of $\bar{\mathbb{B}}$ is defined as $\sup_{g \in \mathrm{dom}(\bar{\mathbb{B}})}(\#\mathrm{dom}(\bar{\mathbb{B}}(g)) - 1)$. The *treewidth* [44, 16] of a structure $\mathbb{A}$ is the minimum width among tree decompositions of $\mathbb{A}$, or ∞ if there are no tree decompositions.

2.1 GNTC and UNTC

In this paper, we consider decidable syntactic fragments of (first-order) *transitive closure logic* (*TC*) [30]. Let $\mathcal{V}$ be an infinite set of *variables*, disjoint from $\mathcal{R}$. We write $\mathsf{FV}(\varphi)$ for the set of *free variables* occurring in a formula φ. For a sequence $\bar{x}$ of variables, we use $\varphi(\bar{x})$ to indicate that φ is a formula where $\mathsf{FV}(\varphi)$ is the set of variables occurring in $\bar{x}$. We shall sometimes abuse notation and write $\bar{x}$ also to denote the *set* of variables occurring in $\bar{x}$ – it will be clear from the context when we use it as a set. For a structure $\mathbb{A}$, formula φ and *interpretation* (*i.e.*, variable assignment) $I : \mathcal{V} \rightharpoonup \mathrm{dom}(\mathbb{A})$ *s.t.* $\mathsf{FV}(\varphi) \subseteq \mathrm{dom}(I)$, we write $\mathbb{A} \models_I \varphi$ to denote that $\mathbb{A}$ satisfies φ under I. We write $\mathbb{A} \models \varphi$ when φ is a *sentence* (*i.e.*, φ has no free variables). The *size* $\|\varphi\|$ of a formula φ is defined as the total number of symbols occurring in φ. For a sequence $\bar{y} = (y_1, \ldots, y_n) \in \mathcal{V}^*$ and a pairwise distinct sequence $\bar{x} = (x_1, \ldots, x_n) \in \mathcal{V}^*$ of the same length $n \geq 0$, we write $\varphi[\bar{y}/\bar{x}]$ for the formula φ in which each variable x_i has been replaced with y_i for each $i \in [n]$.

For a logic $\mathcal{L}$ (such as GNFO, GNTC, etc.), we denote by *SAT-*$\mathcal{L}$ its *satisfiability problem*, that is, the problem of, given a sentence $\varphi \in \mathcal{L}$, whether there exists some structure $\mathbb{A}$ such that $\mathbb{A} \models \varphi$, in which case we say it is *satisfiable* and that $\mathbb{A}$ is a *model of* φ.[8]

The guarded negation fragment of TC (*GNTC*) is defined as follows. A *guard* is an atom $R\bar{x}$ where $R \in \mathcal{R} \sqcup \{=\}$. Here, $R\bar{x}$ satisfies $\|\bar{x}\| = ar(R)$ where $ar(=) \mathrel{\hat{=}} 2$. We use α, β to denote guards.[9] The *GNTC formulas* are generated by the following grammar:

$$\varphi, \psi \ ::= \ \alpha \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \exists x\, \varphi \mid \underbrace{\alpha \wedge \neg\varphi}_{\text{(G-N)}} \mid \underbrace{[\alpha \wedge \beta \wedge \varphi]^*_{\bar{u}\bar{v}}\, \bar{x}\bar{y}}_{\text{(G-TC)}}.$$

Here, in the form of the *guarded negation formula* (G-N), we assume that φ is "guarded" by the atom α, *i.e.*, $\mathsf{FV}(\varphi) \subseteq \mathsf{FV}(\alpha)$. Also, for any formula of form (G-TC) – which we call *unparameterized guarded transitive closure formula*, *transitive closure formula* or simply *TC-formula* –, we assume the following:

- $\bar{x}$, $\bar{y}$, $\bar{u}$, and $\bar{v}$ have the same length (called, the *arity* of the TC-formula), and $\bar{u}\bar{v}$ is pairwise distinct,
- $\bar{u}$ and $\bar{v}$ are guarded by the atoms α and β, respectively: $\bar{u} \subseteq \mathsf{FV}(\alpha)$ and $\bar{v} \subseteq \mathsf{FV}(\beta)$,
- there are no TC-parameters: $\mathsf{FV}(\alpha \wedge \beta \wedge \varphi) \subseteq \bar{u}\bar{v}$.

A GNTC formula φ is a *GNFO formula* if φ does not contain TC-formulas (given by the grammar without the (G-TC) rule).

▶ Remark 1. GNTC is similar to "GNF(TC)" of [6], but modified in the following two points:

- our TC-formulas have both a left guard α and a right guard β; and
- we use classical guards from other logics (*e.g.*, GNFO and GFO), as opposed to the (exponential-sized) formula "gdd" from [6].

[8] $\mathbb{A}$ may be infinite since we shall deal with logics which do not enjoy the finite model property.
[9] We can extend guards to existentially quantified atoms obtaining a semantically equivalent logic via polynomial-time translations.

These modifications do not affect the expressive power. Further, the form of TC-formulas in GNTC is crucial for the polynomial-time reduction from GNTC to UNTC (*cf.* § 3): if $\bar{v}$ is not guarded (as in GNF(TC)), we have to unfold the TC once, and it may make an exponential blowup if the nesting of TC is unbounded. The second modification seems also necessary for the polynomial-time reduction from GNTC to UNTC in the model checking problem of § 8 (*cf.* [13, Remark 5.2]).

The *unary negation* fragment of GNTC (*UNTC*) is given by the following grammar:[10]

$$\varphi, \psi \ ::= \ \alpha \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \exists x\, \varphi \mid \neg\varphi \mid [\varphi]^*_{uv}\, xy.$$

Here, α is an atom $R\bar{x}$ where $R \in \mathcal{R} \sqcup \{=\}$, and in the form of the *negation formula* $\neg\varphi$, the number of free variables is at most one: $\#\mathsf{FV}(\varphi) \leq 1$.[11] In the form of TC-formula $[\varphi]^*_{uv}\, xy$, there are no TC-parameters: $\mathsf{FV}(\varphi) \subseteq \{u, v\}$. We say that a UNTC formula φ is a *UNFO formula* if φ does not contain TC-formulas. In terms of expressive power, GNTC strictly contains UNTC and is incomparable with GNFP. For example, the formula $\exists y(\theta(x, y) \wedge [\theta(u, v)]^*_{uv}(y, x))$ where $\theta(x, y) = R(x, y) \wedge \neg R(y, x)$ – saying that x belongs to a positive-length cycle made only of strictly forward R-edges – is not expressible either in GNFP or UNTC (see the full version for a proof).

In this paper, for satisfiability problems, we will use the following model theoretic property. A similar property holds even for GNFP-UP [6, Proposition 6].

▶ **Proposition 2** (*bounded treewidth model property* [20, Remark 7.2 and Theorem 8.5])**.** *Every satisfiable UNTC formula φ is satisfiable in a structure of treewidth at most $\|\varphi\| - 1$.*

2.2 2APTA

For satisfiability problems, we will give reductions to *2-way alternating parity tree automata* (2APTAs) [48]. Here, we use only binary trees as inputs (*cf.* Footnote 7) and we employ labeled backward transitions as in [29]. The *non-emptiness problem* for 2APTA is the following problem: given a 2APTA, is there a tree in the language of the 2APTA? We will rely on the following complexity result for 2APTAs.

▶ **Proposition 3** ([48])**.** *The non-emptiness problem for 2APTAs is in* ExpTime.

▶ Remark 4. In this paper, we use only *weak* [35] 2APTAs (a succinct form of Büchi automata [33]), as we consider only TC, and hence alternation-free fixpoint; see also, *e.g.*, [36, 14].

3 Reduction to UNTC

In this section we show that there is a polynomial-time reduction from SAT-GNTC to SAT-UNTC. However, the reduction does not imply a reduction from SAT-GNFO to SAT-UNFO since, as we shall see, it requires the expressive power of transitive closure even when restricted to GNFO sentences.

[10] Ignoring minor syntactic differences, UNTC can be equivalently defined as the syntactic fragment of GNTC where guards are restricted to formulas "$x = x$" ($x \in \mathcal{V}$).

[11] Allowing binary negation renders UNTC undecidable, as is already the case for UNFO with atomic negations or with inequality [47, §7.3]. Even the query equivalence problem for negation-free UNTC (*EPUNTC*) is undecidable with atomic negations (by [38, Theorem 50]) or with inequality (by [39, Theorem 4.9][41]).

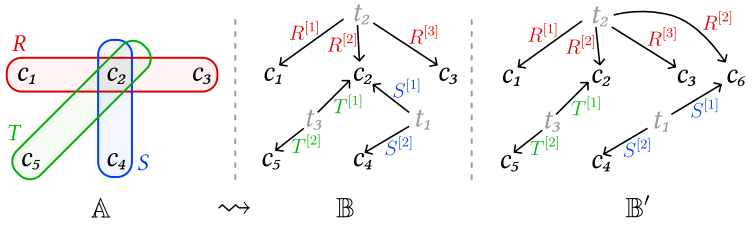


Figure 2 Idea of reduction of SAT-GNTC to SAT-UNTC. The structure $\mathbb{A}$ is depicted by a set of domain elements and some colored hyperedges representing tuples in the ternary relation R, the binary relation T, and the binary relation S. In the structures $\mathbb{B}$ and $\mathbb{B}'$, all relations are binary, and hence they are depicted as edge-labeled graphs. Observe that $\mathbb{B}$ and $\mathbb{B}'$ both encode $\mathbb{A}$, and that c_2 and c_6 in $\mathbb{B}'$ should be regarded as representing the same element since they are both the 2nd component in the R-tuple represented by t_2.

▶ **Theorem 5.** *There is a polynomial-time translation from GNTC sentences to UNTC sentences preserving both satisfiability and finite-satisfiability.*[12] *Further, the UNTC formulas in the reduction use only binary relations.*

Proof. The idea is to use fresh domain elements to encode tuples, and some binary relations of the form $R^{[i]}(t, c)$ to denote that the tuple t contains the element c in its i-th component. For example, in Figure 2, the tuple (c_1, c_2, c_3) of $\mathbb{A}$ is represented in $\mathbb{B}$ by the element t_2, together with the pairs (t_2, c_1), (t_2, c_2), and (t_2, c_3) in the interpretations of $R^{[1]}$, $R^{[2]}$, and $R^{[3]}$. Hence, some domain elements of $\mathbb{B}$ represent tuples of $\mathbb{A}$ (which we call "tuple elements") while others represent the elements contained in the components of tuples of $\mathbb{A}$ ("component elements"). However, we cannot enforce these relations to be functional, and thus there could be several c such that $R^{[i]}(t, c)$. In particular, we may have two different but "semantically equivalent" encodings. Indeed, the same tuple $R(c_1, c_2, c_3)$ of $\mathbb{A}$ may also be represented in a structure like $\mathbb{B}'$, where $R^{[2]}$ contains both (t_2, c_2) and (t_2, c_6), with c_6 coming from another relation S. This illustrates that the encoding need not be injective: different component elements may play the role of the same domain element of $\mathbb{A}$. This is not a problem in the presence of transitive closure, since we can encode the "equality", consisting of the transitive closure of the relation "the elements c and c' both appear in the i-th position of the same tuple". We can then do an encoding ensuring that we always work modulo this quotient of equality – for this we must take advantage of working in the presence of a transitive closure operator.

Now more concretely, let φ be a GNTC sentence, where we assume no repetition in the names of the bound variables. Let EQ be a fresh binary relation and let $\varphi[EQ/{=}]$ be the formula obtained from φ by replacing each appearance of $x = y$ with $EQ(x, y)$. Henceforth we shall use $\mathcal{R}$ to denote the (finite) set of relation names occurring in $\exists x\ EQ(x, x) \wedge \varphi[EQ/{=}]$ (in particular, $EQ \in \mathcal{R}$), and we shall use $\mathcal{R}_{\text{ar}}$ to denote the (finite) set of pairs (R, i) where $R \in \mathcal{R}$ and $i \in \{1, \ldots, ar(R)\}$ (in particular, $(EQ, 1), (EQ, 2) \in \mathcal{R}_{\text{ar}}$).

For each $(R, i) \in \mathcal{R}_{\text{ar}}$, we use a binary relation $R^{[i]}$. Each pair $(c, c') \in (R^{[i]})^{\mathbb{A}}$ denotes that the tuple represented by c contains, in its i-th component, the element c'. We divide the elements representing tuples from those representing the elements within those tuples:

[12] A sentence φ is *finitely-satisfiable* if there exists a *finite* structure that models the sentence. The *finite-satisfiability problem* of $\mathcal{L}$ is that given a sentence $\varphi \in \mathcal{L}$, whether φ is finitely-satisfiable.

$$\Phi_{sep} \mathrel{\hat{=}} \neg\exists x,y,z \bigvee_{(R,i),(S,j)\in\mathcal{R}_{\mathrm{ar}}} R^{[i]}(y,x) \wedge S^{[j]}(x,z) \qquad \mathit{tup}(t) \mathrel{\hat{=}} \exists x \bigvee_{(R,i)\in\mathcal{R}_{\mathrm{ar}}} R^{[i]}(t,x)$$

$$\Phi_{EQ} \mathrel{\hat{=}} \forall x\ (\mathit{comp}(x) \to \exists t \bigwedge_{i\in[2]} EQ^{[i]}(t,x)) \qquad \mathit{comp}(x) \mathrel{\hat{=}} \neg\mathit{tup}(x)$$

Note that Φ_{EQ} can be equivalently written in the syntax of UNFO. We shall call domain elements satisfying *tup* as *tuple elements* and the remaining ones *component elements* (denoted by *comp*). Note that Φ_{sep} ensures that component elements do not have outgoing relations, and hence that relations $R^{[i]}$ always link tuple elements to component elements.

We then let the following formula denote that two elements are supposed to represent the same element within a tuple.

$$(x \stackrel{*}{=} y) \mathrel{\hat{=}} \Big[\exists t \bigvee_{i,j\in[2]} (EQ^{[i]}(t,u) \wedge EQ^{[j]}(t,v)) \vee \bigvee_{(R,i)\in\mathcal{R}_{\mathrm{ar}}} (R^{[i]}(t,u) \wedge R^{[i]}(t,v))\Big]^*_{uv} xy$$

In the reduction we shall use partial functions $\mu : \mathcal{V} \rightharpoonup \mathcal{V} \times \mathcal{R} \times \mathbb{N}$, for which we adopt the notation $\mu(x) = (t, R[i])$ to denote $\mu(x) = (t, R, i)$. We will inductively define, for subformulas ψ of φ, UNTC formulas of the form "$\Phi_\mu(\psi)$", where $\mathrm{dom}(\mu) = \mathsf{FV}(\psi)$, having, as free variables, $\mathit{free}(\mu) \mathrel{\hat{=}} \{t \in \mathcal{V} : \exists x, i, R \text{ s.t. } \mu(x) = (t, R[i])\}$. The final formula is $\Phi_\emptyset(\varphi)$, where $\emptyset$ is the empty partial function.

Definition of $\Phi_\mu(\psi)$ *Conjunction and disjunction:* If $* \in \{\wedge, \vee\}$ then $\Phi_\mu(\psi' * \psi'') \mathrel{\hat{=}} \Phi_{\mu'}(\psi') * \Phi_{\mu''}(\psi'')$, where μ', μ'' are the restrictions of μ such that $\mathsf{FV}(\psi') = \mathrm{dom}(\mu')$ and $\mathsf{FV}(\psi'') = \mathrm{dom}(\mu'')$.

