Concrete Domains in Logics: A Survey

Stéphane Demri, CNRS, LMF, ENS Paris-Saclay, Université Paris-Saclay
Karin Quaas, Universität Leipzig

In this short survey, we present logical formalisms in which reasoning about concrete domains is embedded in formulae at the atomic level. These include temporal logics with concrete domains, description logics with concrete domains as well as variant formalisms. We discuss several proof techniques to solve logical decision problems for such formalisms, including those based on constrained automata or on translation into decidable second-order logics. We also present recent results mainly related to decidability and complexity as well as a selection of open problems.

1. INTRODUCTION

Reasoning about concrete domains. A concrete domain is a relational structure with a fixed non-empty domain and a family of relations. Typical examples of concrete domains are $(\mathbb{N}; <)$, $(\mathcal{P}(\mathbb{N}); \subseteq)$, and the set of finite words over the alphabet $\{0, 1\}^*$ with the prefix relation. Multiple logical formalisms have been developed to express properties on concrete domains, ranging from quantifier-free languages to first-order languages. A prominent example is Presburger arithmetic [Pre29; Grä88; Haa18] understood as the first-order theory on the natural numbers with addition and the standard ordering relation. Other concrete domains are relevant in computer science, in particular those related to the verification of programs with typed variables (real numbers, finite binary trees, lists, etc.).

Reasoning about concrete domains embedded in logical formalisms can be materialised in at least two ways; combinations are also possible. One option is to keep the semantical structures of the plain logic but to enrich the logical language so that values in the concrete domains can be inferred from the models. This is what is done in [BEH95] in which a version of constrained LTL is introduced that is interpreted over standard LTL models but the logical language is enriched with Presburger constraints to reason about the number of occurrences of events in the models. Graded modal logics are of the same nature, see e.g. [Tob01], as the models are standard Kripke-style structures but the modal language can state constraints about the number of successor worlds satisfying a specific formula.

Another option is to enrich the semantical structures of the plain logic $\mathcal{L}$ with values from a concrete domain $\mathcal{D}$ (see e.g. [BH91]). The logic $\mathcal{L}(\mathcal{D})$ – $\mathcal{L}$ with the concrete domain $\mathcal{D}$ – can express constraints between these values thanks to atomic formulae that can compare these values. For instance, every position of an LTL-model can be associated with two natural numbers. In LTL$(\mathbb{N}; <)$, the atomic formula $x < \mathsf{X}y$ expresses that the value of the variable x at the current position is strictly less than the value of y at the next position.

This paper is mainly dedicated to logical formalisms with concrete domains of the second kind, i.e., extensions of $\mathcal{L}$ in which the semantical structures of the plain logic are augmented with concrete values.

From ontologies to database-driven systems. Logical formalisms in which reasoning about concrete domains is embedded in formulae at the atomic level and the models are Kripke-style structures (transition systems, interpretations for description logics [BHLS17]) are quite ubiquitous in theoretical computer science. Usually, every world (state, domain element) is associated with a collection of values from concrete domains. For example, in timed logics [AH93; AH94; AFH96], states have a timestamp in $\mathbb{R}$ whereas in logics for the verification of counter automata, see e.g. [CC00; BQS19], every state comes with a tuple of counter values in $\mathbb{N}$. This can be gener-

alised to other specification languages for the behaviour of non-terminating programs by understanding valuation sequences as snapshots of program's variables at specific points of time during its execution. Similarly, in description logics with concrete domains handling ontologies with values from concrete domains, domain elements are enriched with tuples of values, see e.g. [BH91; Lut02b; Lut03; Lut04b; LM07]. Dealing with concrete domains in logics happens to be natural and further examples can be found in spatial-temporal logics [WZ00; BC02], in first-order temporal logics for modelling dynamical biological behaviours [FR09; Fag20], in synthesis problems [EFR21; CDOT21], for the verification of hierarchical systems [FK18] and for reasoning about sequences of memory states [BDL09]. Such logical formalisms have been also shown relevant for analysing database-driven systems [DHV14]. For instance, a variant of first-order LTL has been introduced in [DHV14] to verify data-driven web applications.

The expressive power of the plain logics augmented with constraints about concrete domains often allows us to encode counting mechanisms, leading to undecidability of the main reasoning tasks (satisfiability, model-checking), see e.g. [CC00; Lut02b]. However, properties of the concrete domains have been identified to get decidable logical problems, see e.g. [BC02; LM07; CKL16; BR20], and syntactic restrictions have been also elaborated to reduce the expressive power at the gain of having better computational properties, see e.g. [Laz11; BQS19].

Motivations. In view of the plethora of recent works dealing with logics with concrete domains and the diversity of proof methods and decision problems, we would like to provide a short survey about recent advances, in a uniform framework that allows us to compare the different techniques and contributions. Pinpointing the most promising research directions is an important motivation, too.

Content. In this paper, we present results about the decidability/complexity status of reasoning tasks for logics with concrete domains focusing on recent works as well as on the relationships between seemingly distinct formalisms. In Section 2.1, we introduce the notion of concrete domains, we provide examples and we state a few standard results in particular related to numerical domains and to string domains. The remaining part of Section 2 deals with the introduction of temporal and description logics equipped with concrete domains. The automata-based approach is described in Section 3, extending the approach for modal and temporal logics, see e.g. [VW86; VW94]. Decidability results obtained by translation into second-order theories are sketched in Section 4, following recent developments in [Car15; CKL16; CT16]. Comparisons with other proof techniques are also provided there. Whereas the previous sections mainly contain formalisms in which contraints between values from the concrete domains have a local scope, Section 5 presents richer logical formalisms with global constraints (including the use of the freeze operator, and its restriction to express value repetitions) and a selection of decidability/complexity results. Finally, Section 6 concludes the paper with a selection of open questions.

2. LOGICAL FORMALISMS WITH CONCRETE DOMAINS

2.1. Concrete domains and constraints

Concrete domains. A relational signature $\sigma = \{R_1, R_2, \dots\}$ is a countable set of relation symbols, where every symbol R_i has an associated positive arity. A *concrete domain over a relational signature* σ is a tuple $\mathcal{D} = (\mathbb{D}; R_1^{\mathcal{D}}, R_2^{\mathcal{D}}, \dots)$, where $\mathbb{D}$ is the *domain*, and for each $R \in \sigma$, $R^{\mathcal{D}} \subseteq \mathbb{D}^k$ is the interpretation of the relation symbol R in $\mathcal{D}$, where k is the arity of R. We often identify the relation $R^{\mathcal{D}}$ with the relation symbol R; in that case we specify a concrete domain as $(\mathbb{D}; R_1, R_2, \dots)$ or $(\mathbb{D}; \sigma)$.

Let us consider some examples of concrete domains. A generic concrete domain is $(\mathbb{D};\equiv)$, where $\mathbb{D}$ is an arbitrary set of *data values* and $\equiv$ is an equivalence relation over $\mathbb{D}$. We may extend this to $(\mathbb{D};\equiv,(=_{\mathfrak{d}})_{\mathfrak{d}\in\mathbb{D}})$, where $=_{\mathfrak{d}}$ is a unary predicate stating the equality with the constant $\mathfrak{d}$ (the singleton $=_{\mathfrak{d}}$ is also simply written $\mathfrak{d}$ in concrete domains). *Numerical concrete domains* belong to the concrete domains probably studied the most in computer science. We mention here the integers $(\mathbb{Z};+,<,=,0,1)$ that are used for analysing and reasoning about systems with integer variables [CC00; BQS19] and for representing infinite-state systems [Haa18]. Another example is the real numbers $(\mathbb{R};<,=)$, used in the verification of real-timed systems [AH93; AH94; AFH96]. Concrete domains that can be used to *reason about strings* play a key role in the analysis of programs [HW12; AAC$^+$14; KS15; HSZ17]. Let A be a finite or infinite set of letters, in the following called the *alphabet*, and let A^* denote the set of all finite strings over A. Given two finite strings $u,v\in A^*$, we use $u\cdot v$ (often abbreviated by uv) to denote the concatenation of u and v. The pair $(A^*;\cdot)$, where $\cdot$ is a ternary relation interpreted as concatenation, is an example of a concrete domain with domain A^*. There are many interesting partial orders that can be used to form concrete domains with domain A^*; we mention here the following ones. A *subsequence* of a string $v\in A^*$ is a string u that is obtained from v by removing any letters from v; in that case we write $u<_{\text{seq}}v$. If u is contiguous, we call u a *subword* of v, and write $u<_{\text{sbwd}}v$. The *prefix* $<_{\text{pre}}$ and *suffix* $<_{\text{suf}}$ relations are special cases of the subword order, with the obvious respective meanings.

Constraints. Let $\text{VAR}=\{x,y,\ldots\}$ be a countable set of variables. A *term over* VAR is an expression of the form $\mathsf{X}^i x$, where $x\in\text{VAR}$ and X^i is a (possible empty) sequence of i symbols X. A term $\mathsf{X}^i x$ should be understood as a variable (that needs to be interpreted) but, later on, we will see that the prefix X^i will have a temporal interpretation. We write T_{VAR} to denote the set of all terms over VAR. For all $i\in\mathbb{N}$, we write $\text{T}^{\leq i}_{\text{VAR}}$ to denote the subset of terms of the form $\mathsf{X}^j x$, where $j\leq i$. For instance, $\text{T}^{\leq 0}_{\text{VAR}}=\text{VAR}$. Constraints are defined over a concrete domain and a set $T\subseteq\text{T}_{\text{VAR}}$ of terms. Let $\mathcal{D}$ be a concrete domain over a relational signature σ. An *atomic constraint c over $\mathcal{D}$ and T* is an expression of the form $R(\mathtt{t}_1,\ldots,\mathtt{t}_d)$, where $R\in\sigma$ of arity d and $\mathtt{t}_i\in T$, for all $1\leq i\leq d$. A *constraint* C is defined as a (finite) Boolean combination of atomic constraints; we use $\wedge$, $\vee$ and $\neg$ for the standard Boolean connectives. Constraints are interpreted on valuations $\mathfrak{v}:T\to\mathbb{D}$ that assign elements from $\mathbb{D}$ to the terms in T, so that $\mathfrak{v}$ *satisfies* $R(\mathtt{t}_1,\ldots,\mathtt{t}_d)$, written $\mathfrak{v}\models R(\mathtt{t}_1,\ldots,\mathtt{t}_d)$, if and only if, $(\mathfrak{v}(\mathtt{t}_1),\ldots,\mathfrak{v}(\mathtt{t}_d))\in R^{\mathcal{D}}$. The Boolean connectives are interpreted in the usual way. A constraint C over $\mathcal{D}$ and T is *satisfiable* $\stackrel{\text{def}}{\Leftrightarrow}$ there is a valuation $\mathfrak{v}:T\to\mathbb{D}$ such that $\mathfrak{v}\models C$. Similarly, a constraint C_1 *entails* a constraint C_2 (written $C_1\models C_2$) $\stackrel{\text{def}}{\Leftrightarrow}$ for all valuations $\mathfrak{v}$, we have $\mathfrak{v}\models C_1$ implies $\mathfrak{v}\models C_2$. In this document, most of the logics are based on concrete domains whose satisfiability problem restricted to finite conjunctions of atomic constraints can be solved in NP, if not in PTIME.

2.2. Models with concrete domains

In order to define logics with concrete domain $\mathcal{D}$, the semantical structures of such logics (typically, Kripke-style structures) are enriched with valuations that interpret the variables of the logic by elements of the concrete domain.

Kripke structures. Let $\text{PROP}=\{p,q,\ldots\}$ be a countably infinite set of propositional variables. A *Kripke structure* $\mathcal{K}$ is a triple $(\mathcal{W},\mathcal{R},l)$, where $\mathcal{W}$ is a set of *worlds*, $\mathcal{R}\subseteq\mathcal{W}\times\mathcal{W}$ is the accessibility relation and $l:\mathcal{W}\to\mathcal{P}(\text{PROP})$ is a labelling function. A Kripke structure $\mathcal{K}$ is *total* whenever for all $w\in\mathcal{W}$, there is $w'\in\mathcal{W}$ such that $(w,w')\in R$. Totality is a standard property for defining classes of models for temporal logics such

as CTL*. For instance, an LTL model $\mathcal{K} = (\mathcal{W}, R, l)$ is a Kripke structure where $(\mathcal{W}, R)$ is isomorphic to $(\mathbb{N}; \text{succ})$, where succ is the successor relation on $\mathbb{N}$. Given a Kripke structure $\mathcal{K} = (\mathcal{W}, \mathcal{R}, l)$ and a world $w \in \mathcal{W}$, an *infinite path* π from w is an ω-sequence $w_0, \ldots, w_n, \ldots$ such that $w_0 = w$ and for all $i \in \mathbb{N}$, we have $(w_i, w_{i+1}) \in \mathcal{R}$. Finite paths are defined accordingly.

Decorated Kripke structures. A *$\mathcal{D}$-decorated Kripke structure* $\mathcal{K}$ is a structure of the form $(\mathcal{D}, \mathcal{W}, \mathcal{R}, l, \mathfrak{v})$ such that $\mathcal{D} = (\mathbb{D}; \sigma)$ is a concrete domain, $(\mathcal{W}, \mathcal{R}, l)$ is a Kripke structure and $\mathfrak{v} : \mathcal{W} \times \text{VAR} \to \mathbb{D}$ is a valuation function. A $\mathcal{D}$-decorated Kripke structure can be understood as a plain Kripke structure in which to each world is associated a valuation interpreting the variables from VAR by elements in $\mathbb{D}$. We may omit the labelling function l from $\mathcal{K}$ if there is no need for propositional labelling (as in Section 2.3). Similarly, $\mathcal{D}$ might be omitted if it is clear from the context. Note also that numerous variants exist for the above definition, for instance one may admit a family of accessibility relations (instead of a single one) or consider disjoint sets of variables interpreted in distinct concrete domains (instead of a single concrete domain).