Atoms: The case of atoms (or guards) $R\bar{x}$ is defined by guessing a tuple t which contains $\bar{x}$ when seen as an R-tuple. However, each variable of $\bar{x}$ is not explicitly available but rather needs to be accessed via some $S^{[j]}(t', \cdot)$ which is stored in $\mu(\bar{x}[i]) = (t', S[j])$.

$$\Phi_\mu(R(\bar{x})) \mathrel{\hat{=}} \exists t\ \Phi^t_\mu(R(\bar{x})), \text{ where}$$
$$\Phi^t_\mu(R(x_1,\dots,x_n)) \mathrel{\hat{=}} \mathit{tup}(t) \quad \wedge \quad \exists \hat{x}_1,\dots,\hat{x}_n\ \exists \check{x}_1,\dots,\check{x}_n \bigwedge_{i\in[n]} R^{[i]}(t,\hat{x}_i) \quad \wedge \bigwedge_{\substack{i\in[n] \\ \mu(x_i)=(t_i,S[j_i])}} S^{[j_i]}(t_i,\check{x}_i) \quad \wedge \bigwedge_{i\in[n]} \hat{x}_i \stackrel{*}{=} \check{x}_i.$$

Quantification: The existential quantification consists of guessing a component element c. In our encoding, these elements are endpoints of $R^{[i]}$-relations. Hence, for doing this, we guess some tuple t, a relation R and an index i such that c is the i-th component of the R-tuple t.

$$\Phi_\mu(\exists x\ \varphi) \mathrel{\hat{=}} \exists t\ \mathit{tup}(t) \wedge \bigvee_{(R,i)\in\mathcal{R}_{\mathrm{ar}}} \Big((\exists c\ R^{[i]}(t,c)) \wedge \Phi_{\mu\sqcup\{x\mapsto(t,R[i])\}}(\varphi)\Big),$$

Negation: Guarded negation is transformed into unary negation by guessing a tuple t as before.

$$\Phi_\mu(R(x_1,\dots,x_n) \wedge \neg\varphi) \mathrel{\hat{=}} \exists t\ \Phi^t_\mu(R(x_1,\dots,x_n)) \wedge \neg\Phi_{\{x_i\mapsto(t,R[i]):x_i\in\mathsf{FV}(\varphi)\}}(\varphi)$$

Observe that $\Phi_{\{x_i\mapsto(t,R[i]):x_i\in\mathsf{FV}(\varphi)\}}(\varphi)$ is unary, since it has t as sole free variable, thus fulfilling the requirement of UNTC.

Transitive closure: Finally, the guarded transitive closure is converted into a unary transitive closure between two tuples t, t':

$$\Phi_\mu([\underbrace{R(\bar{u}\bar{v}') \wedge S(\bar{u}'\bar{v}) \wedge \psi(\bar{u}\bar{v})}_{\eta}]^*_{\bar{u}\bar{v}}\, \bar{x}\bar{y}) \mathrel{\hat{=}} \Phi_\mu(\bigwedge_i EQ(\bar{x}[i], \bar{y}[i])) \quad \vee \tag{1}$$

$$\Big(\underbrace{\exists t_R, t_S\ \Phi^{t_R}_\mu(R(\bar{x}\bar{y}')) \wedge \Phi^{t_S}_\mu(S(\bar{x}'\bar{y}))}_{\text{(a)}} \wedge \underbrace{[\Phi_{\{\bar{u}[i]\mapsto(t,R[i]),\bar{v}[i]\mapsto(t',S[i])\}_i}(\eta)]^*_{tt'} t_R t_S}_{\text{(b)}}\Big) \tag{2}$$

where the variables of $\bar{u}'$ and $\bar{v}'$ are contained in those of $\bar{u}$ and $\bar{v}$, respectively; and $\bar{x}'$ is such that $\bar{x}'[i] = \bar{x}[j]$ *iff* $\bar{u}'[i] = \bar{u}[j]$, and similarly for $\bar{y}'$. The formula can be read as follows: "either the transitive closure is satisfied in 0 steps (line 1), or we verify $R(\bar{x}\bar{y}')$ and $S(\bar{x}'\bar{y})$, whence there exist tuples t_S, t_R witnessing these facts (line 2-a), and this pair (t_R, t_S) of tuples must be in the transitive closure of $\Phi(\eta)$ under the function mapping $\bar{u}[i]$ to the $R[i]$-component of the first tuple t, and similarly for $\bar{v}[i]$ and $S[i]$ with t' (line 2-b)". Note that $\Phi_{\{\bar{u}[i]\mapsto(t,R[i]),\bar{v}[i]\mapsto(t',S[i])\}_i}(\eta)$ (line 2-b) has t, t' as sole free variables, meeting the requirements of UNTC.

▷ Claim 6. φ is (finitely-) satisfiable *iff* $\Phi_\emptyset(\varphi[EQ/=]) \wedge \Phi_{sep} \wedge \Phi_{EQ}$ is (finitely-) satisfiable.

Proof of Claim 6. ⇒ For the left-to-right direction, given a model of φ, consider adding a new domain element for each tuple and the relations $R^{[i]}$ as explained before. It follows that the resulting structure verifies $\Phi_\emptyset(\varphi[EQ/=]) \wedge \Phi_{sep} \wedge \Phi_{EQ}$.

⇐ For the right-to-left direction, suppose we have a model $\mathbb{B}$ of $\Phi_\emptyset(\varphi[EQ/=]) \wedge \Phi_{sep} \wedge \Phi_{EQ}$. Consider the structure $\mathbb{A}$ resulting from applying the following operations in the following order:

- **(i)** quotienting by $\stackrel{*}{=}$, that is, replacing every element c with its $\stackrel{*}{=}$-equivalence class $[c]_{\stackrel{*}{=}}$, defined as $[c]_{\stackrel{*}{=}} \mathrel{\hat{=}} \{c' \in \mathrm{dom}(\mathbb{B}) : \mathbb{B} \models (c \stackrel{*}{=} c')\}$; then
- **(ii)** for each arity n relation R of $\mathcal{R}$, adding a tuple $(c_1, \ldots, c_n)$ to R if for some c we have that (c, c_i) is in $R^{[i]}$ for every i; and finally
- **(iii)** removing the tuple elements, that is, removing, for each possible R, i, all elements c appearing as first component of a $R^{[i]}$-pair.

Observe that applying these operations to either $\mathbb{B}$ or $\mathbb{B}'$ of Figure 2 we obtain a structure isomorphic to the original structure $\mathbb{A}$.

It remains to show that $\mathbb{A} \models \varphi[EQ/=]$. The proof goes via structural induction on $\varphi[EQ/=]$, using the following inductive hypothesis: For every

- subformula ψ of $\varphi[EQ/=]$
- $\mu : \mathcal{V} \rightharpoonup \mathcal{V} \times \mathcal{R} \times \mathbb{N}$ such that $\mathrm{dom}(\mu) = \mathsf{FV}(\psi)$
- assignment $\xi : \mathit{free}(\mu) \to \mathrm{dom}(\mathbb{B})$ such that the image $Im(\xi)$ ranges over the tuple elements of $\mathbb{B}$,
- assignment $\xi' : \mathsf{FV}(\psi) \to \mathrm{dom}(\mathbb{A})$ mapping each variable $x \in \mathsf{FV}(\psi)$ to $[c]_{\stackrel{*}{=}}$, where c is a $R^{[i]}$-successor of $\xi(t)$ in $\mathbb{B}$ such that $\mu(x) = (t, R[i])$

we have

$$\mathbb{B} \models_\xi \Phi_\mu(\psi) \quad \textit{iff} \quad \mathbb{A} \models_{\xi'} \psi. \tag{3}$$

We proceed by induction on the structure of ψ.

$\psi = R(x_1, \ldots, x_n)$ Let $\mathbb{B}'$ be the model resulting after quotienting $\mathbb{B}$ by $\stackrel{*}{=}$ (step (i) of the construction). By the definition of $\Phi_\mu(\psi)$ we have that $\mathbb{B} \models_\xi \Phi_\mu(\psi)$ *iff* there are $b_0, \ldots, b_n, c_1, \ldots, c_n \in \mathrm{dom}(\mathbb{B})$ such that

1. b_i is a tuple element of $\mathbb{B}$ for each $i = 0, \ldots, n$,
2. $[c_i]_{\equiv_*}$ is a $R^{[i]}$-successor of $[b_0]_{\equiv_*}$ in $\mathbb{B}'$ for each $i \in [n]$, and
3. $[c_i]_{\equiv_*}$ is also a $S_i^{[j_i]}$-successor of $[b_i]_{\equiv_*}$ in $\mathbb{B}'$ for each $i \in [n]$,

where for each $i \in [n]$ we have $\mu(x_i) = (t_i, S_i[j_i])$, $\xi'(x_i) = [c_i]_{\equiv_*}$ and $\xi(t_i) = b_i$. We obtain $\mathbb{A}$ by applying steps (ii) and (iii) of the construction to $\mathbb{B}'$. One can see that $(\xi'(x_1), \ldots, \xi'(x_n)) \in R^{\mathbb{A}}$ *iff* $\mathbb{B} \models_\xi \Phi_\mu(\psi)$.

$\psi = \exists x\ \varphi$ By definition, we have that $\mathbb{B} \models_\xi \Phi_\mu(\exists x\ \varphi)$ *iff*

$$\mathbb{B} \models_\xi \exists t\ tup(t) \wedge \bigvee_{(R,i) \in \mathcal{R}_{\mathrm{ar}}} \Big((\exists c\ R^{[i]}(t,c)) \wedge \Phi_{\mu \sqcup \{x \mapsto (t, R[i])\}}(\varphi) \Big),$$

iff there exists a tuple element $b \in \mathrm{dom}(\mathbb{B})$, a $R^{[i]}$-successor c of b, and $(R,i) \in \mathcal{R}_{\mathrm{ar}}$ such that $\mathbb{B} \models_{\tilde{\xi}} \Phi_{\tilde{\mu}}(\varphi)$, where $\tilde{\xi} = \xi \sqcup \{t \mapsto b\}$ and $\tilde{\mu} = \mu \sqcup \{x \mapsto (t, R[i])\}$. By inductive hypothesis, this happens *iff*

$$\begin{aligned} &\text{there exists a tuple element } b \in \mathrm{dom}(\mathbb{B}), \text{ a } R^{[i]}\text{-successor } c \text{ of } b, \text{ and} \\ &(R,i) \in \mathcal{R}_{\mathrm{ar}} \text{ such that } \mathbb{A} \models_{\tilde{\xi}'} \varphi, \text{ where } \tilde{\xi}' = \xi' \sqcup \{x \mapsto [c]_{\equiv_*}\}, \text{ and } c \text{ is a} \\ &R^{[i]}\text{-successor of } \tilde{\xi}(t) (= b) \text{ in } \mathbb{B} \end{aligned} \tag{4}$$

For the left-to-right implication of (3), suppose that $\mathbb{B} \models_\xi \Phi_\mu(\exists x\ \varphi)$. Hence (4) is true and then $\mathbb{A} \models_{\tilde{\xi}'} \varphi$, where $\tilde{\xi}'$ extends ξ'. This implies that $\mathbb{A} \models_{\xi'} \exists x\ \varphi$.

For the right-to-left implication, suppose $\mathbb{A} \models_{\xi'} \exists x\ \varphi$. Then there exists $[c]_{\equiv_*} \in \mathrm{dom}(\mathbb{A})$ such that $\mathbb{A} \models_{\tilde{\xi}'} \varphi$, where $\tilde{\xi}' = \xi' \sqcup \{x \mapsto [c]_{\equiv_*}\}$ for some component element $c \in \mathrm{dom}(\mathbb{B})$. Since $\mathbb{B} \models \Phi_{EQ}$, there is a tuple element $b \in \mathrm{dom}(\mathbb{B})$ which has c as a $EQ^{[1]}$-successor. Then (4) holds for $(R,i) = (EQ, 1)$ and therefore $\mathbb{B} \models_\xi \Phi_\mu(\exists x\ \varphi)$.

$\psi = R(x_1, \ldots, x_n) \wedge \neg\varphi$

Without loss of generality, we can assume that $\mathsf{FV}(\varphi) = \{x_1, \ldots, x_n\}$. As in the case of the guards, we have that $\mathbb{B} \models_\xi \Phi_\mu(\psi)$ *iff* there are $b_0, \ldots, b_n, c_1, \ldots, c_n \in \mathrm{dom}(\mathbb{B})$ such that items 1), 2) and 3) hold, and also

$$\mathbb{B} \not\models_{\{t \mapsto b_0\}} \Phi_{\{x_i \mapsto (t, R[i])\}_{i \in [n]}}(\varphi). \tag{5}$$

By inductive hypothesis (5) is equivalent to

$$\mathbb{A} \not\models_{\xi'} \varphi, \tag{6}$$

where $\xi' = \{x_i \mapsto [c_i]_{\equiv_*} : i \in [n]\}$. By item 2) and the definition of $\mathbb{A}$ in (ii) we have that $([c_1]_{\equiv_*}, \ldots, [c_n]_{\equiv_*}) \in R^{\mathbb{A}}$. Hence (6) is equivalent to $\mathbb{A} \models_{\{x_i \mapsto [c_i]_{\equiv_*}\}_{i \in [n]}} \psi$, and this concludes the proof of this case.

$\psi = [R(\bar{u}\bar{v}') \wedge S(\bar{u}'\bar{v}) \wedge \varphi]^*_{\bar{u}\bar{v}}\bar{x}\bar{y}$ For simplicity, let us assume that $\bar{v}' = \bar{u}' = \emptyset$. The general case is similar. Hence we may assume that ψ is of this form:

$$\psi = [\underbrace{\alpha(\bar{u}) \wedge \beta(\bar{v}) \wedge \varphi}_{\eta}]^*_{\bar{u}\bar{v}}\bar{x}\bar{y}$$

where

$$\alpha(\bar{u}) = R(u_1, \ldots, u_n),$$
$$\beta(\bar{v}) = S(v_1, \ldots, v_n),$$

and $\|\bar{u}\| = \|\bar{v}\| = \|\bar{x}\| = \|\bar{y}\| = n$ (we notate u_i for $u[i]$ and the same for the other tuples of length n). As we reasoned in the case of the guards, let $\mathbb{B}'$ be the model resulting after quotienting $\mathbb{B}$ by $\stackrel{*}{=}$ (step (i) of the construction of $\mathbb{A}$).

By the definition of $\Phi_\mu(\psi)$ we have that $\mathbb{B} \models_\xi \Phi_\mu(\psi)$ *iff* one of the following is true:

1. $\mathbb{B} \models_\xi \Phi_\mu(EQ(x_i, y_i))$ for each $i \in [n]$;
2. there are $N \geq 2$ and elements $(b^{(j)})_{j\in[N]}$, $(c_i^{(j)})_{i\in[n],j\in[N-1]}$, $(d_i^{(j)})_{i\in[n],j\in[N-1]}$ of $\mathrm{dom}(\mathbb{B})$, such that
   a. $b^{(j)}$ is a tuple element of $\mathbb{B}$ for each $j \in [N]$,
   b. $b^{(1)}$ and $b^{(N)}$ satisfy $\mathbb{B} \models_{\xi\sqcup\{t_R\mapsto b^{(1)}\}} \Phi_\mu^{t_R}(R(\bar{x}\bar{y}'))$ and $\mathbb{B} \models_{\xi\sqcup\{t_S\mapsto b^{(N)}\}} \Phi_\mu^{t_S}(S(\bar{x}'\bar{y}))$,
   c. for all $j \in [N-1]$, $[c_i^{(j)}]_{\stackrel{*}{=}}$ is a $R^{[i]}$-successor of $[b^{(j)}]_{\stackrel{*}{=}}$ in $\mathbb{B}'$ for each $i \in [n]$, and $[d_i^{(j)}]_{\stackrel{*}{=}}$ is a $S^{[i]}$-successor of $[b^{(j+1)}]_{\stackrel{*}{=}}$ in $\mathbb{B}'$ for each $i \in [n]$,
   d. for all $j \in [N-1]$, we have $\mathbb{B} \models_{\xi_j} \Phi_{\tilde{\mu}}(\eta)$, where $\tilde{\mu} = \{u_i \mapsto (t, R[i]), v_i \mapsto (t', S[i])\}_{i\in[n]}$ and $\xi_j = \{t \mapsto b^{(j)}, t' \mapsto b^{(j+1)}\}$.

   The case 1 is straightforward, since it is just the case of the guards, so let us focus on case 2. By 2a and 2c, the side conditions of the inductive hypothesis are satisfied; applying it to 2d, we have that 2d holds *iff* $\mathbb{A} \models_{\xi'_j} \eta$ for all $j \in [N-1]$ where

$$\xi'_j = \{u_i \mapsto [c_i^{(j)}]_{\stackrel{*}{=}}, v_i \mapsto [d_i^{(j)}]_{\stackrel{*}{=}}\}_{i\in[n]}.$$

   Together with 2b and 2c, this is true *iff* $\mathbb{A} \models_{\xi'} \psi$, where $\xi' = \{x_i \mapsto [c_i^{(1)}]_{\stackrel{*}{=}}, y_i \mapsto [d_i^{(N-1)}]_{\stackrel{*}{=}}\}_{i\in[n]}$.

$\psi = \varphi_1 * \varphi_2$ for $* \in \{\vee, \wedge\}$ Straightforward. ◁

This concludes the reduction. ◀

4 Satisfiability of UNTC: Proof Approach

UNTC enjoys, as all decidable logics of this family, a "tree-like" model property. In other words, a sentence φ is satisfiable if, and only if, it is satisfiable in a structure of treewidth $\leq \|\varphi\|$. It is then sufficient to produce an algorithm which answers whether there exists such a bounded treewidth model of φ. We shall define a tree automaton $\mathcal{A}^\varphi$ that decides such a property, based on local properties of the bags, in such a way that $\mathcal{A}^\varphi$ has a non-empty language *iff* φ is satisfiable. Intuitively, such an automaton $\mathcal{A}^\varphi$ will guess a tree decomposition of a (possibly infinite) structure and verify that it is a model of φ.

As is usual in similar approaches, we see tree decompositions as binary trees labeled by structures ("bags") of size $\leq \|\varphi\|$ and, in order to make the alphabet finite, we abstract the information contained in bags in a standard way (using a fixed set of elements of size $\leq \|\varphi\|$). However, the crucial and challenging part of the construction is to define the states and transitions so that the automaton $\mathcal{A}^\varphi$ is only of single exponential size, as otherwise the satisfiability procedure would suffer from an exponential blowup.

The way we approach this issue is by defining some "local" rules on abstract semantics that need to be verified on the tree decomposition to ensure that the denoted structure verifies φ. Such rules are given via a form of "local checker". In the rest of the section we build some more precise intuition on the nature of the abstract semantics and the tableau-like rules of the local checker. This serves as a guide to the more technical terminology and definitions presented in § 5–7.

A tree decomposition $\bar{\mathbb{A}}$ is just a binary tree, labeled by finite structures, often referred to as *bags*. The structure $\odot\bar{\mathbb{A}}$ that such a decomposition denotes is the result of making the disjoint union of all the finite structures, and "gluing" any two equal elements from adjacent

structures, as will be shown in Figure 3. Such structure $\odot\bar{\mathbb{A}}$ is unique up to isomorphisms. We say that $\bar{\mathbb{A}}$ has width $k-1$ if the maximum cardinality among all the labeled structures is k (*e.g.*, the one in Figure 3 has width 2). It is plain that for every tree decomposition of width less than k there is another one denoting the same structure and using only some k fixed "named" elements[13] $\{\mathsf{c}_1, \ldots, \mathsf{c}_k\}$, and for this reason (combined with the bounded treewidth property of UNTC, *cf.* Proposition 2) we restrict to such tree decompositions.

Note that, standing at a bag g of a tree decomposition $\bar{\mathbb{A}}$, we can distinguish the following possible "*locations*" where an element of $\odot\bar{\mathbb{A}}$ can be found:

- "*the* c_i *element of this bag*", to refer to the element of $\odot\bar{\mathbb{A}}$ which is "locally" represented as c_i in g, *i.e.*, the element of $\odot\bar{\mathbb{A}}$ obtained as the result of gluing c_i of the structure in g with some other elements,
- "*in the direction* $d = 1$" (*resp.* $d = 2$) to refer to any (non-local) element of $\odot\bar{\mathbb{A}}$ that can be found in the left subtree (*resp.* right subtree) of g, and
- "*in the direction* $d =$ -1" (*resp.* $d =$ -2) to denote an element which is elsewhere in the tree (*i.e.*, in a node outside the subtree rooted at g), assuming g is a left-child in $\bar{\mathbb{A}}$ (*resp.* assuming it is a right-child).

Hence, each bag g has a set (or "domain") of available locations $\mathcal{D}_g^{\bar{\mathbb{A}}} \subseteq \{\mathsf{c}_1, \ldots, \mathsf{c}_k\} \sqcup \{1, 2, \text{-}1, \text{-}2\}$, which depends on the domain of the labeled structure, and whether g is a left/right child and has left/right children. We can then *abstract* an element w from $\odot\bar{\mathbb{A}}$ by its location $\mathsf{Abs}_g^{\bar{\mathbb{A}}}(w) \in \mathcal{D}_g^{\bar{\mathbb{A}}}$ – *i.e.*, either by a "local name" c_i or by a direction $d \in \{1, 2, \text{-}1, \text{-}2\}$ where it can be found from g. Conversely, the *concretization* of an element c_i in the context of $(\bar{\mathbb{A}}, g)$ is the corresponding element of $\odot\bar{\mathbb{A}}$, and the concretization of a direction d is the set of all elements of $\odot\bar{\mathbb{A}}$ that can be found in the direction d from g; denoted by $\mathsf{C}_g^{\bar{\mathbb{A}}}(\mathsf{c}_i)$ and $\mathsf{C}_g^{\bar{\mathbb{A}}}(d)$ respectively. In this way each bag g induces the *finite* partition $\{\mathsf{C}_g^{\bar{\mathbb{A}}}(z)\}_{z \in \mathcal{D}_g^{\bar{\mathbb{A}}}}$ on the (possibly infinite) domain of $\odot\bar{\mathbb{A}}$.

In light of this, for reasoning about the satisfiability of UNTC, instead of working with classical interpretations (*i.e.*, mappings from variables to elements of $\odot\bar{\mathbb{A}}$) we shall work with *abstract interpretations*, mapping each variable x to a set[14] $\mathscr{I}(x) \subseteq \mathcal{D}_g^{\bar{\mathbb{A}}}$ of locations relative to a bag g. A *concretization* of such an abstract interpretation $\mathscr{I}$ is then just a (classical) interpretation with the same domain as $\mathscr{I}$, but mapping each variable x to an element of $\mathsf{C}_g^{\bar{\mathbb{A}}}(z)$ for some $z \in \mathscr{I}(x)$ – *i.e.*, to an element of the concretization of some $\mathscr{I}(x)$ member.

By means of such abstract interpretations we define a *local checker* (LC) over tree decompositions. Concretely, an *instance* of the LC (called "state" in later sections) consists of a tree decomposition $\bar{\mathbb{A}}$, a node g thereof, a set Γ of UNTC formulas, and an abstract interpretation $\mathscr{I}$ of the free variables of these formulas. We denote such instances by $(\Gamma)^{\bar{\mathbb{A}}}_{\mathscr{I},g}$. The LC has the task of checking that there exists some concretization of the abstract interpretation $\mathscr{I}$ (in the context of $(\bar{\mathbb{A}}, g)$), satisfying all formulas of Γ on $\odot\bar{\mathbb{A}}$. In particular, for a sentence φ, the LC on the instance $(\varphi)^{\bar{\mathbb{A}}}_{\emptyset,\varepsilon}$ checks if $\odot\bar{\mathbb{A}}$ is a model of φ.[15]

The LC will solve this problem by making calls to a Boolean combination of other LC instances – in some sense mimicking the Boolean operations of the formula it checks. For this reason it is useful to have two (dual) "modes" of instances: the "positive mode" (denoted

[13] This is a standard construction, by inserting an additional bag between each pair of adjacent bags, *cf.*, *e.g.*, [40, p. 27:23].

[14] For convenience, we map to sets, not elements. This definition simplifies the rule (move), because the concretization of a direction d corresponds to a set in an adjacent bag (*cf.* moving set $\mathcal{M}^{\bar{\mathbb{A}}}_{g,d}$).

[15] Although the fact of working with abstractions may compromise completeness due to the loss of information outside the bag at the given location, the local checker remains complete under the evaluation strategy we will exhibit.

with a superscript 1) defined just as before, and the "negative mode" (with superscript 0) testing the opposite – namely, that no concretization verifies all formulas of Γ. We will hence use positive Boolean formulas on these signed model checking instances (we shall use $\wedge$ and $\vee$ as Boolean connectors to avoid confusion with the UNFO syntax). For example, we will denote by $(\Gamma)^{1,\bar{\mathbb{A}}}_{\mathscr{I},g} \vee (\Delta)^{0,\bar{\mathbb{A}}}_{\mathscr{I},g'}$ the truth value $p \vee \neg q$ where p is the truth value of the model checking problem on $(\Gamma)^{1,\bar{\mathbb{A}}}_{\mathscr{I},g}$ and q is the one on $(\Delta)^{1,\bar{\mathbb{A}}}_{\mathscr{I},g'}$.

With this intuition, the LC for UNFO is defined via the intuitive derivation rules of Figure 6 – somewhat akin to sequent calculus rules or tableau rules – showing how instances are derived from other instances.[16] A rule "$A \rightsquigarrow B$" should be read as "*for checking A, it suffices to check B*" if A is in positive mode (*i.e.*, $p = 1$) or as "*for checking A, it is necessary to check B*" if A is in negative mode ($p = 0$).[17] Notably, to apply the rule ($\neg$), all the free variables of φ should be concretized; this condition is required for the soundness of the LC.[18]

The rules for the transitive closure (TC), given in Figure 7, are slightly more involved and necessitate further definitions. However, the reader can already see that they correspond to the different ways of shrinking a TC unravelling: either by consuming an element from one side if its endpoint is local to the current bag – (TC-l) and (TC-r) –, or by splitting it into two TCs when an endpoint is not local – (TC-sp-l) and (TC-sp-r).

Using these rules, a *derivation tree* (called "run" in later sections) for an instance $(\Gamma)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$ is an instance-labeled (possibly infinite) tree such that each positive-instance (*resp.* negative-instance) is "consistent" with some (*resp.* every) applicable rule.[19] For example, if a node in the tree is labeled A and has two children with labels B and C, and if the applicable rules for A are $A \rightsquigarrow B \wedge C$ and $A \rightsquigarrow D \vee A$, then it will verify the derivation tree condition at the node if A is positive (it is consistent with the first rule), but not if A is negative (it is not consistent with the second rule).

While these rules can be easily seen to be sound, they still do not yield an algorithm, since: (i) the input tree decomposition is an infinite object and (ii) the derivation trees may be infinite. However, the LC rules can be straightforwardly interpreted as the transition function of a (2APTA) tree automaton, where derivation trees are seen as "runs" and each LC instance $(\Gamma)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$ is abstracted as a "state" $\Gamma_{\mathscr{I}}$ with priority p.[20] One can then define *accepting* derivation trees in the same way as accepting runs for 2APTA, *i.e.*, requiring that every infinite branch has infinitely many negative instances. With these semantics, the local (model) checker is indeed both sound and complete (corollaries of Theorems 13 and 20): an instance is a yes-instance if, and only if, there exists an accepting derivation tree that contains it at the root.

However, the derivation trees starting with an instance $(\{\varphi\})^{0,\bar{\mathbb{A}}}_{\mathscr{I},\varepsilon}$ may in principle yield an automaton $\mathcal{A}^{\varphi}$ with an infinite number of states of the form $\Gamma_{\mathscr{I}}$. To solve this, we show that starting with the sentence φ, the number of states needed in any accepting run is not

[16] In these rules, we use "Γ, Δ" and "Γ, φ" to denote $\Gamma \cup \Delta$ and $\Gamma \cup \{\varphi\}$ respectively, and the shorthand $\wedge^0 \mathrel{\hat{=}} \vee$, $\wedge^1 \mathrel{\hat{=}} \wedge$, $\vee^0 \mathrel{\hat{=}} \wedge$, $\vee^1 \mathrel{\hat{=}} \vee$.

[17] For understanding rule (move), $g \diamond d$ denotes the bag adjacent to g in the direction d, and $\mathscr{I}_d$ is the expected way of updating the abstract interpretation $\mathscr{I}$ with the information that we have moved towards d.

[18] While rule ($\neg$) is complete for unary negation cases, it would be incomplete for GNFO, *e.g.*, when two variables indicate the vertices not belonging to the same bag at the same time.

[19] More precisely, a rule $A \rightsquigarrow \dot{\varphi}$ is "consistent" with a node if its left-hand side A is the node's label and by replacing all the children labels with "true" makes the positive Boolean formula $\dot{\varphi}$ true.

[20] As everywhere else in this section, we disregard minor technicalities; for instance, the 2APTA would actually require some auxiliary states.

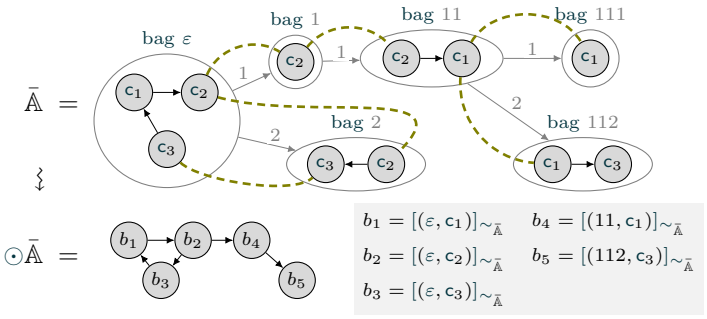


Figure 3 Illustrative example of the gluing operator $\odot$.

too many, and can be drawn from a set $\mathrm{cl}(\varphi)$ of states which is of (single) exponential size (Theorems 13 and 20 and Propositions 15 and 22). Crucially, this result relies on devising a *derivation strategy* for the LC rules, using only instances with formulas from $\mathrm{cl}(\varphi)$ which always finds an accepting derivation tree whenever there is one – the intuition behind the strategy will be given in § 6.2. In this way, we obtain an exponential-time reduction from the satisfiability of UNTC to the non-emptiness problem for 2APTA, which in turn is in ExpTime (Proposition 3), thus obtaining a 2ExpTime procedure (Corollary 23).

Organization of SAT-UNTC proof. In § 5 we fix the notations for our LC. We then give the LC for UNFO in § 6 and extend it to UNTC in § 7.

5 Abstract Semantics on Tree Decompositions

In this section we introduce an "abstract" semantics on tree decompositions, an alternative to the standard semantics. For simplicity, we consider only structures of binary relations (*i.e.* $ar(R) = 2$ for each $R \in \mathcal{R}$), relying on the polynomial-time reduction of Theorem 5, and we consider only the following structures whose "names" are drawn from the *fixed names* $\{\mathsf{c}_1, \ldots, \mathsf{c}_k\}$ for tree decompositions of width at most $k-1$:

$$\mathsf{NSTR}_k \;\hat{=}\; \{\mathbb{A} \in \mathsf{STR} : \mathrm{dom}(\mathbb{A}) \subseteq \{\mathsf{c}_1, \ldots, \mathsf{c}_k\}\}.$$

Hence, we consider NSTR_k-labeled binary trees (henceforth just *trees*) to express tree decompositions of width less than k.

For a tree $\bar{\mathbb{A}}$, the *glued structure* $\odot\bar{\mathbb{A}}$ is defined as the structure obtained from the disjoint union of all structures labeled in $\bar{\mathbb{A}}$ by gluing elements having the same name coming from adjacent bags. Each element of $\odot\bar{\mathbb{A}}$ is expressed as a quotient class $[(g, c)]_{\sim_{\bar{\mathbb{A}}}}$ *w.r.t.* the "gluing" equivalence relation $\sim_{\bar{\mathbb{A}}}$, where (g, c) is a pair of a bag $g \in \mathrm{dom}(\bar{\mathbb{A}})$ and its element $c \in \mathrm{dom}(\bar{\mathbb{A}}(g))$. By definition, for every tree decomposition $\bar{\mathbb{B}}$ of any structure $\mathbb{A}$, the glued structure $\odot\bar{\mathbb{B}}$ is isomorphic to $\mathbb{A}$.

For instance, consider the tree $\bar{\mathbb{A}}$ with a single binary relation name, in Figure 3. Structures are depicted as directed graphs, where node labels indicate the corresponding (named) domain elements. Then, the structure $\odot\bar{\mathbb{A}}$ has the shape shown in Figure 3, by taking the quotient *w.r.t.* the equivalence relation $\sim_{\bar{\mathbb{A}}}$ expressed by dashed lines.