2.3. Temporal logics with concrete domains

We introduce temporal logics with concrete domains of the form LTL($\mathcal{D}$) and CTL*($\mathcal{D}$), naturally extending what is known for the temporal logics LTL and CTL*. Other options are possible and some are evoked in the document, but these two cases allow us to illustrate the main features.

Linear case. We write LTL($\mathcal{D}$) to denote the variant of LTL for which models are $\mathcal{D}$-decorated Kripke structures with underlying LTL models. LTL($\mathcal{D}$) formulae are defined below; in short, propositional variables from LTL are replaced by atomic constraints over $\mathcal{D}$ and $\mathrm{T}_{\mathrm{VAR}}$. The set of *LTL($\mathcal{D}$)-formulae* is defined as follows.

$$\phi ::= R(\mathtt{t}_1, \ldots, \mathtt{t}_d) \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{X}\phi \mid \phi \mathsf{U} \phi,$$

where $\mathtt{t}_i$ is a term in $\mathrm{T}_{\mathrm{VAR}}$, and $R \in \sigma$ is of arity d. The temporal operators X and U are interpreted as in LTL. Note that the symbol 'X' is overloaded here (temporal operator and operator in the terms to access future variable values) but this should not cause any confusion in the sequel. Other standard operators like $\rightarrow$, or the temporal operators *sometimes* F and *always* G can be defined in the usual way.

An LTL($\mathcal{D}$) model is a map of the form $\mathfrak{v} : \mathbb{N} \times \text{VAR} \to \mathbb{D}$. Given an LTL($\mathcal{D}$) model $\mathfrak{v}$, we write $\mathfrak{v}(i, \mathsf{X}^j x)$ to denote the value $\mathfrak{v}(i + j, x) \in \mathbb{D}$. The satisfaction relation $\models$ for LTL($\mathcal{D}$) is defined as follows ($i \in \mathbb{N}$).

— $\mathfrak{v}, i \models R(\mathtt{t}_1, \ldots, \mathtt{t}_d) \stackrel{\text{def}}{\Leftrightarrow} (\mathfrak{v}(i, \mathtt{t}_1), \ldots, \mathfrak{v}(i, \mathtt{t}_d)) \in R^{\mathcal{D}}$,
— $\mathfrak{v}, i \models \phi \wedge \psi \stackrel{\text{def}}{\Leftrightarrow} \mathfrak{v}, i \models \phi$ and $\mathfrak{v}, i \models \psi$; $\mathfrak{v}, i \models \neg\phi \stackrel{\text{def}}{\Leftrightarrow}$ not $\mathfrak{v}, i \models \phi$,
— $\mathfrak{v}, i \models \mathsf{X}\phi \stackrel{\text{def}}{\Leftrightarrow} \mathfrak{v}, i + 1 \models \phi$,
— $\mathfrak{v}, i \models \phi \mathsf{U} \psi \stackrel{\text{def}}{\Leftrightarrow}$ there is $j \geq i$ s.t. $\mathfrak{v}, j \models \psi$ and for all $j' \in [i, j-1]$, we have $\mathfrak{v}, j' \models \phi$.

As usual, ϕ is *satisfiable* iff there is a model $\mathfrak{v}$ such that $\mathfrak{v}, 0 \models \phi$. Note that given an atomic constraint $R(\mathtt{t}_1, \ldots, \mathtt{t}_d)$ such that $\mathtt{t}_j = \mathsf{X}^{m_j} x_j$ for all j and $M = \max\{m_1, \ldots, m_d\}$, the satisfaction of $\mathfrak{v}, i \models R(\mathtt{t}_1, \ldots, \mathtt{t}_d)$ only depends on the values of the variables $x_1, \ldots, x_d$ at the positions $i, \ldots, i + M$ (understood as a finite path).

Recall that satisfiability for LTL is PSPACE-complete [SC85]. A natural question is whether solving the satisfiability problem for LTL($\mathcal{D}$), denoted by SAT(LTL($\mathcal{D}$)), is computationally more complex than for plain LTL; obviously, this depends on $\mathcal{D}$. In comparison with plain LTL, new constraints between variable values need to be satisfied due to the interpretation to the specific concrete domain $\mathcal{D}$. Moreover, as terms can refer to the values of variables at the next position, the propagation of constraints

may lead to a possibly infinite "network" of constraints (known as constraint graphs, see e.g. [Lut04a; DD07; LOS20]). For instance, in LTL$(\mathbb{N};<)$, no infinite sequence of variables with strictly decreasing values is admitted; so $\mathsf{G}(x > \mathsf{X}x)$ is not satisfiable. Considering another concrete domain, the formula $0 \cdot x = \mathsf{X}x \rightarrow \mathsf{X}(1 \cdot x = \mathsf{X}x)$ in LTL$(\{0,1\}^*; 0\cdot, 1\cdot)$ where $0\cdot$ is a binary relation concatenating the letter 0 (similarly for $1\cdot$). is interpreted by "If 0 is popped from the 'stack' x, then at the next step 1 is pushed onto it."

One of the first results established for LTL$(\mathcal{D})$ can be found in [BC02], where $\mathrm{SAT}(\text{LTL}(\mathbb{R};<,=))$ and $\mathrm{SAT}(\text{LTL}(\mathbb{Q};<,=))$ are shown in PSPACE by adapting the proof method from [SC85] to decide LTL satisfiability in PSPACE. In [DD07; ST11], conditions generalising the above result are identified that guarantee a PSPACE upper bound. Besides, $\mathrm{SAT}(\text{LTL}(\{0,1\}^*; <_{\text{pre}}, =))$ is in PSPACE, too (see e.g. [KW15; DD16]) although $(\{0,1\}^*; <_{\text{pre}}, =)$ and $(\mathbb{Q};<,=)$ are quite different structures. When a concrete domain $\mathcal{D}$ is able to encode a counting mechanism, LTL$(\mathcal{D})$ becomes quite expressive, for instance $\mathrm{SAT}(\text{LTL}(\mathbb{N}; \text{succ}, =))$ is undecidable [CC00]. A natural question is which properties of $\mathcal{D}$ make $\mathrm{SAT}(\text{LTL}(\mathcal{D}))$ decidable, or even better in PSPACE. We shall see that several answers have been proposed in the literature.

Branching case. Next, we introduce the temporal logic CTL$^*(\mathcal{D})$ understood as the branching-time extension of LTL$(\mathcal{D})$. *State formulae* ϕ and *path formulae* Φ of CTL$^*(\mathcal{D})$ are defined below

$$\phi := \neg\phi \mid \phi \wedge \phi \mid \mathsf{E}\Phi \qquad \Phi := \phi \mid R(\mathtt{t}_1, \ldots, \mathtt{t}_d) \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathsf{X}\Phi \mid \Phi\mathsf{U}\Phi,$$

where $\mathtt{t}_i$ is a term in $\mathrm{T}_{\mathrm{VAR}}$, and $R \in \sigma$ is of arity d. State formulae are interpreted on worlds from a $\mathcal{D}$-decorated Kripke structure, whereas path formulae are interpreted on LTL$(\mathcal{D})$ models (or, equivalently, on infinite paths from $\mathcal{D}$-decorated Kripke structures). The two satisfaction relations are defined as follows (we omit the standard clauses for Boolean connectives), where $\mathcal{K} = (\mathcal{D}, \mathcal{W}, \mathcal{R}, \mathfrak{v})$ is a total $\mathcal{D}$-decorated Kripke structure, $w \in \mathcal{W}$, and π is an infinite path of $\mathcal{K}$.

— $\mathcal{K}, w \models \mathsf{E}\Phi \stackrel{\text{def}}{\Leftrightarrow}$ there is an infinite path π starting from w such that $\mathcal{K}, \pi \models \Phi$,
— $\mathcal{K}, \pi \models R(\mathtt{t}_1, \ldots, \mathtt{t}_d) \stackrel{\text{def}}{\Leftrightarrow} (\mathfrak{v}(\pi(0), \mathtt{t}_1), \ldots, \mathfrak{v}(\pi(0), \mathtt{t}_d)) \in R^{\mathcal{D}}$, where $\mathfrak{v}(\pi(0), \mathsf{X}^j x) \stackrel{\text{def}}{=} \mathfrak{v}(w_j, x)$ with w_j the jth world of π,
— $\mathcal{K}, \pi \models \mathsf{X}\Phi \stackrel{\text{def}}{\Leftrightarrow} \mathcal{K}, \pi[1, +\infty[\models \Phi$, where for any n, $\pi[n, +\infty[$ is the suffix of π truncated by the n first worlds,
— $\mathcal{K}, \pi \models \Phi\mathsf{U}\Psi \stackrel{\text{def}}{\Leftrightarrow}$ there is $j \geq 0$ such that $\mathcal{K}, \pi[j, +\infty[\models \Psi$ and for all $j' \in [i, j-1]$, we have $\mathcal{K}, \pi[j', +\infty[\models \Phi$.

Several results from [BC02] can can be adapted to the branching case for a quite large family of concrete domains, see e.g. [Gas09]. For instance, $\mathrm{SAT}(\text{CTL}^*(\mathbb{R};<,=))$ and $\mathrm{SAT}(\text{CTL}^*(\mathbb{Q};<,=))$ are in 2EXPTIME (see also Section 3.3). Decidability of strict fragments of CTL$^*(\mathbb{Z};<,=,(=_n)_{n\in\mathbb{Z}})$ is shown in [BG06; Gas07]. It is only recently in [Car15; CKL16] that decidability has been established for the full logic using a translation into a decidable second-order logic (details are provided in Section 4).

THEOREM 2.1 ([CKL16]). $\mathrm{SAT}(\textit{CTL}^*(\mathbb{Z};<,=,(=_n)_{n\in\mathbb{Z}}))$ *is decidable.*

The proof of Theorem 2.1 does not provide a sharp complexity upper bound and actually apart from determining the decidability status of logics of the form CTL$^*(\mathcal{D})$, characterising the computational complexity is a general issue too. By contrast, $\mathrm{SAT}(\text{LTL}(\mathbb{Z};<,=,(=_n)_{n\in\mathbb{Z}}))$ is known to be in PSPACE [DG08; ST11] (with integers encoded in binary).

DL notation	TL notation
concept name A; concept $\exists r.C$	prop. variable p; formula $\mathsf{EX}_r\ \phi$
path constraint $\exists P.\ R(S^{i_1}x_1, \ldots, S^{i_d}x_d)$ [CT16]	atomic constraint $\mathsf{E}P.\ R(\mathsf{X}^{i_1}x_1, \ldots, \mathsf{X}^{i_d}x_d)$
concept $\exists p_1, \ldots, p_d.\ R(x_1, \ldots, x_d)$ [BR20]	atomic constraint $R(\mathsf{E}P_1\ x_1, \ldots, \mathsf{E}P_d\ x_d)$

Fig. 1. A selection of correspondences between DLs and TLs

In order to conclude this section, we would like to note that it is also possible to define logics of the form LTL(Δ) or CTL*(Δ) where Δ is a *class* of concrete domains instead of a fixed concrete domain $\mathcal{D}$ as in CTL*($\mathcal{D}$). For instance, in [Car15; CKL16], the class of semi-orders is taken for Δ.

2.4. Description logics with concrete domains

Concrete domains are also handled in description logics by adding concrete values in ontologies, following the seminal work [BH91]. Since [Sch91], correspondences between modal logics and description logics are well identified and this applies also to relationships with temporal logics. Below, we present several ways to consider concrete domains in description logics. To avoid the introduction of too lengthy definitions, we present the new logics following the way the logics LTL($\mathcal{D}$) and CTL*($\mathcal{D}$) have been already defined (we limit ourselves to a selection of typical features). Figure 2.4 contains a table with the appropriate syntactic correspondences dedicated to the readers that are fluent in description logics lingua. Apart from the gain of space, this should facilitate any comparison with temporal logics with concrete domains.

We adapt and extend several definitions introduced so far. Given a set $\mathsf{N}_\mathbf{R} = \{r, s, \ldots\}$ of *role names*, the notion of $\mathcal{D}$-decorated Kripke structure is generalised to structures of the form $(\mathcal{D}, \mathcal{W}, (\mathcal{R}_r)_{r\in\mathsf{N}_\mathbf{R}}, l, \mathfrak{v})$. Instead of having a single accessibility relation, the $\mathcal{D}$-decorated Kripke structures now admit a family of accessibility relations indexed by role names $r \in \mathsf{N}_\mathbf{R}$. A *role path* $P = r_1 \cdots r_n$ is a (possibly empty) word in $\mathsf{N}_\mathbf{R}^*$. Sometimes, role paths admit *abstract features* that are role names interpreted by deterministic relations, see e.g. [Lut01]. Herein, we do not consider such role names (except in Section 3.3).

Let us recall how concrete values and constraints between them are considered in LTL($\mathcal{D}$) and CTL*($\mathcal{D}$). An atomic constraint $R(\mathsf{X}^{i_1}x_{j_1}, \ldots, \mathsf{X}^{i_d}x_{j_d})$ holds at a position $i \in \mathbb{N}$ along the path π understood as an ω-sequence of valuations, whenever $R^\mathcal{D}(\pi(i+i_1)(x_{j_1}), \ldots, \pi(i+i_d)(x_{j_d}))$ holds. Hence, only the values for the variables $x_{j_1}, \ldots, x_{j_d}$ at the positions $i, i+1, \ldots, i+\max(i_1, \ldots, i_d)$ determine whether $R(\mathsf{X}^{i_1}x_{j_1}, \ldots, \mathsf{X}^{i_d}x_{j_d})$ holds true. Moreover, $R(\mathsf{X}^{i_1}x_{j_1}, \ldots, \mathsf{X}^{i_d}x_{j_d})$ is always evaluated along a path: in LTL($\mathcal{D}$) because the models are linear structures, and in CTL*($\mathcal{D}$) because $R(\mathsf{X}^{i_1}x_{j_1}, \ldots, \mathsf{X}^{i_d}x_{j_d})$ always occurs in the scope of a path quantifier E. In description logics with concrete domains, at least two generalisations are performed. Firstly, there are multiple accessibility relations (interpretations of role names) and it is possible to specify that a finite path is defined by taking steps from distinct relations using role paths. Secondly, values picked to satisfy an atomic constraint may be taken from a finite rooted subtree instead of a (linear) path. Both extensions are substantial and require to extend the way constraints are defined.

The set of *$\mathcal{ALC}^\ell(\mathcal{D})$-formulae* ('$\ell$' for 'linear') is defined as follows.

$$\phi ::= p \mid \mathsf{E}P\ R(\mathtt{t}_1, \ldots, \mathtt{t}_d) \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{EX}_r\phi,$$

where $p \in \mathrm{PROP}$, $\mathtt{t}_i$ is a term, P is a role path, $R \in \sigma$ is of arity d and r is a role name. We have the additional proviso that in $\mathsf{E}P\ R(\mathtt{t}_1, \ldots, \mathtt{t}_d)$, if $\mathtt{t}_i = \mathsf{X}^\alpha x$, then $\alpha \leq |P|$ with $|P|$ the length of P (possibly zero), otherwise we would not know how to interpret $\mathtt{t}_i$. The satisfaction relation is defined as follows (obvious clauses are omitted).

— $\mathcal{K}, w \models p \stackrel{\text{def}}{\Leftrightarrow} p \in l(w)$;
— $\mathcal{K}, w \models \mathsf{EX}_r\phi \stackrel{\text{def}}{\Leftrightarrow}$ there is $w' \in \mathcal{R}_r(w)$ such that $\mathcal{K}, w' \models \phi$,
— $\mathcal{K}, w \models \mathsf{E}r_1 \cdots r_n\ R(\mathtt{t}_1, \ldots, \mathtt{t}_d) \stackrel{\text{def}}{\Leftrightarrow}$ there is a finite path $w = w_0\mathcal{R}_{r_1}w_1\mathcal{R}_{r_2}\cdots\mathcal{R}_{r_n}w_n$ such that $R(\mathfrak{d}_1, \ldots, \mathfrak{d}_d)$ holds with $\mathfrak{d}_i \stackrel{\text{def}}{=} \mathfrak{v}(w_j, x)$ assuming that $\mathtt{t}_i = \mathsf{X}^j x$ for all i.

Logics of the form $\text{ALC}^\ell(\mathcal{D})$ can be found in [CT16; LOS20], and these are probably the closest variants to $\text{LTL}(\mathcal{D})/\text{CTL}^*(\mathcal{D})$. An *axiom* is an expression of the form $\phi \sqsubseteq \psi$ where ϕ, ψ are $\text{ALC}^\ell(\mathcal{D})$ formulae. A *terminological box (TBox, for short)* is a set of axioms. We say that $\mathcal{K}$ satisfies $\phi \sqsubseteq \psi$ (written $\mathcal{K} \models \phi \sqsubseteq \psi$) iff for every $w \in \mathcal{W}$, $\mathcal{K}, w \models \phi$ implies $\mathcal{K}, w \models \psi$; this generalises to TBoxes in the expected way. The *satisfiability problem w.r.t. a TBox*, written $\text{TSAT}(\text{ALC}^\ell(\mathcal{D}))$ takes as input an $\text{ALC}^\ell(\mathcal{D})$-formula ϕ and a finite TBox T, and asks whether there exist a $\mathcal{D}$-decorated Kripke structure $\mathcal{K}$ and $w \in \mathcal{W}$ such that $\mathcal{K}, w \models \phi$ and $\mathcal{K} \models T$. It is well known that decision problems for description logics are quite diverse, herein we have picked one that is a good representative.

THEOREM 2.2 ([CT16; LOS20]). $\text{TSAT}(\textit{ALC}^\ell(\mathbb{Z}; <, =, (=_n)_{n\in\mathbb{Z}}))$ *is in* EXPTIME *(with integers encoded in unary).*

Now, let us define the description logics $\text{ALC}^{\text{t}}(\mathcal{D})$, variants of $\text{ALC}^\ell(\mathcal{D})$ in which the values in constraints are extracted from tree-like structures. The set of $\textit{ALC}^{\text{t}}(\mathcal{D})$-*formulae* is defined from

$$\phi ::= p \mid R(\mathsf{E}P_1\ x_1, \ldots, \mathsf{E}P_d\ x_d) \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{EX}_r\phi,$$

where the P_j's are role paths, $R \in \sigma$ is of arity d and r is a role name. Unlike in $\text{ALC}^\ell(\mathcal{D})$, each variable within an atomic constraint comes with a (possibly distinct) role path. The satisfaction relation $\models$ is updated as follows.

— $\mathcal{K}, w \models R(\mathsf{E}P_1\ x_1, \ldots, \mathsf{E}P_d\ x_d) \stackrel{\text{def}}{\Leftrightarrow}$ for all $j \in [1, d]$, assuming $P_j = r_1 \cdots r_n$, there is a finite path $\pi = w_0^j\mathcal{R}_{r_1}w_1^j\mathcal{R}_{r_2}\cdots\mathcal{R}_{r_n}w_n^j$ with $\mathfrak{d}_j = \mathfrak{v}(w_n^j, x_j)$, and $R^{\mathcal{D}}(\mathfrak{d}_1, \ldots, \mathfrak{d}_d)$ holds.

Logics of the form $\text{ALC}^{\text{t}}(\mathcal{D})$ can be found in [Lut01; LM07; BR20] and this is probably the variant that is the most common for description logics, see also [Lut02a; Lut04a] and the original proposal for description logics with concrete domains in [BH91]. Atomic constraints of the form $R(\mathsf{A}P_1\ x_1, \ldots, \mathsf{A}P_d\ x_d)$ with universal quantifications over values instead can be found in [LM07] but are not developed herein. Similarly, the language can be extended to admit Boolean combinations in the scope of quantification over role paths as in [LOS20; BR20].

In [LM07], ω-*admissible* concrete domains $\mathcal{D}$ are introduced for which a very general decidability result is proved. ω-admissibility implies that the signature σ contains a finite set of binary relations with additional properties, namely:

(1) Satisfiability problem for $\mathcal{D}$-constraints is decidable.
(2) $\mathcal{D}$ satisfies the *patchwork property*, meaning roughly that two satisfiable finite sets of constraints agreeing on common variables admit a satisfiable union. A similar property can be found in [BC02], see condition (††) in Section 3.2.2.
(3) $\mathcal{D}$ satisfies a *compactness property*, meaning roughly that an infinite set of constraints is satisfiable iff any finite subset is satisfiable.

THEOREM 2.3 ([LM07]). *For any ω-admissible concrete domain $\mathcal{D}$,* $\text{TSAT}(\textit{ALC}^{\text{t}}(\mathcal{D}))$ *is decidable.*

By way of example, $(\mathbb{R}; <, =)$ and $(\mathbb{Q}; <, =)$ are ω-admissible (see an early work on $(\mathbb{Q}; <, =)$ in [Lut01]), whereas $(\mathbb{N}; <, =)$ is not. Besides, [BR20] provides a new definition for ω-admissibility, with conditions using model-theoretical characterisations.

2.5. A selection of other formalisms

In this section we shortly present some related formalisms. We start by giving a quick summary about classical first-order (FO) logics. One of the most widely used results is the decidability of the FO theory over $(\mathbb{N}; +, <, =, 0, 1)$, nowadays commonly known as *Presburger arithmetic* [Pre29; Haa18]. Decidability was originally established by Presburger via a quantifier elimination procedure, but several simplifications and other decision procedures were proposed afterwards [FR98; Opp78], see [Haa18] for an exhaustive overview. For concrete domains over finite strings, the undecidability of many FO theories can easily be established, including $(A^*; \cdot)$ [Qui46], $(A^*; <_{\text{seq}})$ [Kus06; KS15; HSZ17], and $(A^*; <_{\text{sbwd}})$ [Kus06]. One exception is the FO theory over $(A^*; <_{\text{pre}})$, for which the decidability is established by the famous theorem by Elgot and Rabin [ER66]. The focus of most research is on restricted fragments of FO, typically the existential FO theory [Mak77; Pla04; Kus06; KS15] or a restricted variable fragment of FO [KS15; HSZ17].

Logics of the form LTL($\mathcal{D}$) have been extended or adapted in order to design logical formalisms for the verification of database-driven systems, see e.g. [DHPV09; DHV14]. For instance, LTL-FO is an extension of LTL obtained by replacing propositional variables by quantifier-free FO formulae about tuples in the underyling database (an overview can be found in [DHV14]). Moreover, universal quantification for variables is also considered, providing similarities with the freeze operator, see e.g. [DHV14, Section 3] and Section 5. As for LTL($\mathcal{D}$), the difficulty for reasoning with LTL-FO rests on the infinite domain from which the data values are taken. Decidability results with a linearly ordered dense data domain can be found in [DHPV09]. Besides, separation logics with data, see e.g. [BBL09; KJW18] are logical formalisms involving concrete domains, but this cannot be developed further here.

Constraints also appear in constraint satisfaction problems (CSPs) [Bod20]. A CSP is a computational problem parameterized by a concrete domain, similarly to the logics defined above. Formally, a CSP over a concrete domain $\mathcal{D} = (\mathbb{D}; \sigma)$ is the problem of deciding, given a relational structure $\mathcal{A}$ over the same signature σ, whether there exists a homomorphism from $\mathcal{A}$ to $\mathcal{D}$. In terms of logic, a CSP can be seen as finite conjunction of constraints $R(x_1, \ldots, x_d)$, where x_i is a variable. The main focus of research in the area of CSP is the study of the computational complexity of CSPs. For finite-domain CSPs there has recently been a major breakthrough achieved by the confirmation [Bul17; Zhu17] of the twenty-year-old *dichotomy conjecture* by Feder and Vardi [Bul18], stating that every CSP over a *finite* domain can be solved efficiently or is NP-hard. Ongoing research attempts to generalize methods (based on universal algebra, model theory, and graph homomorphisms) from finite-domain CSP to infinite domains [Bod20].

3. AUTOMATA-BASED APPROACH

In this section, we present the automata-based approach for solving decision problems for logics with concrete domains $\mathcal{L}(\mathcal{D})$ by extending the approach followed in seminal works, for instance for MSO logic [Büc62] and for temporal logics, see e.g. [VW94; KVW00]. In short, this approach consists of reducing logical problems (satisfiability, model-checking) to automata-based decision problems while taking advantage of existing results and decision procedures from automata theory, see e.g. [VW07]. Below, we present the main steps for solving satisfiability and model-checking problems for $\mathcal{L}(\mathcal{D})$ thanks to so-called $\mathcal{D}$-automata (see Section 3.1). Actually, the mate-

rial presented below is quite orthodox in view of the automata-based approach for temporal/description/modal logics. However, solving the nonemptiness problem for $\mathcal{D}$-automata can be trickier than for automata defined on finite alphabets. Indeed, the elements of the concrete domain $\mathcal{D}$ can be much more constrained than letters from a finite alphabet. For many concrete domains, nonemptiness of $\mathcal{D}$-automata can be reduced directly to instances of similar problems for automata on finite alphabets (see [Lut01; Lut04a; DD07]). For a lot of other concrete domains, specific developments about $\mathcal{D}$-automata need to be provided (see e.g. [BG06; ST11; KW15]).

3.1. Constrained automata

In order to handle formulae from the logic LTL($\mathcal{D}$), we introduce a class of (constrained) $\mathcal{D}$-automata [Rev02] (see also [Čer94; ST11; KW15]) generalising Büchi automata accepting languages over finite alphabets, see e.g. [Tho90]. The alphabets of $\mathcal{D}$-automata are of the form $\mathbb{D}^k$ for some $k \geq 1$, so potentially infinite if $\mathcal{D}$ has an infinite domain. Transitions in $\mathcal{D}$-automata are labelled by constraints that allow us to constrain values in $\mathbb{D}^k$ at the current and the next position of the input valuation sequence.

Formally, a $\mathcal{D}$-*automaton* $\mathbb{A}$ with k variables is a structure (S, δ, I, F) such that

— S is a nonempty finite set of *control states* (also known as *locations*),
— $I \subseteq S$ is the set of *initial* states; $F \subseteq S$ is the set of *final* states,
— δ is a finite subset of $S \times \mathrm{C_k} \times \mathrm{S}$ called the *transition relation*, where $\mathrm{C_k}$ is the set of constraints over $\mathcal{D}$ and $\mathrm{T}_k^{\leq 1}$, where $\mathrm{T}_k^{\leq 1} \stackrel{\text{def}}{=} \{x_1, \ldots, x_k\} \cup \{\mathsf{X}x_1, \ldots, \mathsf{X}x_k\}$.

The language $\mathrm{L}(\mathbb{A})$ accepted by $\mathbb{A}$ is a set of sequences of valuations $\mathfrak{v}$ of the form $\{x_1, \ldots, x_k\} \to \mathbb{D}$. As usual, depending on the type of $\mathcal{D}$-automata (finite, Büchi, etc.) the length of the accepted sequences/words varies. For instance, given a Büchi $\mathcal{D}$-automaton $\mathbb{A}$ (the understanding by default herein), $\mathfrak{v}_0\mathfrak{v}_1 \cdots \in \mathrm{L}(\mathbb{A})$ iff there is an infinite run $q_0 \stackrel{C_0}{\to} q_1 \stackrel{C_1}{\to} \cdots$ such that

— for all $i \in \mathbb{N}$, $q_i \stackrel{C_i}{\to} q_{i+1} \in \delta$ and $[x_j \leftarrow \mathfrak{v}_i(x_j), \mathsf{X}x_j \leftarrow \mathfrak{v}_{i+1}(x_j)] \models C_i$.
— $q_0 \in I$ and there is some $q \in F$ that occurs infinitely often in $q_0 q_1 q_2 \cdots$.