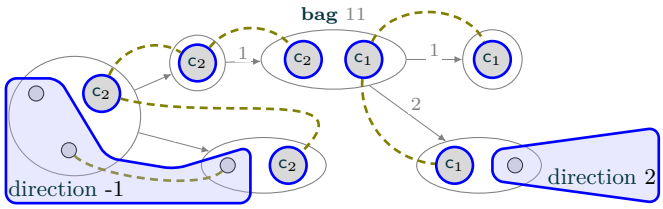


Figure 4 Illustration of $\mathsf{Abs}^{\bar{\mathbb{A}}}_{11}$, with $\bar{\mathbb{A}}$ taken from Figure 3.

5.1 The Abstract Semantics Setting

Inspired by the abstract interpretation from program semantics [17], we consider abstracting nodes based on directions (recall § 2). The *abstract domain* $\mathcal{D}$ is defined as the following set:

$$\mathcal{D} \;\hat{=}\; \{\mathsf{c}_i \mid i \in \mathbb{N}_+\} \sqcup \{\text{-}2, \text{-}1, 1, 2\}.$$

We use e to denote an element in $\mathcal{D}$, c to denote a *fixed name* in $\{\mathsf{c}_i \mid i \in \mathbb{N}_+\}$, and d to denote a *direction* in $\{\text{-}2, \text{-}1, 1, 2\}$, respectively.

Let $\bar{\mathbb{A}}$ be a tree. For each bag $g \in \mathrm{dom}(\bar{\mathbb{A}})$, the *abstraction function* $\mathsf{Abs}^{\bar{\mathbb{A}}}_g \colon \mathrm{dom}(\odot\bar{\mathbb{A}}) \to \mathcal{D}$ is defined as follows:

$$\mathsf{Abs}^{\bar{\mathbb{A}}}_g(w) \;\hat{=}\; \begin{cases} c & \text{if } w = [(g, c)]_{\sim_{\bar{\mathbb{A}}}}, \\ d & \text{else if } w = [(g', c')]_{\sim_{\bar{\mathbb{A}}}} \text{ for some } g' \text{ in the} \\ & \quad \text{direction } d \text{ on } g \text{ and some } c' \in \mathrm{dom}(\bar{\mathbb{A}}(g')). \end{cases}$$

For instance, when $\bar{\mathbb{A}}$ is the tree in Figure 3, the elements $[(\varepsilon, \mathsf{c}_1)]_{\sim_{\bar{\mathbb{A}}}}$, $[(\varepsilon, \mathsf{c}_2)]_{\sim_{\bar{\mathbb{A}}}}$, $[(\varepsilon, \mathsf{c}_3)]_{\sim_{\bar{\mathbb{A}}}}$, $[(11, \mathsf{c}_1)]_{\sim_{\bar{\mathbb{A}}}}$, $[(112, \mathsf{c}_3)]_{\sim_{\bar{\mathbb{A}}}}$ are mapped to $\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, 1, 1$ by $\mathsf{Abs}^{\bar{\mathbb{A}}}_{\varepsilon}$, mapped to $\text{-}1, \mathsf{c}_2, \text{-}1, 1, 1$ by $\mathsf{Abs}^{\bar{\mathbb{A}}}_{1}$, and mapped to $\text{-}1, \mathsf{c}_2, \text{-}1, \mathsf{c}_1, 2$ by $\mathsf{Abs}^{\bar{\mathbb{A}}}_{11}$. Figure 4 is an illustration of $\mathsf{Abs}^{\bar{\mathbb{A}}}_{11}$.

For each bag $g \in \mathrm{dom}(\bar{\mathbb{A}})$, the *concretization function* $\mathsf{C}^{\bar{\mathbb{A}}}_g \colon \mathcal{D} \to \mathcal{P}(\mathrm{dom}(\odot\bar{\mathbb{A}}))$ is defined as the inverse image function of $\mathsf{Abs}^{\bar{\mathbb{A}}}_g$. By definition, for each element c, the set $\mathsf{C}^{\bar{\mathbb{A}}}_g(c)$ is the singleton $\{[(g, c)]_{\sim_{\bar{\mathbb{A}}}}\}$ if c is in the direction 0 (*i.e.*, in the bag g), and the empty set $\emptyset$ otherwise; for each direction d, the set $\mathsf{C}^{\bar{\mathbb{A}}}_g(d)$ consists of elements in the direction d but not in the direction 0. Note that $\mathsf{C}^{\bar{\mathbb{A}}}_g(d)$ is possibly empty even if the bag $g \diamond d$ exists in $\bar{\mathbb{A}}$. Additionally, we lift the domain of $\mathsf{C}^{\bar{\mathbb{A}}}_g$ from $\mathcal{D}$ to its powerset $\mathcal{P}(\mathcal{D})$ by $\mathsf{C}^{\bar{\mathbb{A}}}_g(X) \;\hat{=}\; \bigcup_{e \in X} \mathsf{C}^{\bar{\mathbb{A}}}_g(e)$.

The abstract domain $\mathcal{D}^{\bar{\mathbb{A}}}_g$ of $\bar{\mathbb{A}}$ on g is defined as $\mathcal{D}^{\bar{\mathbb{A}}}_g \;\hat{=}\; \{e \in \mathcal{D} \mid \mathsf{C}^{\bar{\mathbb{A}}}_g(e) \neq \emptyset\}$. The family $\{\mathsf{C}^{\bar{\mathbb{A}}}_g(e)\}_{e \in \mathcal{D}^{\bar{\mathbb{A}}}_g}$ is a partition of $\mathrm{dom}(\odot\bar{\mathbb{A}})$.

A (powerset-lifted) *abstract interpretation* on a bag g is a partial function $\mathscr{I} \colon \mathcal{V} \rightharpoonup \mathcal{P}(\mathcal{D}^{\bar{\mathbb{A}}}_g)$. For an interpretation $I \colon \mathcal{V} \rightharpoonup \mathrm{dom}(\odot\bar{\mathbb{A}})$ and an abstract interpretation $\mathscr{I} \colon \mathcal{V} \rightharpoonup \mathcal{P}(\mathcal{D}^{\bar{\mathbb{A}}}_g)$ on a bag g with $\mathrm{dom}(\mathscr{I}) = \mathrm{dom}(I)$, we say that $\mathscr{I}$ is an *abstraction* of I (or, I is a *concretization* of $\mathscr{I}$) on g if $I(x) \in \mathsf{C}^{\bar{\mathbb{A}}}_g(\mathscr{I}(x))$ for all $x \in \mathrm{dom}(I)$. Additionally, we will use functions $\mathscr{I}$ by lifting the domain from $\mathcal{V}$ to its powerset $\mathcal{P}(\mathcal{V})$ by $\mathscr{I}(X) \;\hat{=}\; \bigcup_{x \in X} \mathscr{I}(x)$.

▶ **Definition 7.** *For a class $\mathbb{F}$ of finite sets of formulas, the state set $\mathcal{Q}_{\mathbb{F}}$ is defined as the set of $\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$, where*

- $\Gamma \in \mathbb{F}$,
- $p \in \{0, 1\}$ *is a priority*,
- $\bar{\mathbb{A}}$ *is a tree*,
- $g \in \mathrm{dom}(\bar{\mathbb{A}})$ *is a bag, and*
- $\mathscr{I}$ *is an abstract interpretation on g s.t.* $\mathsf{FV}(\Gamma) \subseteq \mathrm{dom}(\mathscr{I})$.

We use $\dot{\Gamma}, \dot{\Delta}, \dot{\Lambda}, \ldots$ to denote states. The *priority* $\Omega(\dot{\Gamma})$ of a state $\dot{\Gamma} = \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$ is defined by $\Omega(\dot{\Gamma}) \mathrel{\hat{=}} p$. We write $\mathcal{Q}_{\mathrm{UNFO}}$ (*resp.* $\mathcal{Q}_{\mathrm{UNTC}}$) for the set $\mathcal{Q}_{\mathbb{F}}$ where $\mathbb{F}$ is the class of all finite sets of UNFO (*resp.* UNTC) formulas. For $k \in \mathbb{N}_+$, let

$$\mathcal{Q}^{(k)}_{\mathbb{F}} \mathrel{\hat{=}} \{\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \in \mathcal{Q}_{\mathbb{F}} \mid \bar{\mathbb{A}} \text{ is an } \mathsf{NSTR}_k\text{-labeled tree}\}.$$

We now define the abstract semantics on tree decompositions.

▶ **Definition 8.** *For $\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \in \mathcal{Q}_{\mathbb{F}}$, we define $\models \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$ as follows:*

$$\models \Gamma^{1,\bar{\mathbb{A}}}_{\mathscr{I},g} \iff \odot\bar{\mathbb{A}} \models_I \bigwedge \Gamma \text{ for a concretization } I \text{ of } \mathscr{I} \text{ on } g, \quad \models \Gamma^{0,\bar{\mathbb{A}}}_{\mathscr{I},g} \iff \not\models \Gamma^{1,\bar{\mathbb{A}}}_{\mathscr{I},g}.$$

In particular, when Γ is the singleton of a sentence φ and $p = 1$, it coincides with the standard semantics by definition:

▶ **Proposition 9.** *For every sentence φ, we have: $\odot\bar{\mathbb{A}} \models \varphi$ iff $\models \{\varphi\}^{1,\bar{\mathbb{A}}}_{\emptyset,g}$.*
(Hence, this equivalence holds particularly when $g = \varepsilon$.)

▶ **Example 10.** Recall the tree $\bar{\mathbb{A}}$ in Figure 3. We use E for the unique binary relation name. Let $\varphi \mathrel{\hat{=}} \neg\exists y Exy$. Then, $\odot\bar{\mathbb{A}} \models_I \varphi$ *iff* $I(x)$ has no successor *iff* $I(x) = b_5$. In the abstract semantics, for instance, we have $\models \{\varphi\}^{1,\bar{\mathbb{A}}}_{x\mapsto\{2\},11}$ by $b_5 \in \mathsf{C}^{\bar{\mathbb{A}}}_{11}(2)$ and we have $\not\models \{\varphi\}^{1,\bar{\mathbb{A}}}_{x\mapsto\{\mathsf{c}_1,\mathsf{c}_2,\text{-}1\},11}$ by $b_5 \notin \mathsf{C}^{\bar{\mathbb{A}}}_{11}(\{\mathsf{c}_1,\mathsf{c}_2,\text{-}1\})$.

In the sequel, we use this abstract semantics as an alternative to the standard semantics.

Additionally, in Definition 8, we canonically lift $\mathcal{Q}_{\mathbb{F}}$ to the set $\mathscr{B}_+(\mathcal{Q}_{\mathbb{F}})$ of (positive) Boolean formulas, where, for example, $\models \dot{\varphi} \wedge \dot{\psi}$ *iff* $\models \dot{\varphi}$ and $\models \dot{\psi}$. Here, the set $\mathscr{B}_+(X)$ of *positive Boolean formulas* over a set X of *Boolean variables* is generated by the grammar:

$$\dot{\varphi}, \dot{\psi} ::= p \mid \mathsf{false} \mid \mathsf{true} \mid \dot{\varphi} \vee \dot{\psi} \mid \dot{\varphi} \wedge \dot{\psi}, \qquad (p \in X)$$

where the dot notation is used to distinguish from formulas in UNTC / GNTC. We denote by $\equiv_{\mathsf{L}}$ (*resp.* $\leqq_{\mathsf{L}}$) the semantical equivalence relation (*resp.* entailment relation). For $op \in \{\mathsf{false}, \mathsf{true}, \vee, \wedge\}$, let op^p be op itself if $p = 1$ and be the "*dual*" of op if $p = 0$.

5.2 Moving on Bags in the Abstract Semantics

For each direction $d \in [\text{-}2..2]$, the *neighbouring set* $\mathcal{N}^{\bar{\mathbb{A}}}_{g,d} \subseteq \mathcal{D}^{\bar{\mathbb{A}}}_g$ is defined so that its concretization is the set of all elements in bags in the direction d on g: $\mathsf{C}^{\bar{\mathbb{A}}}_g(\mathcal{N}^{\bar{\mathbb{A}}}_{g,d}) = \bigcup_{g' \in \mathrm{dom}_{g,d}(\bar{\mathbb{A}})} \mathsf{C}^{\bar{\mathbb{A}}}_{g'}(\mathrm{dom}(\bar{\mathbb{A}}(g')))$. More concretely, $\mathcal{N}^{\bar{\mathbb{A}}}_{g,d} \mathrel{\hat{=}} (\mathrm{dom}(\bar{\mathbb{A}}(g \diamond d)) \cup \{d\}) \cap \mathcal{D}^{\bar{\mathbb{A}}}_g$ if $g \diamond d$ is defined and $\mathcal{N}^{\bar{\mathbb{A}}}_{g,d} \mathrel{\hat{=}} \emptyset$ otherwise. Note $\mathcal{N}^{\bar{\mathbb{A}}}_{g,0} = \mathrm{dom}(\bar{\mathbb{A}}(g))$. For instance, when $\bar{\mathbb{A}}$ is the tree in Figure 3 and $g = 11$, we have $\mathcal{N}^{\bar{\mathbb{A}}}_{g,0} = \{\mathsf{c}_1, \mathsf{c}_2\}$, $\mathcal{N}^{\bar{\mathbb{A}}}_{g,1} = \{\mathsf{c}_1\}$, $\mathcal{N}^{\bar{\mathbb{A}}}_{g,2} = \{\mathsf{c}_1, 2\}$, $\mathcal{N}^{\bar{\mathbb{A}}}_{g,\text{-}1} = \{\mathsf{c}_2, \text{-}1\}$, and $\mathcal{N}^{\bar{\mathbb{A}}}_{g,\text{-}2} = \emptyset$.

When $\mathscr{I}(\mathsf{FV}(\Gamma)) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d}$ and $d \neq 0$, we can move from g to the adjacent bag $g \diamond d$. Before defining the abstract interpretation obtained from moving towards d (which we shall define below as $\mathscr{I}_d$), we need to define the abstract domain on which it lives.

For each direction $d \in \{\text{-}2, \text{-}1, 1, 2\}$, the *moving set* $\mathcal{M}^{\bar{\mathbb{A}}}_{g,d} \subseteq \mathcal{D}^{\bar{\mathbb{A}}}_{g\diamond d}$ is defined so that its concretization on $g \diamond d$ coincides with that of d on g: $\mathsf{C}^{\bar{\mathbb{A}}}_{g\diamond d}(\mathcal{M}^{\bar{\mathbb{A}}}_{g,d}) = \mathsf{C}^{\bar{\mathbb{A}}}_g(d)$. More concretely, $\mathcal{M}^{\bar{\mathbb{A}}}_{g,d} \mathrel{\hat{=}} \mathcal{D}^{\bar{\mathbb{A}}}_{g\diamond d} \setminus (\mathrm{dom}(\bar{\mathbb{A}}(g)) \cup \{\text{-}d\})$. For instance, for the tree $\bar{\mathbb{A}}$ in Figures 3 and 4, $\mathcal{M}^{\bar{\mathbb{A}}}_{1,1} = \{\mathsf{c}_1, 2\}$, $\mathcal{M}^{\bar{\mathbb{A}}}_{11,2} = \{\mathsf{c}_3\}$, and $\mathcal{M}^{\bar{\mathbb{A}}}_{112,\text{-}2} = \{\mathsf{c}_2, \text{-}1\}$. Figure 5 depicts $\mathcal{M}^{\bar{\mathbb{A}}}_{1,1}$.

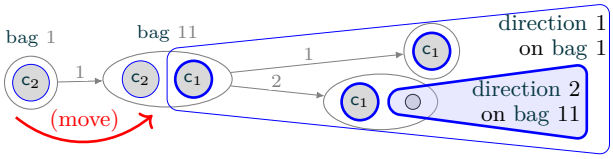


Figure 5 Illustration of $\mathcal{M}^{\bar{\mathbb{A}}}_{1,1}$, with $\bar{\mathbb{A}}$ taken from Figure 3. The direction 1 on bag 1 has the same concretization as $\{\mathsf{c}_1, 2\}$ on bag 11.