The *nonemptiness problem for $\mathcal{D}$-automata*, written $\mathrm{NEP}(\mathcal{D})$, takes as input a $\mathcal{D}$-automaton $\mathbb{A}$ and asks whether $\mathrm{L}(\mathbb{A}) \neq \emptyset$. This is a classical problem in automata theory, apart from being strongly related to satisfiability in temporal logics [VW07].

3.2. Linear-time temporal logics LTL($\mathcal{D}$)

3.2.1. The standard translation in a nutshell. We explain how to construct from a given LTL($\mathcal{D}$)-formula ϕ a $\mathcal{D}$-automaton $\mathbb{A}_\phi$ such that $\mathrm{L}(\mathbb{A})$ corresponds to the models of ϕ. We fix a concrete domain $\mathcal{D} = (\mathbb{D}; R_1, \ldots, R_n)$ over a finite signature and an LTL($\mathcal{D}$) formula ϕ whose terms are among $\mathrm{T}_k^{\leq 1}$, for some $k \in \mathbb{N}$. More general terms, such as $\mathsf{X}^3 x$, could be handled in a similar fashion or eliminated if equality is part of $\mathcal{D}$. Similarly, assuming the finiteness of the signature σ is not a serious restriction as a given formula always contains a finite amount of predicates. Details are omitted herein and these assumptions are intended to simplify the presentation of the approach.

We write AC_k to denote the finite set of atomic constraints built over the terms in $\mathrm{T}_k^{\leq 1}$. Let $\mathrm{cl}(\phi)$ be the *closure set* of ϕ defined as the smallest set containing $\mathrm{AC}_k \cup \{\phi\}$, closed under subformulae and negation (double negations are eliminated), and if $\psi_1 \mathsf{U} \psi_2 \in \mathrm{cl}(\phi)$, then $\mathsf{X}(\psi_1 \mathsf{U} \psi_2) \in \mathrm{cl}(\phi)$. Let $\mathbb{A}_\phi = (S, \delta, I, F_1, \ldots, F_\alpha)$ be the $\mathcal{D}$-automaton with k variables defined as follows (generalised Büchi acceptance $F_1, \ldots, F_\alpha$ can be easily reduced to Büchi acceptance).

— S is the set of subsets of $\mathrm{cl}(\phi)$ that are propositionally maximally consistent, and if $Y \in S$, then for all $\psi \mathsf{U} \varphi \in \mathrm{cl}(\phi)$, we have $\psi \mathsf{U} \varphi \in Y$ iff $\varphi \in Y$ or $\psi, \mathsf{X}(\psi \mathsf{U} \varphi) \in Y$.

— $I \stackrel{\text{def}}{=} \{Y \in S \mid \phi \in Y\}$.

— For all $Y, Y' \in S$, we have $Y \xrightarrow{C} Y' \in \delta \stackrel{\text{def}}{\Leftrightarrow} C = (\bigwedge_{c \in Y \cap \mathrm{AC}_k} c) \wedge (\bigwedge_{c \in (\mathrm{AC}_k \setminus Y)} \neg c)$ and for all $\mathsf{X}\psi \in \mathrm{cl}(\phi)$, we have $\mathsf{X}\psi \in Y$ iff $\psi \in Y'$.

— Let $\{\psi_1 \mathsf{U} \varphi_1, \ldots, \psi_\alpha \mathsf{U} \varphi_\alpha\}$ be the set of until formulae in $\mathrm{cl}(\phi)$. For all $i \in [1, \alpha]$, we have $F_i \stackrel{\text{def}}{=} \{Y \in S \mid \psi_i \mathsf{U} \varphi_i \notin Y \text{ or } \varphi_i \in Y\}$.

By using standard arguments for LTL from [VW94] and the notion of accepted language for a $\mathcal{D}$-automaton, we can show the following result.

THEOREM 3.1. $\mathrm{L}(\mathbb{A}_\phi) = \{\mathfrak{v} : \mathbb{N} \times \{x_1, \ldots, x_k\} \to \mathbb{D} \mid \ \mathfrak{v}, 0 \models \phi\}$.

The construction of $\mathbb{A}_\phi$ works for all concrete domains $\mathcal{D}$; hence one can decide the satisfiability of ϕ by deciding the nonemptiness of $\mathrm{L}(\mathbb{A}_\phi)$.

One way to check nonemptiness of $\mathrm{L}(\mathbb{A}_\phi)$ is to see $\mathbb{A}_\phi$ as the product of an "LTL-component" dealing with the temporal requirements on the one side, and a "$\mathcal{D}$-component" dealing with the satisfiability of ω-sequences made of conjunctions of atomic constraints on the other side. In that context, a *symbolic* representation for $\mathrm{LTL}(\mathcal{D})$ models is essential. This is explained below with the introduction of symbolic models followed by a discussion about solving $\mathrm{NEP}(\mathcal{D})$ in general.

A *symbolic model* $\mathtt{w}$ is a map $\mathbb{N} \to A_k$ with $A_k \stackrel{\text{def}}{=} \mathcal{P}(\mathrm{AC}_k)$. Symbolic models can be seen as standard LTL models, i.e. ω-sequences over the finite alphabet A_k. Given a valuation $\mathfrak{v} : \mathbb{N} \times \{x_1, \ldots, x_k\} \to \mathbb{D}$, we write $\mathtt{symb}(\mathfrak{v}, k)$ to denote the symbolic model obtained from $\mathfrak{v}$ by setting $\mathtt{symb}(\mathfrak{v}, k)(i) = \{c \in \mathrm{AC}_k \mid \mathfrak{v}, i \models c\}$ for all $i \in \mathbb{N}$. A symbolic model $\mathtt{w} : \mathbb{N} \to A_k$ is *$\mathcal{D}$-satisfiable* whenever there is $\mathfrak{v} : \mathbb{N} \times \{x_1, \ldots, x_k\} \to \mathbb{D}$ such that $\mathtt{symb}(\mathfrak{v}, k) = \mathtt{w}$. For instance, the symbolic model $\{\mathsf{X}x < x\}^\omega$ is $(\mathbb{Q}; <)$-satisfiable, but it is not $(\mathbb{N}; <)$-satisfiable.

Symbolic models allow us to interpret ϕ symbolically as an LTL-formula over A_k. We write $\mathtt{w}, i \models_{\mathrm{LTL}} \phi$, where, typically, $\mathtt{w}, i \models_{\mathrm{LTL}} c \stackrel{\text{def}}{\Leftrightarrow} c \in \mathtt{w}(i)$, and the other connectives are interpreted as for LTL. The above example shows that the existence of a symbolic model $\mathtt{w}$ for ϕ does not guarantee the existence of an $\mathrm{LTL}(\mathcal{D})$-model for ϕ, as $\mathtt{w}$ may not be $\mathcal{D}$-satisfiable. However, if the set of all $\mathcal{D}$-satisfiable symbolic models (a set of ω-sequences over A_k) is ω-regular, i.e. it can be expressed by a Büchi automaton, then we can easily solve $\mathrm{SAT}(\mathrm{LTL}(\mathcal{D}))$ thanks to the property below.

LEMMA 3.2. *ϕ is $\mathrm{LTL}(\mathcal{D})$-satisfiable iff there is a $\mathcal{D}$-satisfiable symbolic model $\mathtt{w}$ such that $\mathtt{w}, 0 \models_{LTL} \phi$.*

So let us suppose there exists a Büchi automaton $\mathbb{B}^k_{\mathcal{D}}$ whose accepted language is the set of $\mathcal{D}$-satisfiable symbolic models, restricted to $\{x_1, \ldots, x_k\}$. By [VW94], one can compute a Büchi automaton $\mathbb{B}^{\mathrm{LTL}}_\phi$ whose accepted language is the set of symbolic models $\mathtt{w}$ such that $\mathtt{w}, 0 \models_{\mathrm{LTL}} \phi$. Then the standard product construction for $\mathbb{B}^k_{\mathcal{D}}$ and $\mathbb{B}^{\mathrm{LTL}}_\phi$ yields a Büchi automaton accepting $\mathrm{L}(\mathbb{B}^k_{\mathcal{D}}) \cap \mathrm{L}(\mathbb{B}^{\mathrm{LTL}}_\phi)$, i.e., the set of all $\mathcal{D}$-satisfiable symbolic models of ϕ. This entails the decidability of $\mathrm{SAT}(\mathrm{LTL}(\mathcal{D}))$ as soon as $\mathbb{B}^k_{\mathcal{D}}$ can be effectively computed, and a PSPACE upper bound if $\mathbb{B}^k_{\mathcal{D}}$ satisfies a few reasonable assumptions omitted herein.

3.2.2. Nonemptiness problem for $\mathcal{D}$-automata. Let us summarize recent approaches to solve $\mathrm{NEP}(\mathcal{D})$ for $\mathcal{D}$-automata. For many concrete domains $\mathcal{D}$, $\mathrm{NEP}(\mathcal{D})$ can be reduced to the nonemptiness problem for Büchi automata. Following the lines of arguments from the previous subsection, this may be the case if the class of $\mathcal{D}$-satisfiable symbolic

models is effectively ω-regular (see e.g. [BC02; LM07]), or even if it is provably not ω-regular, but $\mathcal{D}$-satisfiability of symbolic models can relatively easily be identified, see e.g. [DD07; LOS20]. Some other concrete domains seem to require specific treatments (see e.g. [ST11; KW15]). Of course, another situation occurs if $\mathrm{NEP}(\mathcal{D})$ is undecidable. For instance, for $\mathcal{D}_\dagger = (\mathbb{N}; =, 0, \mathrm{succ})$, $\mathrm{NEP}(\mathcal{D}_\dagger)$ is undecidable by a simple reduction from the halting problem for Minsky machines [Min67].

So let us start with the most favorable case, where the class of $\mathcal{D}$-satisfiable symbolic models is effectively ω-regular. We reduce the problem of deciding $\mathrm{NEP}(\mathcal{D})$ to the nonemptiness problem for Büchi automata, also using the decidability of the entailment problem for $\mathcal{D}$. Let $\mathbb{A}$ be the $\mathcal{D}$-automaton with variables in $\{x_1, \ldots, x_k\}$ under study, and, as before, let $\mathbb{B}^k_{\mathcal{D}}$ be a Büchi automaton accepting the set of $\mathcal{D}$-satisfiable symbolic models restricted to the variables in $\{x_1, \ldots, x_k\}$. Then one can define a Büchi automaton $\mathbb{A} \otimes \mathbb{B}^k_{\mathcal{D}}$ by synchronising the transitions using entailments of constraints such that $\mathrm{L}(\mathbb{A} \otimes \mathbb{B}^k_{\mathcal{D}}) = \{\mathtt{symb}(\mathfrak{v}, k) \mid \mathfrak{v} \in \mathrm{L}(\mathbb{A})\}$, the standard construction is omitted. Consequently, $\mathrm{L}(\mathbb{A} \otimes \mathbb{B}^k_{\mathcal{D}}) \neq \emptyset$ iff $\mathrm{L}(\mathbb{A}) \neq \emptyset$. Hence in this case, $\mathrm{NEP}(\mathcal{D})$ and $\mathrm{SAT}(\mathrm{LTL}(\mathcal{D}))$ admit decision procedures, and complexity characterisations are possible. Let us first review a few examples of ω-regularity that have been considered under various assumptions, see e.g. the globally consistent concrete domains in [Dec92; BC02], ω-admissible concrete domains in [LM07] (see also Section 2.4) and concrete domains satisfying the completion property in [DD07]. Here is an essential property of such concrete domains.

(††) Given a $\mathcal{D}$-satisfiable conjunction of atomic constraints C, for any set of variables Y occurring in C such that $C = C' \wedge C''$ with the variables in Y occur exactly in C', for any valuation $Y \to \mathbb{D}$ satisfying C', there is a conservative extension (over all the variables in C) that satisfies C.

This includes $(\mathbb{R}; <, =)$, $(\mathbb{Q}; <, =)$ and also many temporal and spatial domains from [Lut01; BC02; Lut04a]. For instance, let $\mathcal{D}_A = (I_{\mathbb{Q}}; (R_i)_{i \in [1,13]})$ be the concrete domain such that $I_{\mathbb{Q}}$ is the set of closed intervals $[r, r'] \subseteq \mathbb{Q}$ and $(R_i)_{i \in [1,13]}$ is the family of 13 Allen's relations [All83]. Similarly, the concrete domain RCC8 with space regions in $\mathbb{R}^2$ contains topological relations between spatial regions, see e.g. [WZ00] (generalisation to more domains $\mathbb{D}$ possible). In general, for concrete domains $\mathcal{D}$ for the above classes, $\mathtt{w} : \mathbb{N} \to A_k$ is $\mathcal{D}$-satisfiable, essentially if for all $i \in \mathbb{N}$, $\mathtt{w}(i)$ is $\mathcal{D}$-satisfiable, and $\mathtt{w}$ is *locally consistent*, that is for all $R(\mathsf{Xt}_1, \ldots, \mathsf{Xt}_d) \in \mathrm{AC}_k$, we have $R(\mathsf{Xt}_1, \ldots, \mathsf{Xt}_d) \in \mathtt{w}(i)$ iff $R(\mathtt{t}_1, \ldots, \mathtt{t}_d) \in \mathtt{w}(i+1)$.

THEOREM 3.3 ([BC02; DD07; ST11]). $\mathrm{SAT}(\mathit{LTL}(\mathbb{Q}, <, =))$, $\mathrm{SAT}(\mathit{LTL}(\mathbb{R}, <, =))$, $\mathrm{SAT}(\mathit{LTL}(\mathsf{RCC8}))$ *and* $\mathrm{SAT}(\mathit{LTL}(\mathcal{D}_A))$ *are* PSPACE*-complete.*

Strictly speaking, the PSPACE upper bound in [BC02] does not involve automata but it is possible to reformulate the results with Büchi automata, which is partially done in [DD07]. A substantial contribution in [BC02] is the design of sufficient conditions on $\mathcal{D}$ to get a general decidability result for $\mathrm{SAT}(\mathrm{LTL}(\mathcal{D}))$. For instance, the PSPACE upper bound for $\mathrm{SAT}(\mathrm{LTL}(\mathsf{RCC8}))$ in [BC02] improves the EXPSPACE bound from [WZ00].

For the less favorable case where the class of $\mathcal{D}$-satisfiable symbolic models is not effectively ω-regular – this is the case, for instance, for $(\mathbb{N}; <)$ – specific methods are developed for solving $\mathrm{NEP}(\mathcal{D})$. The concrete domains $\mathcal{D}$ in [ST11] are of the form $(\mathbb{D}; <, P_1, \ldots, P_l, =_{\mathfrak{d}_1}, \ldots, =_{\mathfrak{d}_m})$, where $(\mathbb{D}; <)$ is a linear ordering and the P_i's are unary relations. A saturation construction on $\mathcal{D}$ guarantees the existence of a so-called potential function [ST11], leading to a PSPACE upper bound for $\mathrm{NEP}(\mathcal{D})$ for many concrete domains satisfying reasonable computational properties including $(\mathbb{Q}; <)$, $(\mathbb{Z}; <)$ and $(\mathbb{N}; <)$. The PSPACE upper bound for $\mathrm{SAT}(\mathrm{LTL}(\mathbb{N}; <))$ can be obtained as a conse-

quence of [ST11, Theorems 16 & 19]. If the class of $\mathcal{D}$-satisfiable symbolic models is not necessarily ω-regular, it is also established that the $\mathcal{D}$-automata for such domains can express languages as those for automata from [BC17] that go strictly beyond Büchi automata. The need to capture the class of $\mathcal{D}$-satisfiable symbolic models that go beyond Büchi automata justifies the need for MSO extensions, introduced in [Boj04] and used in [CKL16] in order to settle the EHD-approach (see forthcoming Section 4).

Besides, in [KW15], concrete domains of the form either $\mathcal{D}_{\mathbb{Q}^*} = (\mathbb{Q}^*; <_{\text{pre}}, \leq_{\text{lex}}, =_{\mathfrak{d}_1}, \ldots, =_{\mathfrak{d}_m})$ or $\mathcal{D}_{[1,\alpha]^*} = ([1,\alpha]^*; <_{\text{pre}}, \leq_{\text{lex}}, =_{\mathfrak{d}_1}, \ldots, =_{\mathfrak{d}_m})$ for some $\alpha \geq 2$ are considered. Neither $\mathcal{D}_{\mathbb{Q}^*}$ nor $\mathcal{D}_{[1,\alpha]^*}$ falls into the classes of concrete domains designed in [BC02; LM07; DD07] (see above). The problem $\mathrm{NEP}(\mathcal{D}_{\mathbb{Q}^*})$ is shown in PSPACE [KW15, Theorem 6] by a sophisticated analysis based on an underlying well-quasi-ordering. The crux of the proof consists in showing that in case of nonemptiness, there is a specific run with a so-called noncontracting loop [KW15, Corollary 19]. It would be worth investigating the similarities with the proof method in [ST11], although the classes of concrete domains are incomparable. As a consequence of the runs analysis, the complexity of SAT(LTL($\mathcal{D}_{\mathbb{Q}^*}$)) can be characterised.

THEOREM 3.4 ([KW15]). SAT(*LTL*($\mathcal{D}_{\mathbb{Q}^*}$)) *and* SAT(*LTL*($[1,\alpha]^*$)) *($\alpha \geq 2$) are* PSPACE*-complete.*

The results about SAT(LTL($[1,\alpha]^*$)) can be reduced from those for SAT(LTL($\mathcal{D}_{\mathbb{Q}^*}$)) and for $\mathrm{NEP}(\mathcal{D}_{\mathbb{Q}^*})$, see [KW15, Theorem 4] although solving the problem with a finite alphabet is more constrained.

3.2.3. Model-checking. We write $\mathrm{MC}(\text{LTL}(\mathcal{D}))$ to denote the *model-checking problem* that takes as input a $\mathcal{D}$-automaton $\mathbb{A}$ and a formula ϕ in LTL($\mathcal{D}$) both over the set of variables $\{x_1, \ldots, x_k\}$, and asks whether there exists $\mathfrak{v} = \mathfrak{v}_0\mathfrak{v}_1 \cdots \in \mathrm{L}(\mathbb{A})$ (understood as an LTL($\mathcal{D}$) model) such that $\mathfrak{v}, 0 \models \phi$ (this is a simple option among many definitions for the model-checking problem). An instance $\mathbb{A}$, ϕ of $\mathrm{MC}(\text{LTL}(\mathcal{D}))$ can be solved by checking the nonemptiness of a product automaton made of $\mathbb{A}$ and $\mathbb{A}_\phi$ (see Section 3.2.1) that accepts exactly $\mathrm{L}(\mathbb{A}) \cap \mathrm{L}(\mathbb{A}_\phi)$. As for plain temporal logics, $\mathrm{MC}(\text{LTL}(\mathcal{D}))$ can be solved with $\mathcal{D}$-automata, leading to optimal complexity results. For instance, $\mathrm{MC}(\text{LTL}(\mathbb{N}; <))$ and $\mathrm{MC}(\text{LTL}(\mathbb{R}; <))$ are PSPACE-complete, see e.g. [BC02; ST11].

3.3. Branching-time temporal logics and description logics

In this section, we provide explanations to show how the automata-based approach can be extended to logics $\text{CTL}^*(\mathcal{D})$ and $\text{ALC}^{\text{t}}(\mathcal{D})$, whence handling logics whose models are $\mathcal{D}$-decorated Kripke structures. If the class of $\mathcal{D}$-satisfiable symbolic (linear) models is ω-regular, then the generalisation can be done smoothly, by using tree automata instead of Büchi word automata. Let us provide the key steps.

First, using similar arguments as for CTL^*, one establishes that the logic under study has the tree model property (see e.g. [Lut04a, Lemma 15], [Gas09, Lemma 3.3] and [CT16, Lemma 11]). This means that the $\mathcal{D}$-decorated Kripke structures can be restricted to (infinite) trees. Moreover, the branching degree, that is the maximal number of children per node, of these trees is only polynomial in the size of the instance. Let us use K to denote the branching degree of the trees.

The second step consists of introducing the notion of symbolic K-tree models, which generalize symbolic (word) models from Section 3.2.1 in a natural way. Recall that $A_k \stackrel{\text{def}}{=} \mathcal{P}(\mathrm{AC}_k)$. A *symbolic K-tree model* $\mathtt{t}$ is a map $[1,K]^* \to (A_k)^K$ satisfying certain local consistency conditions. Intuitively, a letter in $(A_k)^K$ encodes constraints between a node and its K children. As in the linear case, a symbolic K-tree model $\mathtt{t}$ is *$\mathcal{D}$-satisfiable* whenever there is a $\mathcal{D}$-decorated Kripke tree model $\mathfrak{v} : [1,K]^* \times \{x_1, \ldots, x_k\} \to \mathbb{D}$ that corresponds to $\mathtt{t}$. We omit technical details as they

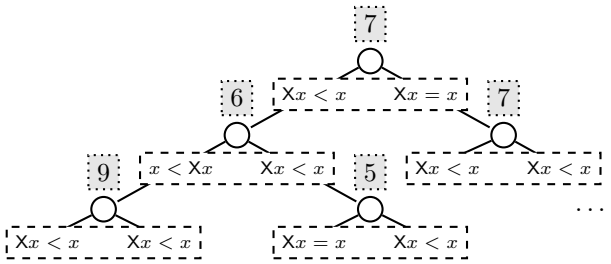


Fig. 2. Part of a tree with branching degree 2. The labels in the gray dotted boxes on top of each node represent a $\mathcal{D}$-decorated Kripke tree model for $k = 1$ and $\mathcal{D} = (\mathbb{N}; <, =)$. The labels in the dashed boxes below each node represent a corresponding symbolic 2-tree model.

are very similar to the linear case. In Figure 3.3, we show a K-tree for $k = 1$ and $K = 2$, representing a $\mathcal{D}$-decorated Kripke tree model (on top of the nodes in dotted lines) and a corresponding symbolic K-tree model (below the nodes in dashed lines), where $\mathcal{D}$ is $(\mathbb{N}; <, =)$. We remark that a *constraint graph* $\mathcal{G}_{\mathtt{t}}$ can be induced by any symbolic tree model $\mathtt{t}$ (see an early reference to such graphs in [Lut01, Section 4]) and $\mathcal{D}$-satisfiability of $\mathtt{t}$ can be reformulated as the existence of an homomorphism between $\mathcal{G}_{\mathtt{t}}$ and $\mathcal{D}$ as done with CSPs. This approach with homomorphisms is extensively developed in [Car15; CKL16; CT16] (see also Section 4).

3.3.1. CTL$^*(\mathcal{D})$ in favorable cases. Suppose that the set of $\mathcal{D}$-satisfiable symbolic K-tree models can be characterised by Büchi (or Rabin) tree automata. In this case, we can solve $\mathrm{SAT}(\mathrm{CTL}^*(\mathcal{D}))$ very similarly to the linear case. In order to extend the linear case, we write $\mathtt{t}, \mathfrak{w} \models_{\mathrm{CTL}^*} \phi$ if ϕ is interpreted on some symbolic K-tree model $\mathtt{t}$ (and a world/node $\mathfrak{w}$ in $\mathtt{t}$) symbolically as for CTL*. The following lemma is the branching counterpart of Lemma 3.2 (with similar assumptions on the input formula ϕ), see also [Lut04a, Lemma 15] and [CT16, Theorem 11].

LEMMA 3.5 ([GAS09]). *ϕ is CTL$^*(\mathcal{D})$-satisfiable iff there is a $\mathcal{D}$-satisfiable symbolic K-tree model $\mathtt{t}$ for some $K \in \mathcal{O}(|\phi|)$ such that $\mathtt{t}, \varepsilon \models_{CTL^*} \phi$.*

Since there is a Büchi (or Rabin) tree automaton $\mathbb{B}_{\mathcal{D}}^{k,K}$ whose accepted language is the set of $\mathcal{D}$-satisfiable symbolic K-tree models, and there is a tree automaton $\mathbb{B}_{\phi}^{\mathrm{CTL}^*}$ whose accepted language is the set of symbolic models $\mathtt{t}$ such that $\mathtt{t}, \varepsilon \models_{\mathrm{CTL}^*} \phi$ [KVW00], one can compute a tree automaton accepting exactly the $\mathcal{D}$-satisfiable symbolic tree models $\mathtt{t}$ such that $\mathtt{t}, \varepsilon \models_{\mathrm{CTL}^*} \phi$.

Let us name some examples of concrete domains $\mathcal{D}$ for which the set of $\mathcal{D}$-satisfiable symbolic K-tree models can be characterised by Büchi (or Rabin) tree automata. This includes all concrete domains in Section 3.2 having this property for the linear case. Here are a few new examples: $(\mathbb{R}^n; <, =)$ and $(\mathbb{Q}^n; <, =)$ for some $n \geq 1$ and the concrete domain IPC++, see e.g. [Dem06], whose domain is $\mathbb{N}$ and the relations include $x \equiv_k y$ $[\mathfrak{d}, \mathfrak{d}']$, $x \equiv_k$ $[\mathfrak{d}, \mathfrak{d}']$ (periodicity constraints), $x = y$ and $x < \mathfrak{d}$.

THEOREM 3.6 ([GAS09; GAS07]). SAT(*CTL$^*(\mathcal{D})$*) *is* 2EXPTIME*-complete for every concrete domain $\mathcal{D}$ in listed above.*

The logic CTL$^*(\mathbb{N}; <)$ does not fall in the scope of Theorem 3.6 and the decidability of $\mathrm{SAT}(\mathrm{CTL}^*(\mathbb{N}; <))$ has been first established in [CKL13] using the EHD-approach (see Section 4). Notably, significant syntactic fragments of CTL$^*(\mathbb{N}; <)$ are shown to admit a decidable satisfiability problem in [BG06] by using integral relational automata [Čer94]. However, the complexity of $\mathrm{SAT}(\mathrm{CTL}^*(\mathbb{N}; <))$ remains open despite

the substantial advances made in [BG06; CKL13] but the recent results from [LOS20] might help to close the gap.

3.3.2. The typical case of ALC$^{\mathrm{t}}$ with rational numbers. To conclude Section 3, we present the main steps of the automata-based approach for solving TSAT(ALC$^{\mathrm{t}}(\mathcal{D}_{\mathbb{Q}})$) with $\mathcal{D}_{\mathbb{Q}} = (\mathbb{Q}; <, >, \leq, \geq, =, \neq)$. Note that ALC$^{\mathrm{t}}(\mathcal{D}_{\mathbb{Q}})$ is distinct from ALC$^{\mathrm{t}}(\mathbb{Q}; <)$ as the atomic constraints in formulae occur in $R(\mathsf{E}P_1\ x_1, \ldots, \mathsf{E}P_d\ x_d)$ (see Section 2.4). We present the key results from [Lut04a] and we assume that the roles are abstract features (i.e. interpreted by deterministic binary relations) in order to stick to the assumptions from [Lut04a]. Interestingly, the automata-based approach for description logics with concrete domains has been first introduced in [Lut01] (with journal version [Lut04a]).