When $\mathscr{I}(\mathsf{FV}(\Gamma)) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d}$ and $d \in \{\text{-}2, \text{-}1, 1, 2\}$, the abstract interpretation $\mathscr{I}_d \colon \mathcal{V} \rightharpoonup \mathcal{P}(\mathcal{D}^{\bar{\mathbb{A}}}_{g\diamond d})$ on $g \diamond d$ is defined as follows: $\mathscr{I}_d(x) \mathrel{\hat{=}} \begin{cases} (\mathscr{I}(x) \setminus \{d\}) \cup \mathcal{M}^{\bar{\mathbb{A}}}_{g,d} & \text{if } d \in \mathscr{I}(x), \\ \mathscr{I}(x) & \text{otherwise.} \end{cases}$ By construction, $\mathsf{C}^{\bar{\mathbb{A}}}_{g\diamond d}(\mathscr{I}_d(x)) = \mathsf{C}^{\bar{\mathbb{A}}}_{g}(\mathscr{I}(x))$. Thus, we have: $\models \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$ *iff* $\models \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I}_d,g\diamond d}$.

Such an $\mathscr{I}_d$ can be taken thanks to $\mathscr{I}(\mathsf{FV}(\Gamma)) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d}$; in general, there is no $\mathscr{I}'$ such that $\mathsf{C}^{\bar{\mathbb{A}}}_{g\diamond d}(\mathscr{I}'(x)) = \mathsf{C}^{\bar{\mathbb{A}}}_{g}(\mathscr{I}(x))$. Nevertheless, by an appropriate evaluation (§ 6.2), we can always reach a state $\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$ satisfying $\mathscr{I}(\mathsf{FV}(\Gamma)) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d}$ for some $d \in \{\text{-}2, \text{-}1, 1, 2\}$.

6 Local Checker for SAT-UNFO

The satisfiability problem for UNFO is known to be in 2ExpTime [47, Theorem 4.5]. In [47], this is shown via an exponential reduction to the satisfiability problem for two-way modal μ-calculus. In this section, we present a different 2ExpTime algorithm by a direct 2APTA construction based on a local checker on tree decompositions. We will extend this algorithm to UNTC in the next section.

6.1 A Local Checker

We next define the *local checker* (*LC*) for UNFO, based on 2APTAs. For convenience in writing examples, given Γ in "$\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$", we may put $\mathscr{I}(x)$ in the superscript of some occurrences x, may just write d for $\{d\}$ in the superscript, and may use the set notations as in the sequent calculus. For instance, $(Rx^{\{\text{-}1,1\}}y, \exists x Sxy^2)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$ denotes the state $\{Rxy, \exists x Sxy\}^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$, with $\mathscr{I}(x) = \{\text{-}1, 1\}$ and $\mathscr{I}(y) = \{2\}$. For two sequences $\bar{x} = x_1 \dots x_k$ and $\bar{c} = c_1 \dots c_k$ where the elements of $\bar{x}$ are pairwise distinct, we write $\mathscr{I}[\bar{c}/\bar{x}]$ for the $\mathscr{I}$ in which each $\mathscr{I}(x_i)$ has been substituted with c_i for each $i \in [k]$.

The binary relation $(\rightsquigarrow) \subseteq \mathcal{Q}_{\mathrm{UNFO}} \times \mathscr{B}_+(\mathcal{Q}_{\mathrm{UNFO}})$ is defined as the smallest relation closed under the *rules* in Figure 6. We lift this relation to $(\rightsquigarrow) \subseteq \mathscr{B}_+(\mathcal{Q}_{\mathrm{UNFO}}) \times \mathscr{B}_+(\mathcal{Q}_{\mathrm{UNFO}})$, as the smallest relation that contains the original $(\rightsquigarrow)$ and is single-step compatible with operators in positive Boolean formula (*e.g.*, if $\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \rightsquigarrow \dot{\psi}$, then $\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \vee \dot{\rho} \rightsquigarrow \dot{\psi} \vee \dot{\rho}$).

A *run* starting from $\dot{\Gamma}$ is a $\mathcal{Q}_{\mathrm{UNFO}}$-labeled tree τ where $\tau(\varepsilon) = \dot{\Gamma}$ and such that for every $w \in \mathrm{dom}(\tau)$ we have that for $\left\{ \begin{array}{ll} \text{some } \tau(w) \rightsquigarrow \dot{\psi} & \text{if } \Omega(\tau(w)) = 1 \\ \text{every } \tau(w) \rightsquigarrow \dot{\psi} & \text{if } \Omega(\tau(w)) = 0 \end{array} \right\}$ the positive Boolean formula resulting from replacing in $\dot{\psi}$ each child state of w (*i.e.*, each element $\tau(wa)$) with true is semantically equivalent to true.

A run τ is *accepting* if for every infinite path $a_1 a_2 \dots$ in τ, there are infinitely many n with $\Omega(\tau(a_1 a_2 \dots a_n)) = 0$ (in which case we say that the path has priority 0). We write $\vdash \dot{\Gamma}$ if there is an accepting run starting from $\dot{\Gamma}$. The definition of runs is almost the same as that of runs for alternating automata. For example, consider $\tau(w)$ with odd priority

$$
\begin{aligned}
()^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} &\rightsquigarrow \mathsf{true}^p && \text{(emp)}\\
(\Gamma, \varphi \wedge \psi)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} &\rightsquigarrow (\Gamma, \varphi, \psi)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} && (\wedge)\\
(\Gamma, \varphi \vee \psi)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} &\rightsquigarrow (\Gamma, \varphi)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \vee^p (\Gamma, \psi)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} && (\vee)\\
(\neg\varphi)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} &\rightsquigarrow (\varphi)^{1-p,\bar{\mathbb{A}}}_{\mathscr{I},g} \quad \text{if} \begin{cases} \mathscr{I}(x) \subseteq \mathrm{dom}(\bar{\mathbb{A}}(g)) \text{ and } \#\mathscr{I}(x) = 1 \\ \text{for each } x \in \mathsf{FV}(\varphi) \end{cases} && (\neg)\\
\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} &\rightsquigarrow \bigvee^p_{e \in \mathscr{I}(x)} \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I}[\{e\}/x],g} && \text{(conc)}\\
(\alpha)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} &\rightsquigarrow \mathsf{true}^p \quad \text{if} \begin{cases} \text{for each } x \in \mathsf{FV}(\alpha), \text{ there is a } c \in \mathrm{dom}(\bar{\mathbb{A}}(g)) \text{ } s.t. \\ \mathscr{I}(x) = \{c\}, \text{ and } \bar{\mathbb{A}}(g) \models_{\{x \mapsto c \mid x \mapsto \{c\} \in \mathscr{I}\}} \alpha \end{cases} && (\alpha)\\
(\Gamma, \exists x \varphi)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} &\rightsquigarrow (\Gamma, \varphi[z/x])^{p,\bar{\mathbb{A}}}_{\mathscr{I}[\mathcal{D}^{\bar{\mathbb{A}}}_g/z],g} && (\exists)\\
(\Gamma, \Delta)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} &\rightsquigarrow \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \wedge^p \Delta^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \quad \text{if} \begin{cases} \mathscr{I}(x) \subseteq \mathrm{dom}(\bar{\mathbb{A}}(g)) \text{ and } \#\mathscr{I}(x) = 1 \\ \text{for each } x \in \mathsf{FV}(\Gamma) \cap \mathsf{FV}(\Delta) \end{cases} && \text{(split)}\\
\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} &\rightsquigarrow \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I}_d,g \diamond d} \quad \text{if } \mathscr{I}(\mathsf{FV}(\Gamma)) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d} && \text{(move)}
\end{aligned}
$$

Figure 6 *Rules* of the LC for UNFO. Here, z is used as a fresh variable.

$\Omega(\tau(w)) = 1$ and let $\dot{\Gamma}, \dot{\Delta}, \dot{\Lambda} \in \mathcal{Q}_{\mathrm{UNFO}}$. If there is a rule $\tau(w) \rightsquigarrow \dot{\Gamma} \wedge \dot{\Delta}$, the condition is passed at the position w when both $\dot{\Gamma}$ and $\dot{\Delta}$ occur in the children of w. Similarly, if there is a rule $\tau(w) \rightsquigarrow \dot{\Gamma} \vee \dot{\Delta}$, the condition is passed at w when either $\dot{\Gamma}$ or $\dot{\Delta}$ occurs. If multiple rules exist for $\tau(w)$, *e.g.*, $\tau(w) \rightsquigarrow \dot{\Gamma} \wedge \dot{\Delta}$ and $\tau(w) \rightsquigarrow \dot{\Lambda}$, then they can be combined into the single rule $\tau(w) \rightsquigarrow (\dot{\Gamma} \wedge \dot{\Delta}) \vee \dot{\Lambda}$. When $\tau(w) = \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$ has priority $p = 0$, the definition is given so that the merged formula for $\Gamma^{0,\bar{\mathbb{A}}}_{\mathscr{I},g}$ is the "dual" of that for $\Gamma^{1,\bar{\mathbb{A}}}_{\mathscr{I},g}$. Each rule $\dot{\Gamma} \rightsquigarrow \dot{\psi}$ in Figure 6 is defined so that $\models \dot{\Gamma}$ *iff* $\models \dot{\psi}$. The LC is sound and complete *w.r.t.* the semantics on tree decompositions (Theorem 13). Below, we give a toy example.

▶ **Example 11.** Let $\bar{\mathbb{A}}$ be the tree in which $\bar{\mathbb{A}}(1)$, $\bar{\mathbb{A}}(\varepsilon)$, and $\bar{\mathbb{A}}(2)$ are given as follows, and $\bar{\mathbb{A}}(g)$ is undefined otherwise: $\Big(\mathsf{c}_1 \text{-} R \text{-} \mathsf{c}_2\Big)$, $\Big(\mathsf{c}_2\Big)$, $\Big(\mathsf{c}_2 \text{-} S \text{-} \mathsf{c}_3\Big)$. Then, $\odot\bar{\mathbb{A}} = (\circ\text{-}R\text{-}\circ\text{-}S\text{-}\circ)$. Let φ be $\exists x \exists y \exists z (Rxz \wedge Szy)$. Then, $\models (\varphi)^{1,\bar{\mathbb{A}}}_{\emptyset,\varepsilon}$ holds. In our LC, we can also construct an accepting run, as follows, where we abbreviate abstract interpretations $\emptyset[\dots]$ to $-$:

$$
\begin{aligned}
&\exists x \exists y \exists z (Rxz \wedge Szy)^{1,\bar{\mathbb{A}}}_{\emptyset,\varepsilon} \rightsquigarrow^*_{(\exists)(\mathrm{conc})} \bigvee_{e_1 \in \{\mathsf{c}_2,1,2\}} (\exists y \exists z (Rx^{e_1} z \wedge Szy))^{1,\bar{\mathbb{A}}}_{-,\varepsilon}\\
&\rightsquigarrow^*_{(\exists)(\mathrm{conc})(\wedge)} \bigvee_{e_1,e_2,e_3 \in \{\mathsf{c}_2,1,2\}} (Rx^{e_1} z^{e_3}, Sz^{e_3} y^{e_2})^{1,\bar{\mathbb{A}}}_{-,\varepsilon} \geqq_{\mathsf{L}} (Rx^1 z^{\mathsf{c}_2}, Sz^{\mathsf{c}_2} y^2)^{1,\bar{\mathbb{A}}}_{-,\varepsilon}\\
&\rightsquigarrow_{(\mathrm{split})} (Rx^1 z^{\mathsf{c}_2})^{1,\bar{\mathbb{A}}}_{-,\varepsilon} \wedge (Sz^{\mathsf{c}_2} y^2)^{1,\bar{\mathbb{A}}}_{-,\varepsilon} \rightsquigarrow^*_{(\mathrm{move})} (Rx^{\mathsf{c}_1} z^{\mathsf{c}_2})^{1,\bar{\mathbb{A}}}_{-,1} \wedge (Sz^{\mathsf{c}_2} y^{\mathsf{c}_3})^{1,\bar{\mathbb{A}}}_{-,2} \rightsquigarrow^*_{(\alpha)} \geqq_{\mathsf{L}} \mathsf{true}.
\end{aligned}
$$

Thus, there is a (finite) accepting run, and hence $\vdash (\varphi)^{1,\bar{\mathbb{A}}}_{\emptyset,\varepsilon}$.

6.2 Evaluation Strategy

We now present a strategy to evaluate $\models \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$ in our LC.

Step 1 We eliminate the outermost $\wedge$ using the rule $(\wedge)$, and the outermost $\vee$ and $\exists$ by nondeterministically selecting one disjunct via the rules $(\vee)(\exists)(\mathrm{conc})$, as much as possible. We can then assume:

a) $\#\mathscr{I}(x) = 1$ for each $x \in \mathsf{FV}(\Gamma)$, and
b) Γ is of the form $(\psi_1, \dots, \psi_n)$ where each ψ_i is either an atom α or a negation formula $\neg\rho$, from which it follows that
c) for each ψ_i, for some $d \in$ [-2..2], $\mathscr{I}(\mathsf{FV}(\psi_i)) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d}$.

Here when ψ_i is of the form α, this is derived from $\models \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$. When ψ_i is of the form $\neg\rho$, it immediately follows from the condition of unary negation: $\#\mathsf{FV}(\rho) \leq 1$.

Step 2 If ψ_i satisfies the condition c) for $d = 0$, we can eliminate ψ_i by applying (split). We can then assume that

c') for each ψ_i, for a $d \in \{-2, -1, 1, 2\}$, $d \in \mathscr{I}(\mathsf{FV}(\psi_i)) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d}$.

Here for the eliminated state $(\psi_i)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$, we apply (α)[21] if ψ_i is α and apply $(\neg)$ if ψ_i is $\neg\rho$, and then we return to Step 1.

Step 3 From the condition c'), by applying the rule (split), we can assume that

c") for some $d \in \{-2, -1, 1, 2\}$, $d \in \mathscr{I}(\mathsf{FV}(\Gamma)) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d}$.

Step 4 By the condition c"), we apply the rule (move) based on the argument of § 5.2; then, we move from g to the adjacent bag $g \diamond d$. Finally, we go back to Step 1.

6.3 Closure Property

Based on the evaluation strategy above, we can obtain a stronger completeness with a closure property, as we shall see in Theorem 13.

▶ **Definition 12.** *For a UNFO formula φ, the* closure $\mathrm{cl}(\varphi)$ *is the set of UNFO formula sets defined as follows:*

$$\begin{aligned}
\mathrm{cl}(\alpha) &\triangleq \{(\alpha), ()\}, & \mathrm{cl}(\varphi \vee \psi) &\triangleq \{(\varphi \vee \psi)\} \cup \mathrm{cl}(\varphi) \cup \mathrm{cl}(\psi),\\
\mathrm{cl}(\varphi \wedge \psi) &\triangleq \{(\varphi \wedge \psi)\} \cup \{(\Gamma, \Delta) \mid \Gamma \in \mathrm{cl}(\varphi), \Delta \in \mathrm{cl}(\psi), \mathsf{FV}(\Gamma) \cap \mathsf{FV}(\Delta) \subseteq \mathsf{FV}(\varphi) \cap \mathsf{FV}(\psi)\},\\
\mathrm{cl}(\exists x\varphi) &\triangleq \{(\exists x\varphi)\} \cup \bigcup_{z \text{ is fresh}} \mathrm{cl}(\varphi[z/x]), & \mathrm{cl}(\neg\varphi) &\triangleq \{(\neg\varphi)\} \cup \mathrm{cl}(\varphi).
\end{aligned}$$

By easy induction on φ, we have the following monotonicity: $(\Gamma, \Delta) \in \mathrm{cl}(\varphi)$ implies $\Gamma \in \mathrm{cl}(\varphi)$.

For a class $\mathbb{F}$ of finite formula sets, $\mathbb{F}$-*runs* are defined in the same way as runs, where for each rule $\dot{\Gamma} \rightsquigarrow \dot{\psi}$ in the LC (Figure 6), each state $\dot{\Delta}$ occurring in $\dot{\psi}$ has been replaced with $\begin{cases} \mathsf{false} & \text{if } p = 1 \\ \mathsf{true} & \text{if } p = 0 \end{cases}$ if $\dot{\Delta} \notin \mathcal{Q}_{\mathbb{F}}$. For a state $\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \in \mathcal{Q}_{\mathbb{F}}$, we write $\vdash_{\mathbb{F}} \dot{\Gamma}$ if there is an accepting $\mathbb{F}$-run starting from $\dot{\Gamma}$. We then have the following soundness and completeness theorem with a closure property, based on the strategy of § 6.2.

▶ **Theorem 13.** *Let φ be a UNFO formula. For every $\dot{\Gamma} \in \mathcal{Q}_{\mathrm{cl}(\varphi)}$, we have: $\models \dot{\Gamma}$ iff $\vdash_{\mathrm{cl}(\varphi)} \dot{\Gamma}$.*

On the size of the closure, we have the following.

▶ **Proposition 14.** *For all UNFO formulas φ, the cardinality of $\mathrm{cl}(\varphi)$, up to renaming free variables, is at most $(2\|\varphi\|)^{2\|\varphi\|}$.*

Proof. By easy induction on φ. ◀

Hence, we can give an exponential bound on the number of states, up to an appropriate equivalence, as follows.

[21] Here, we sometimes need also (move); *e.g.*, in Figure 3, $(Ex^{c_2}y^{c_3})^{1,\bar{\mathbb{A}}}_{-,\varepsilon} \rightsquigarrow_{(\mathrm{move})} (Ex^{c_2}y^{c_3})^{1,\bar{\mathbb{A}}}_{-,2} \rightsquigarrow_{(\alpha)} \mathsf{true}$.

$$\dots \rightsquigarrow \dots \qquad \text{(all the rules for UNFO given in Figure 6)}$$

$$(\Gamma, [\psi]^*_{uv} xy)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \rightsquigarrow (\Gamma, x = y)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \vee^p (\Gamma, \psi[zz'/uv], [\psi]^*_{uv} z'y)^{p,\bar{\mathbb{A}}}_{\mathscr{I}[\{c\}/z][\mathcal{D}^{\bar{\mathbb{A}}}_g/z'],g}$$
$$\text{if } \mathscr{I}(x) = \{c\} \subseteq \mathrm{dom}(\bar{\mathbb{A}}(g)), \qquad \text{(TC-l)}$$

$$(\Gamma, [\psi]^*_{uv} xy)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \rightsquigarrow (\Gamma, x = y)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \vee^p (\Gamma, [\psi]^*_{uv} xz, \psi[zz'/uv])^{p,\bar{\mathbb{A}}}_{\mathscr{I}[\mathcal{D}^{\bar{\mathbb{A}}}_g/z][\{c\}/z'],g}$$
$$\text{if } \mathscr{I}(y) = \{c\} \subseteq \mathrm{dom}(\bar{\mathbb{A}}(g)), \qquad \text{(TC-r)}$$

$$(\Gamma, [\psi]^*_{uv} xy)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \rightsquigarrow \bigvee^p_{d' \in [\text{-}2..2] \setminus \{d\}} (\Gamma, [\psi]^*_{uv} xz, \psi[zz'/uv], [\psi]^*_{uv} z'y)^{p,\bar{\mathbb{A}}}_{\mathscr{I}[\mathcal{N}^{\bar{\mathbb{A}}}_{g,d}/z][\mathcal{N}^{\bar{\mathbb{A}}}_{g,d'}/z'],g}$$
$$\text{if } \mathscr{I}(x) = \{d\} \subseteq \{\text{-}2, \text{-}1, 1, 2\},\ \mathscr{I}(y) = \{e\} \not\subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d}, \qquad \text{(TC-sp-l)}$$

$$(\Gamma, [\psi]^*_{uv} xy)^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \rightsquigarrow \bigvee^p_{d' \in [\text{-}2..2] \setminus \{d\}} (\Gamma, [\psi]^*_{uv} xz, \psi[zz'/uv], [\psi]^*_{uv} z'y)^{p,\bar{\mathbb{A}}}_{\mathscr{I}[\mathcal{N}^{\bar{\mathbb{A}}}_{g,d'}/z][\mathcal{N}^{\bar{\mathbb{A}}}_{g,d}/z'],g}$$
$$\text{if } \mathscr{I}(y) = \{d\} \subseteq \{\text{-}2, \text{-}1, 1, 2\},\ \mathscr{I}(x) = \{e\} \not\subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d}. \qquad \text{(TC-sp-r)}$$

Figure 7 *Rules* of the LC for UNTC. Here, z and z' are used as fresh variables.

▶ **Proposition 15.** *For UNFO formulas φ, the number of $\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \in \mathcal{Q}^{(k)}_{\mathrm{cl}(\varphi)}$ is $2^{\mathcal{O}(\|\varphi\|(\log\|\varphi\|+k))}$, up to forgetting $\mathscr{I}(x)$ for $x \notin \mathsf{FV}(\Gamma)$, $\bar{\mathbb{A}}$, and g and up to renaming free variables.*

Proof. By $(2\|\varphi\|)^{2\|\varphi\|}$ (the number of $\Gamma \in \mathrm{cl}(\varphi)$; Proposition 14) $\times$ 2 (for p) $\times$ $\mathcal{O}((2^{(k+4)})^{\|\varphi\|})$ (the number of $\mathscr{I}$; note $\#\mathsf{FV}(\Gamma) \leq \|\varphi\|$ and $\#\mathcal{D}^{\bar{\mathbb{A}}} \leq k+4$) $\leq 2^{\mathcal{O}(\|\varphi\|(\log\|\varphi\|+k))}$. ◀

6.4 Reduction to 2APTAs

Using the LC, we can naturally reduce the satisfiability problem over structures of treewidth at most $k-1$ to the non-emptiness problem for 2APTAs.[22] Using Proposition 15, we can show that the size of the 2APTA is $2^{\mathrm{poly}(\|\varphi\|,k)}$. Because UNFO has the bounded treewidth model property [47, Theorem 3.4] (Proposition 2), we can give an exponential-time reduction from the satisfiability problem for UNFO to the non-emptiness problem for 2APTAs, which is in ExpTime [48] (Proposition 3). We thus have the following known result.

▶ **Corollary 16** (also shown in [47, Theorem 4.5], even for UNFP)**.** *SAT-UNFO is in* 2ExpTime.

7 Local Checker for SAT-UNTC

In this section, we extend the local checker from UNFO to UNTC.

7.1 A Local Checker

Recall the state set $\mathcal{Q}_{\mathrm{UNTC}}$ (Definition 8). We now define the relation $(\rightsquigarrow) \subseteq \mathcal{Q}_{\mathrm{UNTC}} \times \mathscr{B}_+(\mathcal{Q}_{\mathrm{UNTC}})$ as the minimal binary relation closed under the *rules* in Figure 7. We lift this relation to $(\rightsquigarrow) \subseteq \mathscr{B}_+(\mathcal{Q}_{\mathrm{UNTC}}) \times \mathscr{B}_+(\mathcal{Q}_{\mathrm{UNTC}})$ following § 6.1. All notions of runs, accepting

[22] A minor difference of the LC from 2APTAs is that some auxiliary predicates (*e.g.* $\mathcal{D}^{\bar{\mathbb{A}}}_g$, $\mathcal{N}^{\bar{\mathbb{A}}}_{g,d}$, and $\mathcal{M}^{\bar{\mathbb{A}}}_{g,d}$) are used in the rules of the LC (Figure 6), but we can easily encode them inside 2APTAs.

runs, $\vdash \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$, and $\vdash_{\mathbb{F}} \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$ are defined exactly as before, except that there are some extra rules for UNTC, which we discuss next. The four TC rules above are introduced for obtaining a closure property (Example 17 and § 7.3).

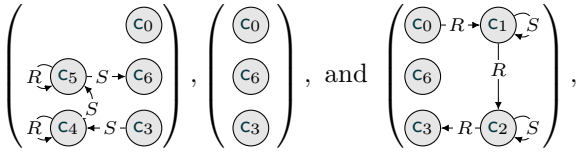
Rules (TC-l) and (TC-r) These rules are obtained by left-unfolding and right-unfolding a TC-formula in a standard way.

Rule (TC-sp-l) This rule is also obtained by unfolding. We confirm the property "$\models \dot{\Gamma}$ *iff* $\models \dot{\psi}$" (where $\dot{\Gamma} \rightsquigarrow \dot{\psi}$) for odd p (the even case is similar). (⇐) Trivial. (⇒) There are some $n \geq 1$ and $d_1, \ldots, d_{n-1} \in$ [-2..2] *s.t.* $\models (\Gamma, \bigwedge_{i=1}^{n} \psi[z_{i-1}z_i/uv])^{p,\bar{\mathbb{A}}}_{\mathscr{I}[\mathcal{N}^{\bar{\mathbb{A}}}_{g,d_1},\ldots,\mathcal{N}^{\bar{\mathbb{A}}}_{g,d_{n-1}}/z_1\ldots z_{n-1}],g}$, where $z_0 = x$, $z_n = y$, and $z_1, \ldots, z_{n-1}$ are fresh. Let $d_0 = d$ and let d_n be such that $e \in \mathcal{N}^{\bar{\mathbb{A}}}_{g,d_n}$. By $d_0 \neq d_n$, there is some $m \in [n]$ such that $d_{m-1} = d$ and $d_m \neq d$. By letting $d' = d_m$, we can show it.

Rule (TC-sp-r) Similar to the above.

The LC is sound and complete (Theorem 20). Below, we give a toy example.

▶ **Example 17.** Let $\bar{\mathbb{A}}$ be the tree in which $\bar{\mathbb{A}}(1)$, $\bar{\mathbb{A}}(\varepsilon)$, and $\bar{\mathbb{A}}(2)$ are given as follows:



Let φ be the UNTC formula $[\exists z(Rxz \wedge Szy)]^*_{xy}xy$ and let $\psi \mathrel{\hat{=}} \exists z(Rxz \wedge Szy)$. Intuitively, φ means that there is a $(RS)^*$-path from x to y. Let $\mathscr{I}$ be such that $\mathscr{I}(x) = \{\mathsf{c}_0\}$ and $\mathscr{I}(y) = \{\mathsf{c}_6\}$. Then $\models (\varphi)^{1,\bar{\mathbb{A}}}_{\mathscr{I},\varepsilon}$ holds, by the form of $\odot\bar{\mathbb{A}}$. In the LC, we construct an accepting run, as follows. First, we have:

$$(\varphi)^{1,\bar{\mathbb{A}}}_{\mathscr{I},\varepsilon} \rightsquigarrow^*_{\text{(TC-l)(conc)}} \geqq_{\mathsf{L}}$$
$$(\psi[z_0^{\mathsf{c}_0} z_1^2/xy], [\psi]^*_{xy} z_1^2 y^{\mathsf{c}_6})^{1,\bar{\mathbb{A}}}_{-,\varepsilon} \rightsquigarrow^*_{\text{(TC-r)(conc)}} \geqq_{\mathsf{L}} \psi[z_0^{\mathsf{c}_0} z_1^2/xy], [\psi]^*_{xy} z_1^2 z_5^1, \psi[z_5^1 z_6^{\mathsf{c}_6}/xy]^{1,\bar{\mathbb{A}}}_{-,\varepsilon}.$$

We then split the formula $[\psi]^*_{xy} z_1 z_5$ as follows:

★
$$(\psi[z_0^{\mathsf{c}_0} z_1^2/xy],\ \underline{[\psi]^*_{xy} z_1^2 z_5^1},\ \psi[z_5^1 z_6^{\mathsf{c}_6}/xy])^{1,\bar{\mathbb{A}}}_{-,\varepsilon}$$
$$\rightsquigarrow^*_{\text{(TC-sp-l)(conc)}} \geqq_{\mathsf{L}} (\psi[z_0^{\mathsf{c}_0} z_1^2/xy], [\psi]^*_{xy} z_1^2 z_2^2,\ \underline{\psi[z_2^2 z_4^1/xy]},\ [\psi]^*_{xy} z_4^1 z_5^1, \psi[z_5^1 z_6^{\mathsf{c}_6}/xy])^{1,\bar{\mathbb{A}}}_{-,\varepsilon}$$
$$\rightsquigarrow^*_{(\wedge)(\exists)\text{(conc)}} \geqq_{\mathsf{L}} (\psi[z_0^{\mathsf{c}_0} z_1^2/xy], [\psi]^*_{xy} z_1^2 z_2^2,\ Rz_2^2 z_3^{\mathsf{c}_3},\ Sz_3^{\mathsf{c}_3} z_4^1,\ [\psi]^*_{xy} z_4^1 z_5^1, \psi[z_5^1 z_6^{\mathsf{c}_6}/xy])^{1,\bar{\mathbb{A}}}_{-,\varepsilon}$$
$$\rightsquigarrow_{\text{(split)}} (\psi[z_0^{\mathsf{c}_0} z_1^2/xy], [\psi]^*_{xy} z_1^2 z_2^2, Rz_2^2 z_3^{\mathsf{c}_3})^{1,\bar{\mathbb{A}}}_{-,\varepsilon} \ \wedge\ (Sz_3^{\mathsf{c}_3} z_4^1, [\psi]^*_{xy} z_4^1 z_5^1, \psi[z_5^1 z_6^{\mathsf{c}_6}/xy])^{1,\bar{\mathbb{A}}}_{-,\varepsilon}.$$

Here, in the step using (TC-sp-l), the assignment $z_4 \mapsto \{1\}$ is guessed, which indicates a fixed name that escapes from direction 2 during the iterations of TC.

Left-hand side We have:

$$(\psi[z_0^{\mathsf{c}_0} z_1^2/xy], [\psi]^*_{xy} z_1^2 z_2^2, Rz_2^2 z_3^{\mathsf{c}_3})^{1,\bar{\mathbb{A}}}_{-,\varepsilon} \rightsquigarrow_{\text{(move)(conc)}} \geqq_{\mathsf{L}} (\psi[z_0^{\mathsf{c}_0} z_1^{\mathsf{c}_1}/xy], [\psi]^*_{xy} z_1^{\mathsf{c}_1} z_2^{\mathsf{c}_2}, Rz_2^{\mathsf{c}_2} z_3^{\mathsf{c}_3})^{1,\bar{\mathbb{A}}}_{-,2}$$
$$\rightsquigarrow^*_{\text{(split)}(\exists)\text{(conc)}(\wedge)(\alpha)} \geqq_{\mathsf{L}} ([\psi]^*_{xy} z_1^{\mathsf{c}_1} z_2^{\mathsf{c}_2})^{1,\bar{\mathbb{A}}}_{-,2} \rightsquigarrow^*_{\text{(TC-l)(TC-l)(conc)}} \geqq_{\mathsf{L}} (\psi[z_1^{\mathsf{c}_1} z^{\mathsf{c}_2}/xy], z^{\mathsf{c}_2} = z_2^{\mathsf{c}_2})^{1,\bar{\mathbb{A}}}_{-,2}$$
$$\rightsquigarrow^* \geqq_{\mathsf{L}} \mathsf{true}.$$

Right-hand side Similarly, we have:

$$(Sz_3^{\mathsf{c}_3} z_4^1, [\psi]^*_{xy} z_4^1 z_5^1, \psi[z_5^1 z_6^{\mathsf{c}_6}/xy])^{1,\bar{\mathbb{A}}}_{-,\varepsilon} \rightsquigarrow^*_{\text{(move)(conc)}} (Sz_3^{\mathsf{c}_3} z_4^{\mathsf{c}_4}, [\psi]^*_{xy} z_4^{\mathsf{c}_4} z_5^{\mathsf{c}_5}, \psi[z_5^{\mathsf{c}_5} z_6^{\mathsf{c}_6}/xy])^{1,\bar{\mathbb{A}}}_{-,1}$$
$$\rightsquigarrow^* \geqq_{\mathsf{L}} \mathsf{true}.$$

Hence, we have $\vdash (\varphi)^{1,\bar{\mathbb{A}}}_{\mathscr{I},\varepsilon}$.

7.2 Evaluation Strategy

We now present a strategy to evaluate $\models \Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g}$ in the LC. Step 2 is crucial for the 2ExpTime upper bound.