In order to use tree automata to solve TSAT(ALC$^{\mathrm{t}}(\mathcal{D}_{\mathbb{Q}})$), the formulae involved in the instances are in normal form (negation occurs only in front of propositional variables) and in $R(\mathsf{E}P_1\ x_1, \mathsf{E}P_2\ x_2)$ both P_1, P_2 are of length at most one and at least one P_i is empty (see [Lut04a, Lemma 11]). Such restrictions only require to express atomic constraints between a node and its children, which is exactly the way transitions in tree automata are defined.

The key step in [Lut04a] is to introduce Hintikka trees that are abstract $\mathcal{D}_{\mathbb{Q}}$-decorated Kripke structures defined as infinite K-trees with nodes labelled by finite sets of formulae , which are built from the input formula ϕ and the finite set of axioms T. This is a standard approach to design an automata-based approach so that the question to be solved becomes the existence of Hintikka trees for ϕ, T. Shortly summarizing, the nodes of such Hintikka trees are labelled by sets of formulae that are propositionally consistent, the constraints about a node and its children should be $\mathcal{D}_{\mathbb{Q}}$-satisfiable (and of course there are standard requirements related to EX_r-formulae), see e.g. [Lut04a, Section 4.3] for more details. The final step consists in showing that the satisfiability of ϕ with respect to a finite set of axioms T (assumed to be in normal form) is equivalent to the existence of a Hintikka tree (a standard property) [Lut04a, Lemma 15] and the class of Hintikka trees can be captured by a Büchi tree automaton [Lut04a, Lemma 17] leading to optimal complexity upper bounds.

THEOREM 3.7 ([LUT04A]). TSAT(*ALC*$^{\mathrm{t}}(\mathcal{D}_{\mathbb{Q}})$) *is* EXPTIME*-complete.*

Theorem 3.7 can be refined and extended by considering other concrete domains (for instance ω-admissible ones [LM07]) and by adding new features in the logical languages, see e.g. [Lut02b; Lut04b]. By contrast, adapting Theorem 3.7 to $\mathcal{D}_{\mathbb{N}} = (\mathbb{N}; <, > , \leq, \geq, =, \neq)$, possibly enriched with constant tests, is not a trivial task. For instance, TSAT(ALC$^{\ell}(\mathcal{D}_{\mathbb{N}})$) is shown decidable in [CT16], and in EXPTIME in the follow-up paper [LOS20] (with constants encoded in unary) using an automata-based approach. Moreover, it entails decidability results for description logics in the style of those from [LM07], typically a 2EXPTIME upper bound for TSAT(ALC$^{\mathrm{t}}(\mathcal{D}_{\mathbb{N}})$) [LOS20, Theorem 29]. Interesting open problems in [LOS20] include for instance the question of EXPTIME-easiness of the logic ALC$^{\ell}(\mathcal{D}_{\mathbb{N}})$ with a concrete domain augmented with constant tests (constants encoded in binary).

4. TRANSLATION INTO DECIDABLE MSO THEORIES

In this section, we present an approach initiated in [CKL13] that consists in translating decision problems about temporal logics with concrete domains into decidable MSO-like logics, instead of using $\mathcal{D}$-automata as described in Section 3.

4.1. A selection of decidable MSO-like logics

We present an overview over a selection of extensions of MSO logics that will be useful in Section 4.2. A classical result is the decidability of the satisfiability problem for MSO over infinite words, based on Büchi's famous theorem on the expressive equivalence of MSO and finite automata [Büc60]. This result has later been generalized to infinite trees by Rabin [Rab69]. There is a long history of extending MSO to gain expressiveness beyond regular languages. Here, we are interested in the logic MSO+B, introduced for infinite binary trees [Boj04] and infinite words [BC06], which extends MSO with the *bounding quantifier* B. A formula of the form $\mathsf{B}X.\varphi(X)$ expresses that there exists a finite bound on the size of the sets that satisfy $\varphi(X)$. With MSO+B one can define nonregular languages, and the question whether the satisfiability problem for MSO+B is decidable has been part of an elaborate research program [Boj04; BC06; BT12; BGMS14; Boj14; BPT16; BDG$^+$20]. It turned out that the full logic MSO+B is undecidable [BGMS14; BPT16] over infinite words (and hence infinite trees, too). Undecidability can be proved even for a weaker version of MSO+B, where quantification is allowed over sets of ultimately periodic positions [BDG$^+$20]. In contrast, a decidability result is established for the *weak* version of the logic, denoted by WMSO+B, where set quantifiers are restricted to *finite* sets [BT12]. This result is the base for proving decidability of the satisfiability problem for the logic BMW, which is the set of all Boolean combinations of MSO and WMSO+B [CKL13], interpreted over infinite trees with finite branching degree. The decidability of this logic is the key property for the approach described in the next subsection. The proofs of the decidability for WMSO+B and BMW follow Büchi's approach by proving expressive equivalence of WMSO+B with some automaton model with decidable emptiness problem. We remark that the exact computational complexity of the emptiness problem for this automaton model is open, and so is the complexity of the satisfiability problem for WMSO+B and BMW.

4.2. EHD approach

The works based on the automaton-based approach described in Section 3 leave open the decidability status for the branching-time temporal logic over the concrete domain $\mathcal{Z} = (\mathbb{Z}; <, =, (=_n)_{n\in\mathbb{Z}})$. In [CKL16], a new approach for establishing decidability results for temporal logics over concrete domains is introduced, and besides settling the question for $\mathcal{Z}$ positively (*cf.* Theorem 2.1), this approach also leads to a bunch of other new results. The gist of the method presented in [CKL16] is the establishment of two key properties of concrete domains $\mathcal{D}$ that guarantee the decidability of SAT(CTL$^*(\mathcal{D})$):

THEOREM 4.1 ([CAR15; CKL16]). *Let $\mathcal{D}$ be a concrete domain such that $\mathcal{D}$ is negation-closed, and has the property EHD(BMW). Then* SAT(*CTL*$^*(\mathcal{D})$) *is decidable.*

Let us explain the two properties mentioned in the above theorem. Suppose that σ is the relational signature that $\mathcal{D}$ is defined over. We say that $\mathcal{D}$ is *negation-closed* if the complement of each of the relations in σ is definable in positive existential FO over $\mathcal{D}$. For instance, the concrete domain $\mathcal{Z}$ is negation-closed:

— $\neg(x < y)$ if, and only if, $x = y \vee y < x$,
— $\neg(x = y)$ if, and only if, $x < y \vee x < y$, and
— $\neg(x = n)$ if, and only if, $\exists y(y = n \wedge (x < y \vee y < x))$.

The second condition, called the EHD-property, means that one can establish a characterization of all structures over the signature σ that permit a homomorphism into $\mathcal{D}$, and that characterization can be defined in a suitable logic $\mathcal{L}$. Formally, $\mathcal{D}$ has the EHD($\mathcal{L}$)-property if and only if for every finite subsignature $\tau \subseteq \sigma$ one can compute a

sentence ψ_τ in $\mathcal{L}$ such that for every countable structure $\mathcal{B}$ over signature τ

$$\exists h : \mathcal{B} \to \mathcal{D} \text{ homomorphism} \quad \Longleftrightarrow \quad \mathcal{B} \models \psi_\tau.$$

In Theorem 4.1 we use $\mathcal{L} =$ BMW, and the decidability of this logic (*cf.* Section 4.1) is essential for yielding the result. We may of course use any other decidable logic; for $\mathcal{Z}$ the choice falls naturally to an MSO logic incorporating the bounding quantifier B due to the following characterization [DD07; CKL16]: for every countable structure $\mathcal{B} = (\mathbb{B}; <^{\mathcal{B}})$ over signature $\{<\}$, there exists a homomorphism into $(\mathbb{Z}; <)$ if, and only if,

— $\mathcal{B}$ does not contain any cycles, and,
— for all $a, b \in \mathbb{B}$, there exists some $n \in \mathbb{N}$ such that the length of each path from a to b is *bounded* by n. By a path from a to b we mean a sequence $a_0, a_1, \ldots, a_k$ of elements in $\mathbb{B}$ such that $a_0 = a$, $a_k = b$ and $a_i <^{\mathcal{B}} a_{i+1}$ for all $0 \leq i < k$.

Both properties are definable in WMSO+B, and hence $\mathcal{Z}$ possesses the EHD(WMSO+B)-property. In [CKL16] it is proved that also $\mathcal{Z}$ has the EHD(BMW)-property. Together with the above mentioned fact that $\mathcal{Z}$ is negation-closed, the application of Theorem 4.1 proves Theorem 2.1, i.e., the decidability of $\mathrm{SAT}(\mathrm{CTL}^*(\mathcal{Z}))$. This illustrates the general method: given some concrete domain $\mathcal{D}$, for proving decidability of SAT(CTL$^*(\mathcal{D})$), it is sufficient to prove that $\mathcal{D}$ is negation-closed and has the EHD(BMW)-property.

A natural question is whether this approach can be applied to concrete domains other than $\mathcal{Z}$. In [CFKL17], this question is answered positively for concrete domains belonging to classes of certain tree-like structures. More precisely, the EHD-method is enhanced to *classes* Δ of concrete domains, and the satisfiability problem for a class of concrete domains is the question of deciding, for a given CTL$^*(\Delta)$-formula, whether there exists a concrete domain $\mathcal{D} \in \Delta$ such that there exists a $\mathcal{D}$-decorated model for that formula.

THEOREM 4.2 ([CFKL17]). $\mathrm{SAT}(\mathit{CTL}^*(\Delta))$ *is decidable for the following classes* Δ *of concrete domains*

(*1*) *the class of all semi-linear orders,*
(*2*) *the class of all ordinal trees,*
(*3*) *the class of all order trees of height* h*, for each* $h \in \mathbb{N}$.

Interestingly, in the same paper it is also proved that the concrete domain $\mathcal{T} = (\mathbb{N}^*; <_{\mathrm{pre}}, \perp, =)$, i.e., the set of finite words over $\mathbb{N}$ together with the prefix and the corresponding incomparability relation, does *not* have the EHD(BMW)-property. The proof is based on Ehrenfeucht-Fraïssé games which establish that BMW is not expressive enough to distinguish between structures that permit a homomorphism into $\mathcal{T}$ and those that do not. Recall that the satisfiability problems for LTL$(\mathcal{T})$ is decidable [KW15; DD16], so that we can conclude that the EHD property of a concrete domain provides a sufficient but not a necessary condition for a decidable satisfiability problem.

The EHD method has also been applied successfully to the description logic $\mathrm{ALC}^\ell(\mathcal{D})$: in [CT16], a theorem following the structure of Theorem 4.1 is proved for $\mathrm{ALC}^\ell(\mathcal{D})$, leading to the decidability of $\mathrm{TSAT}(\mathrm{ALC}^\ell(\mathcal{Z}))$.

As pointed out in Section 4.1, the exact computational complexity of the logic BMW is open, so that no upper computational complexity bounds can be inferred for the decidability results presented in this section. Moreover, the logic BMW is rather expressive and may provide little insight into the studied temporal or description logics (*c.f.* the

PSPACE-upper bound for LTL($\mathcal{Z}$) [DG08] or a recently established EXPTIME-upper bound for $\mathrm{TSAT}(\mathrm{ALC}^{\ell}(\mathcal{Z}))$ [LOS20].)

5. ADDING GLOBAL CONSTRAINTS

In this section, we present a selection of results about the addition of global constraints in LTL($\mathcal{D}$). Bibliographical references are provided for further studies and examples.

5.1. What are global constraints?

So far, constraints between values are expressed by atomic constraints $R(\mathtt{t}_1, \ldots, \mathtt{t}_d)$ or by using temporal operators. Though $R(\mathtt{t}_1, \ldots, \mathtt{t}_d)$ has obviously a local scope, global properties can be also handled thanks to the propagation of local constraints. For instance, in LTL($\mathbb{N}; =$), the formula $\mathsf{G}(x = \mathsf{X}x)$ enforces that x takes a unique value all over the linear model. Similarly, the property "the value for x is equal to some future value of y" is entailed by the satisfaction of the formula

$$\mathsf{G}(x' = \mathsf{X}x') \wedge x = x' \wedge \mathsf{XF}(x' = y),$$

that does not assume any further condition about the values for x. However, one can show that this repeating constraint cannot be expressed in LTL($\mathbb{N}; =$), without the introduction of an auxiliary x'. Note that x' plays the role of a rigid variable whose interpretation is constant and its introduction is similar to an existential quantification. In this section, we present extensions of LTL($\mathcal{D}$) in which new binders or atomic formulae allow us to constrain values at unbounded distance, unlike the atomic constraints $R(\mathtt{t}_1, \ldots, \mathtt{t}_d)$. To do so, we introduce a natural first-order extension of LTL($\mathcal{D}$) that is expressive enough to capture well-known explicit global constraints. Similar extensions can be designed for branching-time temporal logics (see e.g. [AFF17]), though omitted here by lack of space. Adding first-order quantification to linear models often leads to very expressive and undecidable formalisms, see e.g. [Krö90]. Presently, the first-order extension is mainly convenient for presentation purposes.

Probably, freeze binding is the best known mechanism to express global constraints. A formula of the form $\downarrow_{\mathtt{r}=y} \phi$ states that freezing the current value of y in the rigid variable $\mathtt{r}$, makes true ϕ ($\mathtt{r}$ possibly occurs in ϕ). Its popularity comes from its high expressive power and the possibility to use it in a restrictive way if computational properties are required. The freeze binding mechanism can be traced back to works about real-time logics to express constraints about time intervals, see e.g. [AH94], modal hybrid logics to mark states, see e.g. [Gor96; ABM01], logics for data trees, see e.g. [Fig10; JL11], or half-order modal logics in which predicate λ-abstraction permits a proper interpretation of constants in modal logics, see e.g. [Fit02; LP05]. More examples can be found in [DLN07, Section 5].