Step 1 By the same strategy as Steps 1 and 2 for UNFO (§ 6.2), we can assume that

a) $\#\mathscr{I}(x) = 1$ for each $x \in \mathsf{FV}(\Gamma)$.

b) Γ is of the form $(\psi_1, \dots, \psi_n)$, where each ψ_i is either an atom α, $\neg\rho$, or $[\rho]^*_{uv}xy$.

c) for each ψ_i, for a $d \in \{\text{-}2, \text{-}1, 1, 2\}$, $d \in \mathscr{I}(\mathsf{FV}(\psi_i)) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d}$, except for the case when ψ_i is a TC-formula.

Step 2 When ψ_i is a TC-formula $[\rho]^*_{uv}xy$, we divide $\Gamma = (\Gamma', \Gamma'')$ into the following four cases, according to whether ψ_i is left- and/or right-unfolded:

i) $\Gamma'' = (\quad [\rho]^*_{uv}xy)$ (observe that $\mathsf{FV}(\Gamma') \cap \mathsf{FV}(\Gamma'') \subseteq \{x, y\}$),

ii) $\Gamma'' = (\Delta, [\rho]^*_{uv}xy)$ where $\mathsf{FV}(\Gamma') \cap \mathsf{FV}(\Gamma'') \subseteq \{y\}$,

iii) $\Gamma'' = (\quad [\rho]^*_{uv}xy, \Lambda)$ where $\mathsf{FV}(\Gamma') \cap \mathsf{FV}(\Gamma'') \subseteq \{x\}$,

iv) $\Gamma'' = (\Delta, [\rho]^*_{uv}xy, \Lambda)$ where $\mathsf{FV}(\Gamma') \cap \mathsf{FV}(\Gamma'') = \emptyset$,

where Δ and Λ are formula sets obtained by left-unfolding and right-unfolding some $[\rho]^*_{uv}{-}{-}$, respectively. Note that $\mathsf{FV}(\Delta) \cap \mathsf{FV}([\rho]^*_{uv}xy) \subseteq \{x\}$ and $\mathsf{FV}([\rho]^*_{uv}xy) \cap \mathsf{FV}(\Lambda) \subseteq \{y\}$ also hold, as free variables are always replaced with fresh variables in unfolded formulas in the rules for TC.

If $\mathscr{I}(x) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,0}$, we apply (split) and (TC-l). For instance, for the case ii), we have:

$$(\Gamma',\ \Delta, [\rho]^*_{uv}xy)^{1,\bar{\mathbb{A}}}_{\mathscr{I},g} \rightsquigarrow_{\text{(split)}} (\Delta)^{1,\bar{\mathbb{A}}}_{-,g} \wedge (\Gamma',\ [\rho]^*_{uv}xy)^{1,\bar{\mathbb{A}}}_{-,g}$$
$$\rightsquigarrow_{\text{(TC-l)}} (\Delta)^{1,\bar{\mathbb{A}}}_{-,g} \wedge (\Gamma',\ \rho[xz^{\mathcal{D}^{\bar{\mathbb{A}}}_g}/uv], [\rho]^*_{uv}zy)^{1,\bar{\mathbb{A}}}_{-,g}.$$

The transformed formula set is still in the case ii).

If $\mathscr{I}(y) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,0}$, we apply (split) and (TC-r), similarly. After that, we can assume that

- for some $d, d' \in \{\text{-}2, \text{-}1, 1, 2\}$, $\mathscr{I}(x) = \{d\}$ and $\mathscr{I}(y) = \{d'\}$.

When $d \neq d'$, we apply (TC-sp-l) or (TC-sp-r). For instance, for the case ii), we have:

$$\underbrace{(\Gamma',\ \Delta, [\rho]^*_{uv}xy)}_{\text{Case ii)}}{}^{1,\bar{\mathbb{A}}}_{\mathscr{I},g} \rightsquigarrow_{\text{(TC-sp-l)}} \geqq_{\mathsf{L}} (\Gamma',\ \Delta, [\rho]^*_{uv}xz,\ \underline{\rho[zz'/uv]},\ [\rho]^*_{uv}z'y)^{1,\bar{\mathbb{A}}}_{\mathscr{I},g}.$$

After applying this argument recursively to the formula $\rho[zz'/uv]$, we eventually reach two sets Δ' and Λ' as follows:

$$\dots \rightsquigarrow^* \geqq_{\mathsf{L}} (\Gamma',\ \Delta, [\rho]^*_{uv}xz, \Delta',\ \Lambda', [\rho]^*_{uv}z'y)^{1,\bar{\mathbb{A}}}_{\mathscr{I},g},$$

where Δ' is obtained by left-unfolding some $[\rho]^*_{uv}x{-}$ and Λ' is obtained by right-unfolding some $[\rho]^*_{uv}{-}y$, *cf.*, the lines (★) in Example 17; the formula $[\psi]^*_{xy}z^2_1z^1_5$ is split to $\Delta' = ([\psi]^*_{xy}z^2_1z^2_2, Rz^2_2z^{c_3}_3)$ and $\Lambda' = (Sz^{c_3}_3z^1_4, [\psi]^*_{xy}z^1_4z^1_5)$. We can then apply (split), as follows:

$$\dots \rightsquigarrow_{\text{(split)}} \underbrace{(\Delta, [\rho]^*_{uv}xz, \Delta')}_{\text{Case iv)}}{}^{1,\bar{\mathbb{A}}}_{\mathscr{I},g} \wedge \underbrace{(\Gamma', \Lambda', [\rho]^*_{uv}z'y)}_{\text{Case ii)}}{}^{1,\bar{\mathbb{A}}}_{\mathscr{I},g}.$$

Both the transformed formula sets are still in the cases iv) and ii), respectively. After the step, we can assume that

c') for each ψ_i, for a $d \in \{\text{-}2, \text{-}1, 1, 2\}$, $d \in \mathscr{I}(\mathsf{FV}(\psi_i)) \subseteq \mathcal{N}^{\bar{\mathbb{A}}}_{g,d}$.

Step 3 Finally, by the condition c'), we apply (split)(move) in the same way as Steps 3 and 4 for UNFO, and then we go back to Step 1.

7.3 Closure Property

Based on the evaluation strategy above, we can obtain a completeness with a closure property (Theorem 20). The following closure is inspired by that for *derivatives* in *regular expressions* [11, 4] and in the positive calculus of relations with transitive closure [37, 40] and by *Fischer–Ladner closure* in PDL [23]. The main difference from them is that our closure has *two* unfoldings: left-unfolding and right-unfolding, based on the rules for TC.

▶ **Definition 18.** *For a UNTC formula φ, the* uni-closure $\mathrm{ucl}(\varphi)$ *is the set of UNTC formula sets defined as follows:*

$$\mathrm{ucl}([\varphi]^*_{uv}xy) \mathrel{\hat{=}} \bigcup_{\substack{z_1 \text{ is } x \text{ or fresh} \\ z_2 \text{ is } y \text{ or fresh}}} \mathrm{ucl}(z_1 = z_2) \cup \bigcup_{z_1, z_2 \text{ are fresh}} \mathrm{ucl}(\varphi[z_1z_2/uv]) \quad \cup \quad \{([\varphi]^*_{uv}xy)\} \tag{i}$$

$$\cup \left\{ (\Delta, [\varphi]^*_{uv}z_2z_3) \mid \Delta \in \mathrm{ucl}(\varphi[z_1z_2/uv]),\ \text{where } z_1, z_2 \text{ are fresh,} \right. \\ \left. z_3 \text{ is } y \text{ or fresh, and } \mathsf{FV}(\Delta) \cap \{z_2, z_3\} \subseteq \{z_2\} \right\} \tag{ii}$$

$$\cup \left\{ ([\varphi]^*_{uv}z_1z_2, \Lambda) \mid \Lambda \in \mathrm{ucl}(\varphi[z_2z_3/uv]),\ \text{where } z_2, z_3 \text{ are fresh,} \right. \\ \left. z_1 \text{ is } x \text{ or fresh, and } \{z_1, z_2\} \cap \mathsf{FV}(\Lambda) \subseteq \{z_2\} \right\} \tag{iii}$$

The other definitions are given based on the cl *of Definition 12.*

▶ **Definition 19.** *For a UNTC formula φ, the* bi-closure $\mathrm{bicl}(\varphi)$ *is defined as follows:*

$$\mathrm{bicl}(\varphi) \mathrel{\hat{=}} \mathrm{ucl}(\varphi) \cup$$

$$\left\{ (\Delta, [\psi]^*_{uv}z_2z_3, \Lambda) \mid [\psi]^*_{uv}xy \text{ is a subformula of } \varphi,\ \Delta \in \mathrm{ucl}(\psi[z_1z_2/uv]),\ \text{and} \right. \\ \Lambda \in \mathrm{ucl}(\psi[z_3z_4/uv]),\ \text{where } z_1, z_2, z_3, z_4 \text{ are fresh, } \mathsf{FV}(\Delta) \cap \{z_2, z_3\} \subseteq \{z_2\}, \\ \left. \{z_2, z_3\} \cap \mathsf{FV}(\Lambda) \subseteq \{z_3\},\ \text{and } \mathsf{FV}(\Delta) \cap \mathsf{FV}(\Lambda) = \emptyset \right\} \tag{iv}$$

We recall the case iv) in Step 2 of the evaluation strategy presented in § 7.1. In this case, by applying (split), we can split to Γ' and $(\Delta, [\rho]^*_{uv}xy, \Lambda)$, and thus we can assume $\Gamma' = \emptyset$. Namely, it suffices to consider bi-unfolded formula sets at the *outermost*. The definition of bicl (Definition 19) is based on this fact. We observe that this extension causes only a quadratic, not an exponential blowup, which is crucial to obtain the 2ExpTime upper bound.[23] Also, note that *two* formula sets in $\mathrm{bicl}(\varphi)$ may simultaneously appear between (TC-sp-l) and (split) (*cf.*, the lines of Example 17 (★)). Based on this, we define the *closure* set $\mathrm{cl}(\varphi)$, as follows: $\mathrm{cl}(\varphi) \mathrel{\hat{=}} \{\Gamma \cup \Delta \mid \Gamma, \Delta \in \mathrm{bicl}(\varphi)\}$. By the strategy presented in § 7.1, we have that the LC is sound and complete *w.r.t.* the semantics on tree decompositions under a closure property, as follows.

▶ **Theorem 20.** *Let φ be a UNTC formula. For all $\dot{\Gamma} \in \mathcal{Q}_{\mathrm{bicl}(\varphi)}$, we have: $\models \dot{\Gamma}$ iff $\vdash_{\mathrm{cl}(\varphi)} \dot{\Gamma}$.*

For the size of the closure set, we have the following.

▶ **Proposition 21.** *For all UNTC formulas φ, the cardinality of $\mathrm{ucl}(\varphi)$, up to renaming free variables, is at most $(2\|\varphi\|)^{2\|\varphi\|}$.*

Proof. By easy induction on φ. ◀

▶ **Proposition 22.** *For UNTC formulas φ, the number of $\Gamma^{p,\bar{\mathbb{A}}}_{\mathscr{I},g} \in \mathcal{Q}^{(k)}_{\mathrm{cl}(\varphi)}$ is $2^{\mathcal{O}(\|\varphi\|(\log\|\varphi\|+k))}$, up to forgetting $\mathscr{I}(x)$ for $x \notin \mathsf{FV}(\Gamma)$, $\bar{\mathbb{A}}$, and g and up to renaming free variables.*

Proof. Similar to Proposition 15, using Proposition 21. ◀

[23] It causes an exponential blowup if $\mathrm{ucl}([\varphi]^*_{uv}xy)$ is defined so that it contains all bi-unfolded formulas $(\Delta, [\varphi]^*_{uv}zz', \Lambda)$ (and if the nesting of TC-formulas is unbounded).

7.4 Reduction to 2APTAs

Similar to § 6.4, using the LC, we can give an exponential-time reduction from SAT-UNTC to the non-emptiness problem for 2APTAs using the LC for UNTC. Using Proposition 22, we can show that the size of the 2APTA is $2^{\mathrm{poly}(\|\varphi\|,k)}$. Thus, in the same vein as § 6.4, we have the following result.

▶ **Corollary 23** (*cf.* [20, Theorem 8.5]). *SAT-UNTC is in* 2ExpTime.

Combining this with Theorem 5 yields the following main result.

▶ **Corollary 24.** *SAT-GNTC is in* 2ExpTime.

8 Model Checking

In this section, we study the combined complexity of the *model checking problem*: given a sentence φ and a structure $\mathbb{A}$, to decide whether $\mathbb{A} \models \varphi$. Here, the size of a structure $\mathbb{A}$ is defined as the sum of the number of domain elements and the number of tuples. For UNFO and GNFO, the following complexity results are known. Here, $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$ is the class of decision problems solvable by a P machine with $\mathcal{O}(\log^2 n)$ accesses to an NP oracle, where n is the input length.

▶ **Proposition 25** ([47, 13]). *The model checking problem is* $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$*-complete for both UNFO [47] and GNFO [13].*

However, the precise complexity was left open for UNTC (and thus also for GNTC) [20, Open Problem 2 and Footnote 16 of version v1]. Nevertheless, the above complexity bounds continue to hold even if we extend with guarded TC-formulas.[24]

The basic idea is the reduction [47] from UNFO to "Tree Block Satisfaction" $\mathbf{TB}(\mathrm{SAT})_{1\times\mathrm{M}}$. The problem $\mathbf{TB}(\mathrm{SAT})_{q\times\mathrm{M}}$ is in $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$ for every fixed $q \geq 1$ [46, Corollary 3.5]. Transitive closure can still be handled by using $\mathbf{TB}(\mathrm{SAT})_{2\times\mathrm{M}}$ instead of $\mathbf{TB}(\mathrm{SAT})_{1\times\mathrm{M}}$. Building on this idea, we can extend the reduction to UNTC. For GNTC, we can reduce the model checking problem to that of UNTC by adding a new domain element for each tuple, *cf.*, [13, Theorem 5.1] for GNFO and GNFP.[25]

▶ **Theorem 26.** *The model checking problem is* $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$*-complete for UNTC and GNTC.*

As a corollary, the model checking problem for $\mathrm{UNFO}^{\mathrm{reg}}$ is also $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-complete, which provides a revised proof of [32, Theorem 18]; see also [20, Footnote 16 of version v1] for a report on the error of the proof presented in [32, Theorem 18].

In our proof, we introduce a useful logic $\mathbf{Where}(\mathrm{EFO})_{q,k}$. The syntax of $\mathbf{Where}(\mathrm{EFO})_{q,k}$ *formulas* is given by the following grammar:

$$\hat{\varphi} \ ::= \ \psi \ \mathsf{where} \ (X_1 x_{1,1} \dots x_{1,\check{k}_1} \leftarrow \hat{\varphi}_1, \ \dots, \ X_m x_{m,1} \dots x_{m,\check{k}_m} \leftarrow \hat{\varphi}_m),$$

where

- ψ is an *EFO formula* (*existential FO formula*) over $\mathcal{R} \sqcup \{X_1, \dots, X_m\}$ such that
  - X_j *occurs at most* q *times* for each $j \in [m]$,

[24] *Cf.*, P^{NP}-complete for LFP and even the *alternation-free* fragment of *UNFP* [47, Theorem 5.5].

[25] Recall that the size of the structure counts both domain elements and tuples. Note that the domain of the resulting structure is exponentially larger than that of the original structure when the maximal arity of atoms is unbounded. In contrast, for UNTC (and also $\mathbf{Where}(\mathrm{EFO})_{q,k}$), the model checking problem is $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-complete even when the size is defined as the number of domain elements.

- $m \geq 0$, $X_1, \ldots, X_m$ are pairwise distinct, and for each $j \in [m]$,
  - $\breve{k}_j \leq k$, and
  - $\mathsf{FV}(\hat{\varphi}_j) \subseteq \{x_{j,1}, \ldots, x_{j,\breve{k}_j}\}$ and the variables $x_{j,1}, \ldots, x_{j,\breve{k}_j}$ are pairwise distinct, where $\hat{\varphi}_j$ is a $\mathbf{Where}(\mathrm{EFO})_{q,k}$ formula.

The arity of X_i is bounded by k, while the arity of $R \in \mathcal{R}$ is not bounded.

Each $\mathbf{Where}(\mathrm{EFO})_{q,k}$ formula $\hat{\varphi}$ naturally expresses an *FO formula* $\hat{\varphi}^\flat$, more precisely, $\hat{\varphi}^\flat$ is defined as the formula ψ in which each occurrence $X_j\bar{z}$ has been replaced with $\hat{\varphi}_j^\flat[\bar{z}/\bar{y}_j]$ (possibly with renaming bound variables in $\hat{\varphi}_j^\flat$ to avoid naming conflicts).

For instance, if $\hat{\varphi}$ is the following $\mathbf{Where}(\mathrm{EFO})_{2,3}$ formula

$$(\exists z\ Xxz \wedge Xzy) \wedge Yw \text{ where } (\\ Xxy \leftarrow x = y \vee Exy \text{ where } (), Yx \leftarrow \exists w \neg Zxxw \text{ where } (Zxyz \leftarrow Eyz)),$$

then $\hat{\varphi}^\flat$ is $(\exists z((x = z \vee Exz) \wedge (z = y \vee Ezy))) \wedge \exists w' \neg Eww'$. The satisfaction relation $\mathbb{A} \models_I \hat{\varphi}$ is defined as $\mathbb{A} \models_I \hat{\varphi}^\flat$, using the standard FO satisfaction relation.

The model checking problem for $\mathbf{Where}(\mathrm{EFO})_{q,k}$ can be naturally encoded by $\mathbf{TB}(\mathrm{SAT})_{q\times \mathrm{M}}$, and hence we can show the problem is $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-complete.

▶ **Lemma 27.** *For each fixed integer $q, k \geq 1$, the model checking problem for $\mathbf{Where}(\mathrm{EFO})_{q,k}$ is $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-complete.*

The model checking problem for UNTC can be reduced to the model checking problem for $\mathbf{Where}(\mathrm{EFO})_{2,2}$ in polynomial time, and thus is $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-complete.

Our approach also shows that, in Theorem 26, the condition of unary negation can be generalized to κ-ary negation for any fixed κ (*cf.*, Footnote 11). We say that a formula φ (in TC) is (κ, λ, ξ)-*genUNTC* if (i) each subformula of the form $\neg\psi$ has at most κ free variables: $\#\mathsf{FV}(\psi) \leq \kappa$, (ii) for each subformula of the form $[\psi]^*_{\bar{u}}\bar{x}$, the arity is at most λ: $\#\bar{u} \leq 2\lambda$, and (iii) each subformula of the form $[\psi]^*_{\bar{u}}\bar{x}$ has at most ξ parameters: $\#(\mathsf{FV}(\psi) \setminus \bar{u}) \leq \xi$.

We say that a formula φ (in TC) is (κ, λ, ξ)-*genGNTC* if (i) each negation appears in the form of $(\bigwedge_{i=1}^{\kappa} \alpha_i) \wedge \neg\psi$ where $\mathsf{FV}(\psi) \subseteq \mathsf{FV}(\bigwedge_{i=1}^{\kappa} \alpha_i)$, (ii) each TC appears in the form of $[(\bigwedge_{i=1}^{\lambda} \alpha_i) \wedge (\bigwedge_{i=1}^{\lambda} \beta_i) \wedge \psi]^*_{\bar{u},\bar{v}}\bar{x}\bar{y}$ where $\bar{u} \subseteq \mathsf{FV}(\bigwedge_{i=1}^{\lambda} \alpha_i)$ and $\bar{v} \subseteq \mathsf{FV}(\bigwedge_{i=1}^{\lambda} \beta_i)$, and (iii) each subformula of the form $[\psi]^*_{\bar{u}}\bar{x}$ has at most ξ parameters: $\#(\mathsf{FV}(\psi) \setminus \bar{u}) \leq \xi$. Note that $(1,1,0)$-genUNTC coincides with UNTC and $(1,1,0)$-genGNTC coincides with GNTC, respectively. Additionally, (κ, λ, ξ)-genGNTC has at least as much expressive power as (κ, λ, ξ)-genUNTC for every $\kappa, \lambda, \xi \geq 0$.

Similar to Theorem 26, the model checking problem for (κ, λ, ξ)-genUNTC can be reduced to that for $\mathbf{Where}(\mathrm{EFO})_{2,\max(\kappa,2\lambda+\xi)}$ in polynomial time, and hence we can show the problem is $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-complete.

▶ **Theorem 28.** *For every fixed integer $\kappa \geq 1$ and $\lambda, \xi \geq 0$, the model checking problem is $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-complete for both (κ, λ, ξ)-genUNTC and (κ, λ, ξ)-genGNTC.*

9 Conclusion

We have given a polynomial-time translation from GNTC to UNTC preserving satisfiability and finite-satisfiability (Theorem 5), and hence the satisfiability problem for GNTC is in 2ExpTime (Corollary 24). We have also given a local checker yielding a direct single exponential-time reduction from SAT-UNTC to the non-emptiness problem for 2APTAs (Corollary 23). Furthermore, we have shown that the model checking problems for UNTC, GNTC, (κ, λ, ξ)-genUNTC, and (κ, λ, ξ)-genGNTC are $\mathsf{P}^{\mathsf{NP}[\mathcal{O}(\log^2 n)]}$-complete (§ 8). A

natural direction would be to extend GNTC while preserving the 2ExpTime satisfiability problem. For instance, we would like to explore the possibility of extending GNTC to the *clique-guarded negation fragment* [13, Section 7] or to unparameterized guarded LFP operators [13], possibly by extending the local checker. Regarding the second extension, it would also be interesting to find a fragment of GNFP-UP including both GNTC and GNFP.

As noted in [6, p. 5], the canonical translation of GNTC into GNFP-UP yields formulas with unbounded parameter depth (*pdepth*) – a syntactic measure of the maximum number of nested parameter changes within a formula. However, we currently lack the model-theoretic tools to show that GNTC is not captured by any fragment of GNFP-UP of bounded *pdepth*.[26]

We leave open the decidability of the finite-satisfiability problem of GNTC. While the finite-satisfiability problem of GNTC is not harder than the finite-satisfiability problem of UNTC (Theorem 5), decidability is open for UNTC [20, Open Question 1] and even for loop-PDL [18] (see also [29, 20]). However, the problem is known to be decidable, *e.g.*, for GNFP [13] (see also [47, 8, 12]) and UNFO with transitive relations [19].

References

1 ISO/IEC JTC 1/SC 32. ISO/IEC 9075-16:2023. Information technology — Database languages SQL. Part 16: Property Graph Queries (SQL/PGQ). Technical report, ISO, 2023. URL: https://www.iso.org/standard/79473.html.

2 ISO/IEC JTC 1/SC 32. ISO/IEC 39075:2024. Information technology — Database languages — GQL. Technical report, ISO, 2024. URL: https://www.iso.org/standard/76120.html.

3 Hajnal Andréka, István Németi, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *J. Philos. Log.*, 27(3):217–274, 1998. doi:10.1023/A:1004275029985.

4 Valentin Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theoretical Computer Science*, 155(2):291–319, 1996. doi:10.1016/0304-3975(95)00182-4.

5 Bartosz Bednarczyk and Emanuel Kieroński. Guarded fragments meet dynamic logic: The story of regular guards. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 89–99, 2025. doi:10.24963/kr.2025/9.

6 Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. A step up in expressiveness of decidable fixpoint logics. In *Annual Symposium on Logic in Computer Science (LICS)*, pages 817–826. ACM, 2016. doi:10.1145/2933575.2933592.

7 Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. A step up in expressiveness of decidable fixpoint logics (extended version of [6]), 2016. HAL_id: hal-01413890, version: v1. URL: https://hal.science/hal-01413890v1.

8 Mikołaj Bojańczyk. Two-way alternating automata and finite models. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2380 of *LNCS*, pages 833–844. Springer, 2002. doi:10.1007/3-540-45465-9_71.

9 Pierre Bourhis, Markus Krötzsch, and Sebastian Rudolph. How to best nest regular path queries. In *Informal Proceedings of the 27th International Workshop on Description Logics (DL)*, volume 1193 of *CEUR Workshop Proceedings*, pages 404–415. CEUR-WS.org, 2014. URL: https://ceur-ws.org/Vol-1193/paper_80.pdf.

[26] In particular, the strict *pdepth* hierarchy of [6] is established via a complexity-based argument, separating the elementary satisfiability of bounded *pdepth* from the non-elementary satisfiability of unbounded *pdepth*. Since the satisfiability problem for GNTC is already elementary, this argument cannot be applied. Further, GN^k-bisimulation games do not track the parameter nesting necessary to prove inexpressibility at a given *pdepth* level.

10 Pierre Bourhis, Markus Krötzsch, and Sebastian Rudolph. Reasonable highly expressive query languages. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2826–2832. AAAI Press, 2015. URL: `http://ijcai.org/Abstract/15/400`.

11 Janusz A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964. `doi:10.1145/321239.321249`.

12 Vince Bárány and Mikołaj Bojańczyk. Finite satisfiability for guarded fixpoint logic. *Information Processing Letters*, 112(10):371–375, 2012. `doi:10.1016/j.ipl.2012.02.005`.

13 Vince Bárány, Balder ten Cate, and Luc Segoufin. Guarded negation. *Journal of the ACM*, 62(3):22:1–22:26, 2015. `doi:10.1145/2701414`.

14 Facundo Carreiro and Yde Venema. PDL inside the μ-calculus: A syntactic and an automata-theoretic characterization. In *Advances in Modal Logic (AiML)*, pages 74–93. College Publications, 2014. URL: `http://www.aiml.net/volumes/volume10/Carreiro-Venema.pdf`.

15 Mariano P. Consens and Alberto O. Mendelzon. GraphLog: a visual formalism for real life recursion. In *ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (PODS)*, pages 404–416. ACM, 1990. `doi:10.1145/298514.298591`.

16 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic*. Number 138 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2012. `doi:10.1017/CBO9780511977619`.

17 Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Annual Symposium on Principles of Programming Languages (POPL)*, pages 238–252. ACM, 1977. `doi:10.1145/512950.512973`.

18 Ryszard Danecki. Propositional dynamic logic with strong loop predicate. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 176 of *LNCS*, pages 573–581. Springer, 1984. `doi:10.1007/BFb0030342`.

19 Daniel Danielski and Emanuel Kieroński. Finite satisfiability of unary negation fragment with transitivity. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 138 of *LIPIcs*, pages 17:1–17:15. Schloss Dagstuhl, 2019. `doi:10.4230/LIPIcs.MFCS.2019.17`.

20 Diego Figueira and Santiago Figueira. A common ancestor of PDL, conjunctive queries, and unary negation first-order, 2025. `doi:10.48550/arXiv.2501.11641`.

21 Diego Figueira, Santiago Figueira, and Edwin Pin. PDL on steroids: on expressive extensions of PDL with intersection and converse. In *Annual Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2023. `doi:10.1109/LICS56636.2023.10175813`.

22 Diego Figueira, Anthony W. Lin, and Liat Peterfreund. Complexity of evaluating GQL queries. In *International Conference on Database Theory (ICDT)*, LIPIcs, pages 13:1–13:18. Schloss Dagstuhl, 2026. `doi:10.4230/LIPIcs.ICDT.2026.13`.

23 Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979. `doi:10.1016/0022-0000(79)90046-1`.

24 Jörg Flum. On the (infinite) model theory of fixed-point logics. In *Models, Algebras, and Proofs*. CRC Press, 1998.

25 Amélie Gheerbrant, Leonid Libkin, Liat Peterfreund, and Alexandra Rogova. GQL and SQL/PGQ: Theoretical models and expressive power. *Proc. VLDB Endow. (VLDB)*, 18(6):1798–1810, 2025. `doi:10.14778/3725688.3725707`.

26 Erich Grädel, Martin Otto, and Eric Rosen. Undecidability results on two-variable logics. *Archive for Mathematical Logic*, 38(4-5):313–354, 1999. `doi:10.1007/s001530050130`.

27 Erich Grädel. Guarded fixed point logics and the monadic theory of countable trees. *Theoretical Computer Science*, 288(1):129–152, 2002. `doi:10.1016/S0304-3975(01)00151-7`.

28 Erich Grädel and Igor Walukiewicz. Guarded fixed point logic. In *Annual Symposium on Logic in Computer Science (LICS)*, pages 45–54. IEEE, 1999. `doi:10.1109/LICS.1999.782585`.

29 Stefan Göller, Markus Lohrey, and Carsten Lutz. PDL with intersection and converse: satisfiability and infinite-state model checking. *The Journal of Symbolic Logic*, 74(1):279–314, 2009. doi:10.2178/jsl/1231082313.

30 Neil Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16(4):760–778, 1987. doi:10.1137/0216051.

31 Neil Immerman, Alex Rabinovich, Tom Reps, Mooly Sagiv, and Greta Yorsh. The boundary between decidability and undecidability for transitive-closure logics. In *EACSL Annual Conference on Computer Science Logic (CSL)*, volume 3210 of *LNCS*, pages 160–174. Springer, 2004. doi:10.1007/978-3-540-30124-0_15.

32 Jean Christoph Jung, Carsten Lutz, Mauricio Martel, and Thomas Schneider. Querying the unary negation fragment with regular path expressions. In *International Conference on Database Theory (ICDT)*, volume 98 of *LIPIcs*. Schloss Dagstuhl, 2018. doi:10.4230/LIPIcs.ICDT.2018.15.

33 Orna Kupferman and Moshe Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Logic*, 2(3):408–429, 2001. doi:10.1145/377978.377993.

34 Martin Lange and Carsten Lutz. 2-ExpTime lower bounds for propositional dynamic logics with intersection. *The Journal of Symbolic Logic*, 70(04):1072–1086, 2005. doi:10.2178/jsl/1129642115.

35 David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Alternating automata, the weak monadic theory of the tree, and its complexity. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 6 of *LNCS*, pages 275–283. Springer, 1986. doi:10.1007/3-540-16761-7_77.

36 David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *Annual Symposium on Logic in Computer Science (LICS)*, pages 422–427. IEEE, 1988. doi:10.1109/LICS.1988.5139.

37 Yoshiki Nakamura. Partial derivatives on graphs for Kleene allegories. In *Annual Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2017. doi:10.1109/LICS.2017.8005132.

38 Yoshiki Nakamura. Existential calculi of relations with transitive closure: Complexity and edge saturations. In *Annual Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2023. doi:10.1109/LICS56636.2023.10175811.

39 Yoshiki Nakamura. Undecidability of the positive calculus of relations with transitive closure and difference: Hypothesis elimination using graph loops. In *International Conference on Relational and Algebraic Methods in Computer Science (RAMiCS)*, volume 14787 of *LNCS*, pages 207–224. Springer, 2024. doi:10.1007/978-3-031-68279-7_13.

40 Yoshiki Nakamura. Derivatives on graphs for the positive calculus of relations with transitive closure. *Logical Methods in Computer Science*, Volume 21, Issue 4, 2025. doi:10.46298/lmcs-21(4:27)2025.

41 Yoshiki Nakamura. Undecidability of the emptiness problem of deterministic propositional while programs with graph loop: Hypothesis elimination using loops, 2025. doi:10.48550/arXiv.2504.20415.

42 Damien Pous. On the positive calculus of relations with transitive closure. In *International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 96 of *LIPIcs*, pages 3:1–3:16. Schloss Dagstuhl, 2018. doi:10.4230/LIPIcs.STACS.2018.3.

43 Juan L. Reutter, Miguel Romero, and Moshe Y. Vardi. Regular queries on graph databases. *Theory of Computing Systems*, 61(1):31–83, 2017. doi:10.1007/s00224-016-9676-2.

44 Neil Robertson and Paul D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

45 Hadar Rotschield and Liat Peterfreund. On the expressiveness of languages for querying Property Graphs in relational databases. *Proc. ACM Manag. Data (PODS)*, 3(5):279:1–279:18, 2025. doi:10.1145/3767715.

46 Philippe Schnoebelen. Oracle circuits for branching-time model checking. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2719 of *LNCS*, pages 790–801. Springer, 2003. doi:10.1007/3-540-45061-0_62.

47 Luc Segoufin and Balder ten Cate. Unary negation. *Logical Methods in Computer Science*, Volume 9, Issue 3, 2013. doi:10.2168/LMCS-9(3:25)2013.

48 Moshe Y. Vardi. Reasoning about the past with two-way automata. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1443 of *LNCS*, pages 628–641. Springer, 1998. doi:10.1007/BFb0055090.