5.2. A selection of global constraints expressed in a first-order setting

Let $\mathrm{RVAR} = \{\mathtt{r}, \mathtt{s}, \mathtt{t}, \ldots\}$ be a set of *rigid variables* interpreted by elements in $\mathcal{D}$. Unlike variables in VAR whose values can vary from one state to another (understood as *flexible* variables), the variables in RVAR take the same value for all the states, like variables in classical predicate logic. Below, we define the first-order extension of LTL($\mathcal{D}$) (written LTL$^{\exists}(\mathcal{D})$) and we revisit a few notions from LTL($\mathcal{D}$). For instance, now, a *term* is either an expression of the form $\mathsf{X}^i x$ for some $i \geq 0$ (as in LTL($\mathcal{D}$)) or a rigid variable $\mathtt{r}$. The set of *LTL*$^{\exists}(\mathcal{D})$ *formulae* is defined as follows.

$$\phi ::= R(\mathtt{t}_1, \ldots, \mathtt{t}_d) \mid \phi \wedge \phi \mid \exists\, \mathtt{r}\ \phi \mid \mathsf{X}\phi \mid \phi \mathsf{U} \phi,$$

where the $\mathtt{t}_i$'s are (newly defined) terms, $R \in \sigma$ is of arity d, and $\mathtt{r} \in \mathrm{RVAR}$.

As for LTL($\mathcal{D}$), a model of LTL$^{\exists}(\mathcal{D})$ is a map $\mathfrak{v} : \mathbb{N} \times \mathrm{VAR} \to \mathbb{D}$, but the formulae are interpreted under an *environment* $\rho : \mathrm{RVAR} \to \mathbb{D}$. Given a model $\mathfrak{v} : \mathbb{N} \times \mathrm{VAR} \to \mathbb{D}$, an environment ρ and $i \in \mathbb{N}$, we write $[\![\mathtt{r}]\!]_{\mathfrak{v},\rho,i}$ to denote $\rho(\mathtt{r})$ and $[\![\mathsf{X}^j x]\!]_{\mathfrak{v},\rho,i}$ to denote $\mathfrak{v}(i+j,x)$. The satisfaction relation $\models$ for LTL$^{\exists}(\mathcal{D})$ is defined as follows (clauses similar to LTL($\mathcal{D}$) are omitted).

— $\mathfrak{v}, i \models_\rho R(\mathtt{t}_1, \ldots, \mathtt{t}_d) \stackrel{\text{def}}{\Leftrightarrow} ([\![\mathtt{t}_1]\!]_{\mathfrak{v},\rho,i}, \ldots, [\![\mathtt{t}_d]\!]_{\mathfrak{v},\rho,i}) \in R^{\mathcal{D}}$,
— $\mathfrak{v}, i \models_\rho \exists\, \mathtt{r}\ \phi \stackrel{\text{def}}{\Leftrightarrow}$ there is $\mathfrak{d} \in \mathbb{D}$ such that $\mathfrak{v}, i \models_{\rho[\mathtt{r} \mapsto \mathfrak{d}]} \phi$.

A sentence ϕ (no free occurrences of rigid variables) is *satisfiable* if there is some model $\mathfrak{v}$ such that $\mathfrak{v}, 0 \models_\rho \phi$ for some arbitrary environment ρ. The model-checking problem can be adapted similary, a well-studied instance is presented in Section 5.3.

Now, let us focus on several LTL$^{\exists}(\mathcal{D})$ fragments obtained by restricting the first-order quantification and motivated by the desire to express specific global constraints. To start with, assuming that $\mathcal{D}$ contains the equality predicate, we introduce the *freeze operator* $\downarrow$ (see e.g. [AH94; Gor96]), already met earlier, such that the formula $\downarrow_{\mathtt{r}=y} \phi$ states that freezing the value of y in the rigid variable $\mathtt{r}$ makes true the formula ϕ. For instance, $\mathsf{G}(\downarrow_{\mathtt{r}=y} \mathsf{XG}(\neg(\mathtt{r} = y)))$ expresses that y never takes twice the same value. We write LTL$^{\downarrow}(\mathcal{D})$ to denote the restriction of LTL$^{\exists}(\mathcal{D})$ such that $\exists$ occurs only in subformulae of the form $\exists\, \mathtt{r}\ (\mathtt{r} = y) \wedge \phi$ and rigid variables occur only in equalities of the form $\mathtt{r} = z$, that amounts to admit first-order quantification only to encode the freeze binder.

Unrestricted use of the freeze quantifier easily leads to undecidability (see below), which motivates the introduction of repeating constraints stating that the current value of x is equal to the value of y at a future position satisfying the formula ϕ, which can be captured by $\downarrow_{\mathtt{r}=x} \mathsf{XF}(\mathtt{r} = y \wedge \phi)$. We write LTL$^{\langle\rangle}(\mathcal{D})$ (see e.g. [DFP16]) to denote the restriction of LTL$^{\downarrow}(\mathcal{D})$ such that $\exists$ and rigid variables occur only in subformulae of the form $\exists\, \mathtt{r}\ (\mathtt{r} = x) \wedge \mathsf{XF}(\mathtt{r} = y \wedge \phi)$ (this constraint can be written $x = \langle\phi\rangle y$). Similarly, we write LTL$^{\langle\top\rangle}(\mathcal{D})$ to denote the fragment of LTL$^{\langle\rangle}(\mathcal{D})$ in which $x = \langle\phi\rangle y$ is allowed only with $\phi = \top$. So, $x = \langle\top\rangle y$ only states the repetition of a value. Other relevant fragments of LTL$^{\exists}(\mathcal{D})$ have been introduced in [Car15, Chapter 8].

5.3. The complexity of freezing or repeating

The power of global constraints on simple concrete domains. As the freeze quantifier turns out to be very expressive, in this paragraph we consider simple concrete domains such as $(\mathbb{N}; =)$ to measure its impact on computability. For instance, $\mathrm{SAT}(\mathrm{LTL}(\mathbb{N}; =))$ is PSPACE-complete. However, one rigid variable in LTL$^{\downarrow}(\mathbb{N}; =)$ leads to undecidability, see the sharp result below improving earlier works [LP05; DLN07; DL09].

THEOREM 5.1 ([FS09]). SAT(*LTL*$^{\downarrow}(\mathbb{N}; =)$) *restricted to a unique rigid variable and to the temporal operator* F *is undecidable.*

Restricting the use of the temporal operators as well as the occurrences of the freeze quantifier has been investigated to regain decidability. The *safety fragment* of LTL$^{\downarrow}(\mathbb{N}; =)$ contains the formulae with at most one rigid variable and all the occurrences of U occur under an even number of negations.

THEOREM 5.2 ([LAZ11]). *The satisfiability problem for the safety fragment of LTL*$^{\downarrow}(\mathbb{N}; =)$ *is* EXPSPACE*-complete.*

In our definition for safety LTL$^{\downarrow}(\mathbb{N}; =)$, multiple (flexible) variables are allowed and no propositional variables are present whereas in [Laz11] the safety fragment is inter-

preted on data words [Bou02] and the language admits letters from a finite alphabet. Nevertheless, EXPSPACE-hardness in Theorem 5.2 is straightforward from [Laz11] as equalities can simulate letters. Moreover, the EXPSPACE upper bound in Theorem 5.2 can be obtained by a reduction into the safety fragment from [Laz11] that increases only polynomially the number of subformulae and the finite alphabet, which is the essential complexity measure, see e.g. the proof of [Laz11, Theorem 2.5]. Safety refers to the classification of verification properties, see e.g. [Sis94]. Typically, any ω-sequence not in some given safety property has a finite prefix such that none of its extensions belong to the property.

The freeze quantifier can easily lead to undecidability but adequate fragments may regain decidability, see Theorem 5.2. In order to illustrate the diversity of interesting fragments, below, we consider another syntactic fragment leading to decidability but this time for a model-checking problem on counter machines. $\mathrm{LTL}^{\downarrow}(\mathbb{N};=)$ models with a unique variable are ω-sequences of natural numbers. Such sequences can be extracted from accepting runs for one-counter automata. A *one-counter automaton (OCA)* $\mathbb{A}$ is a structure (Q, q_I, δ, Q_F) where Q is the finite set of *locations*, $q_0 \in Q$, $\delta \subseteq Q \times \{-1, 0, +1, 0?\} \times Q$ and $Q_F \subseteq Q$. The elements in $\{-1, 0, +1, 0?\}$ are intructions (decrement, skip, increment, zero-test) and *accepting* runs $(q_0, n_0) \to (q_1, n_1) \to \cdots$ are ω-sequences of *configurations* in $(Q \times \mathbb{N})^\omega$ respecting δ (standard details are omitted) such that $(q_0, n_0) = (q_I, 0)$ and some element of Q_F occurs infinitely often. The *model-checking problem* takes as inputs an OCA $\mathbb{A}$ and a sentence $\phi \in \mathrm{LTL}^{\downarrow}(\mathbb{N};=)$ (with a single flexible variable) and asks whether there is an accepting run $(q_0, n_0) \to (q_1, n_1) \to \cdots$ such that $n_0 n_1 n_2 \cdots \models \phi$. The *flat fragment* of $\mathrm{LTL}^{\downarrow}(\mathbb{N};=)$ is made of formulae such that in any positive (resp. negative) occurrence of $\phi_1 \mathsf{U} \phi_2$, $\downarrow$ does not occur in ϕ_1 (resp. in ϕ_2). Unlike the safety fragment, U is not restricted but the occurrences of $\downarrow$ are. Though the model-checking problem over OCA is undecidable in full generality [DLS10], flatness allows us to regain decidability.

THEOREM 5.3 ([BQS19]). *Model-checking problem over OCA for flat* $\mathrm{LTL}^{\downarrow}(\mathbb{N};=)$ *is* NEXPTIME*-complete.*

OCA with *parameterised tests* are studied in [BQS19] to get this upper bound NEXPTIME. Accepting runs of OCA are data words [Bou02], and many logics have data words as models, see [Kar16, Figure 4.7] for a complete recapitulation of complexity results. It is worth noting that herein, the logics $\mathrm{LTL}(\mathcal{D})$ and extensions such as $\mathrm{LTL}^{\downarrow}(\mathcal{D})$ have no propositional variables (unlike many formalisms dedicated to data words) but this can be simulated with atomic constraints when the number of variables is not bounded and the concrete domain is not trivial.

Repeating values and linearly-ordered domains. Most known decidability results with repeating constraints of the form $x = \langle \top \rangle y$ do involve concrete domains with equality only, see e.g. [DFP16]. Adding a linear ordering may have an expensive computational cost as stated below.

THEOREM 5.4 ([CAR15]). $\mathrm{SAT}(\mathit{LTL}^{\langle \top \rangle}(\mathbb{N}; <, =))$ *restricted to repetition constraints of the form* $x = \langle \top \rangle y$ *is undecidable.*

The proof of Theorem 5.4 is by reducing the infinite accepting run problem for incrementing counter automata (incrementing errors are possible in such machines) [Car15]. By contrast, we have seen that $\mathrm{SAT}(\mathrm{LTL}(\mathbb{N}; <, =))$ is PSPACE-complete, and $\mathrm{SAT}(\mathrm{LTL}^{\langle \top \rangle}(\mathbb{N};=))$ is decidable in EXPSPACE [DFP16]. Other undecidability results for constrained LTL with repeating constraints and concrete domains with relations other than equality can be found in [Car15; Bha20] as well as the de-

sign of decidable fragments. Moreover, the works [AK10; DHLT14; Kar16] have investigated richer global constraints on data words with multiple attributes (flexible variables) that can navigate along positions with identical values for some attribute.

6. CONCLUDING REMARKS

In this document, we have presented decidability/complexity results for logics with concrete domains, with a diversity of logical formalisms (description logics, temporal logics) and a diversity of concrete domains. For solving satisfiability and model-checking, we have mainly described the features of the automata-based approach and the EHD-approach. Section 5 is also dedicated to more global constraints, in particular those that can be expressed with the freeze quantifier. The literature is too rich to expect to cover all the topics in a short survey, in particular the logics for the automatic verification of database-driven systems [DHPV09; DHV14] and the separation logics parameterised by theories [BBL09; KJW18] could not be developed further and compared with the formalisms described in Section 2.

As observed in the document, there is a recent tendency in the literature to understand the essential ingredients that allow to get nice computational properties. This is particularly true with the EHD-approach from [Car15; CKL16] that handles rich temporal logics such as CTL^* or with the model-theoretical approach from [BR20] to characterise concrete domains having nice computational properties like the ω-admissible concrete domains from [LM07]. There is a general need to understand better many simple concrete domains when embedded in logical formalisms. For instance, the decidability status of $\mathrm{SAT}(\mathrm{LTL}(\{0,1\}^*; <_{\mathrm{pre}}, <_{\mathrm{suf}}))$ is still open, as far as we know.

Consequently, it should not come as a surprise that more work is required to unify the contributions from [ST11; KW15; CT16; BR20], in particular to understand better the relationships between the automata and the MSO-like formalisms, apart from the goal to provide a uniform framework leading to optimal complexity results. Finally, for many concrete domains, $\mathrm{SAT}(\mathrm{CTL}^*(\mathcal{D}))$ has been shown decidable but with no satisfactory complexity characterisation. There is room for further improvements also here.

REFERENCES

P.A. Abdulla, M.F. Atig, Y. Chen, L. Holík, A. Rezine, P. Rümmer, and J. Stenman. String constraints for verification. In *CAV'14*, volume 8559 of *Lecture Notes in Computer Science*, pages 150–166. Springer, 2014.

C. Areces, P. Blackburn, and M. Marx. Hybrid logics: characterization, interpolation and complexity. *The Journal of Symbolic Logic*, 66(3):977–1010, 2001.

S. Abriola, D. Figueira, and S. Figueira. Logics of repeating values on data trees and branching counter systems. In *FoSSaCS'17*, volume 10203 of *Lecture Notes in Computer Science*, pages 196–212, 2017.

R. Alur, T. Feder, and T. Henzinger. The benefits of relaxing punctuality. *Journal of the Association for Computing Machinery*, 43:116–146, 1996.

R. Alur and T. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.

R. Alur and Th. Henzinger. A really temporal logic. *Journal of the Association for Computing Machinery*, 41(1):181–204, 1994.

Th. Zeume A. Kara, Th. Schwentick. Temporal logics on words with multiple data values. In *FST&TCS'10*, pages 481–492. LZI, 2010.

J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

K. Bansal, R. Brochenin, and E. Lozes. Beyond shapes: Lists with ordered data. In *FoSSaCS'09*, volume 5504 of *Lecture Notes in Computer Science*, pages 425–439. Springer, 2009.

Ph. Balbiani and J.F. Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *FroCoS'02*, volume 2309 of *Lecture Notes in Artificial Intelligence*, pages 162–173. Springer, 2002.

M. Bojańczyk and Th. Colcombet. Bounds in ω-regularity. In *LiCS'06*, pages 285–296. IEEE Computer Society, 2006.

M. Bojańczyk and Th. Colcombet. Boundedness in languages of infinite words. *Logical Methods in Computer Science*, 13(4:3):1–54, 2017.

M. Bojańczyk, L. Daviaud, B. Guillon, V. Penelle, and A.V. Sreejith. Undecidability of a weak version of MSO+U. *Logical Methods in Computer Science*, 16(1), 2020.

R. Brochenin, S. Demri, and E. Lozes. Reasoning about sequences of memory states. *Annals of Pure and Applied Logic*, 161(3):305–323, 2009.

A. Bouajjani, R. Echahed, and P. Habermehl. On the verification problem of nonregular properties for nonregular processes. In *LiCS'95*, pages 123–133, 1995.

L. Bozzelli and R. Gascon. Branching-time temporal logic extended with Presburger constraints. In *LPAR'06*, volume 4246 of *Lecture Notes in Computer Science*, pages 197–211. Springer, 2006.

M. Bojańczyk, T. Gogacz, H. Michalewski, and M. Skrzypczak. On the decidability of MSO+U on infinite trees. In *ICALP'14*, volume 8573 of *Lecture Notes in Computer Science*, pages 50–61. Springer, 2014.

F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *IJCAI'91*, pages 452–457, 1991.

A. Bhaskar. LTL with Local and Remote Data. Master's thesis, Chennai Mathematical Institute, 2020.

F. Baader, I. Horrocks, C. Lutz, and U. Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

M. Bodirsky. *Complexity of Infinite-Domain Constraint Satisfaction*. 2020. Version on Dec. 16, 2020 available on the author's web page.

M. Bojańczyk. A bounding quantifier. In *CSL'04*, volume 3210 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

M. Bojańczyk. Weak MSO+U with path quantifiers over infinite trees. In *ICALP'14*, volume 8573 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2014.

P. Bouyer. A logical characterization of data languages. *Information Processing Letters*, 84(2):75–85, 2002.

M. Bojańczyk, P. Parys, and S. Toruńczyk. The MSO+U theory of (N, $<$) is undecidable. In *STACS'16*, volume 47 of *LIPIcs*, pages 21:1–21:8. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

B. Bollig, K. Quaas, and A. Sangnier. The Complexity of Flat Freeze LTL. *Logical Methods in Computer Science*, 15(3), 2019.

F. Baader and J. Rydval. Description logics with concrete domains and general concept inclusions revisited. In *IJCAR'20*, volume 12166 of *Lecture Notes in Computer Science*, pages 413–431. Springer, 2020.

M. Bojańczyk and S. Toruńczyk. Weak MSO+U over infinite trees. In *STACS'12*, LIPIcs, pages 648–660, 2012.

J.R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, (6):66–92, 1960.

J.R. Büchi. On a decision method in restricted second-order arithmetic. In *International Congress on Logic, Method and Philosophical Science'60*, pages 1–11, 1962.

A.A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *FOCS'17*, pages 319–330. IEEE Computer Society, 2017.

A. A. Bulatov. Constraint satisfaction problems: complexity and algorithms. *ACM SIGLOG News*, 5(4):4–24, 2018.

C. Carapelle. *On the satisfiability of temporal logics with concrete domains*. PhD thesis, Leipzig University, 2015.

H. Comon and V. Cortier. Flatness is not a weakness. In *CSL'00*, volume 1862 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2000.

R. Condurache, C. Dima, Y. Oualhdaj, and N. Troquard. Rational Synthesis in the Commons with Careless and Carefeul Agents. In *AAMAS'21*, pages 368–376, 2021.

K. Čerāns. Deciding properties of integral relational automata. In *ICALP'94*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer, 1994.

C. Carapelle, S. Feng, A. Kartzow, and M. Lohrey. Satisfiability of ECTL* with local tree constraints. *Theory Computing Systems*, 61(2):689–720, 2017.

C. Carapelle, A. Kartzow, and M. Lohrey. Satisfiability of CTL* with constraints. In *CONCUR'13*, Lecture Notes in Computer Science, pages 455–463. Springer, 2013.

C. Carapelle, A. Kartzow, and M. Lohrey. Satisfiability of ECTL* with constraints. *Journal of Computer and System Sciences*, 82(5):826–855, 2016.

C. Carapelle and A.-Y. Turhan. Description Logics Reasoning w.r.t. General TBoxes is Decidable for Concrete Domains with the EHD-property. In *ECAI'16*, volume 285, pages 1440–1448. IOS Press, 2016.

S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. *Information and Computation*, 205(3):380–415, 2007.

S. Demri and M. Deters. Temporal logics on strings with prefix relation. *Journal of Logic and Computation*, 26:989–1017, 2016.

R. Dechter. From local to global consistency. *Artificial Intelligence*, pages 87–107, 1992.

S. Demri. LTL over integer periodicity constraints. *Theoretical Computer Science*, 360(1003):96–123, 2006.

S. Demri, D. Figueira, and M. Praveen. Reasoning about data repetitions with counter systems. *Logical Methods in Computer Science*, 12(3), 2016.

S. Demri and R. Gascon. Verification of qualitative $\mathbb{Z}$ constraints. *Theoretical Computer Science*, 409(1):24–40, 2008.

N. Decker, P. Habermehl, M. Leucker, and D. Thoma. Ordered navigation on multi-attributed data words. In *CONCUR'14*, volume 8704 of *Lecture Notes in Computer Science*, pages 497–511, 2014.

A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *ICDT'09*, volume 361 of *ACM International Conference Proceeding Series*, pages 252–267. ACM, 2009.

A. Deutsch, R. Hull, and V. Vianu. Automatic verification of database-centric system. *SIGMOD Record*, 43(3):5–17, 2014.

S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic*, 10(3), 2009.

S. Demri, R. Lazić, and D. Nowak. On the freeze quantifier in constraint LTL: decidability and complexity. *Information and Computation*, 205(1):2–24, 2007.

S. Demri, R. Lazić, and A. Sangnier. Model checking memoryful linear-time logics over one-counter automata. *Theoretical Computer Science*, 411(22–24):2298–2316, 2010.

L. Exibard, E. Filiot, and P.-A. Reynier. Synthesis of data word transducers. *Logical Methods in Computer Science*, 17(1), 2021.

C. Elgot and M.O. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *The Journal of Symbolic Logic*, 31(2):169–181, 1966.

F. Fages. *A Guided Tour of Artificial Intelligence Research, Volume III: Interfaces and Applications of AI*, chapter Artificial Intelligence in Biological Modeling. Springer, 2020.

D. Figueira. *Reasoning on words and trees with data*. PhD thesis, ENS Cachan, December 2010.

M. Fitting. Modal logic between propositional and first-order. *Journal of Logic and Computation*, 12(6):1017–1026, 2002.

R. Faran and O. Kupferman. LTL with arithmetic and its applications in reasoning about hierarchical systems. In *LPAR'18*, volume 57 of *EPiC Series in Computing*, pages 343–362. EasyChair, 2018.

M.J. Fischer and M.O. Rabin. Super-exponential complexity of Presburger arithmetic. In *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 122–135. Springer, 1998.

F. Fages and A. Rizk. From model-checking to temporal logic constraint solving. In *CP'09*, volume 5732 of *Lecture Notes in Computer Science*, pages 319–334. Springer, 2009.

D. Figueira and L. Segoufin. Future-looking logics on data words and trees. In *MFCS'09*, volume 5734 of *Lecture Notes in Computer Science*, pages 331–343. Springer, 2009.

R. Gascon. *Spécification et vérification de propriétés quantitatives sur des automates à contraintes*. PhD thesis, ENS de Cachan, November 2007. (in French).

R. Gascon. An automata-based approach for CTL* with constraints. *Electronic Notes in Theoretical Computer Science*, 239:193–211, 2009.

V. Goranko. Hierarchies of modal and temporal logics with references pointers. *Journal of Logic, Language, and Information*, 5:1–24, 1996.

E. Grädel. Subclasses of Presburger arithmetic and the polynomial-time hierarchy. *Theoretical Computer Science*, 56:289–301, 1988.

Ch. Haase. A survival guide to Presburger arithmetic. *SIGLOG News*, 5(3):67–82, 2018.

S. Halfon, Ph. Schnoebelen, and G. Zetzsche. Decidability, complexity, and expressiveness of first-order logic over the subword ordering. In *LiCS'17*, pages 1–12. IEEE Computer Society, 2017.

P. Hooimeijer and W. Weimer. StrSolve: solving string constraints lazily. *Automated Software Engineering*, 19(4):531–559, 2012.

M. Jurdziński and R. Lazić. Alternating automata on data trees and XPath satisfiability. *ACM Transactions on Computational Logic*, 12(3):19:1–19:21, 2011.

A. Kara. *Logics on Data Words*. PhD thesis, Universität Dortmund, 2016.

J. Katelaan, D. Jovanović, and G. Weissenbacher. A separation logic with data: Small models and automation. In *IJCAR'18*, volume 10900 of *Lecture Notes in Computer Science*, pages 455–471. Springer, 2018.

F. Kröger. On the interpretability of arithmetic in temporal logic. *Theoretical Computer Science*, 73:47–61, 1990.

P. Karandikar and Ph. Schnoebelen. Decidability in the logic of subsequences and supersequences. In *FST&TCS'15*, volume 45 of *LIPIcs*, pages 84–97. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

D. Kuske. Theories of orders on the set of words. *Theoretical Informatics and Applications*, 40:53–74, 2006.

O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the Association for Computing Machinery*, 47(2):312–360, 2000.

A. Kartzow and Th. Weidner. Model checking constraint LTL over trees. *CoRR*, abs/1504.06105, 2015.

R. Lazić. Safety alternating automata on data words. *ACM Transactions on Computational Logic*, 12(2):10:1–10:24, 2011.

C. Lutz and M. Milicić. A Tableau Algorithm for Description Logics with Concrete Domains and General Tboxes. *Journal of Automated Reasoning*, 38(1-3):227–259, 2007.

N. Labai, , M. Ortiz, and M. Simkus. An Exptime Upper Bound for $\mathcal{ALC}$ with integers. In *KR'20*, pages 425–436. Morgan Kaufman, 2020.

A. Lisitsa and I. Potapov. Temporal logic with predicate λ-abstraction. In *TIME'05*, pages 147–155. IEEE, 2005.

C. Lutz. Interval-based temporal reasoning with general TBoxes. In *IJCAI'01*, pages 89–94. Morgan-Kaufmann, 2001.

C. Lutz. Adding numbers to the SHIQ description logic: First results. In *KR'02*, pages 191–202. Morgan Kaufmann, 2002.

C. Lutz. *The Complexity of Description Logics with Concrete Domains*. PhD thesis, RWTH, Aachen, 2002.

C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*, pages 265–296. King's College Publications, 2003.

C. Lutz. Combining interval-based temporal reasoning with general TBoxes. *Artificial Intelligence*, 152(2):235–274, 2004.

C. Lutz. NEXPTIME-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004.

G.S. Makanin. The problem of solvability of equations in a free semigroup (english translation). *Mathematics of the USSR-Sbornik*, 32(2):129–198, feb 1977.

M. Minsky. *Computation, Finite and Infinite Machines*. Prentice Hall, 1967.

D. Oppen. A 2^2^2^pn upper bound on the complexity of Presburger arithmetic. *Journal of Computer and System Sciences*, 16(3):323–332, 1978.

W. Plandowski. Satisfiability of word equations with constants is in PSPACE. *Journal of the Association for Computing Machinery*, 51(3):483–496, 2004.

M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du premier congrès de mathématiciens des Pays Slaves, Warszawa*, pages 92–101, 1929.

W.V. Quine. Concatenation as a basis for arithmetic. *The Journal of Symbolic Logic*, 11(4):105–114, 1946.

M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the AMS*, (141):1–23, 1969.

P. Revesz. *Introduction to Constraint Databases*. Springer, New York, 2002.

A. Sistla and E. Clarke. The complexity of propositional linear temporal logic. *Journal of the Association for Computing Machinery*, 32(3):733–749, 1985.

K. Schild. A correspondence theory for terminological logics: preliminary report. In *IJCAI-12*, pages 466–471, 1991.

A. Sistla. Safety, Liveness and Fairness in Temporal Logic. *Formal Aspects of Computing*, 6(5):495–512, 1994.

L. Segoufin and S. Toruńczyk. Automata based verification over linearly ordered data domains. In *STACS'11*, pages 81–92, 2011.

W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B, Formal models and semantics*, pages 133–191. Elsevier, 1990.

S. Tobies. PSPACE reasoning for graded modal logics. *Journal of Logic and Computation*, 11:85–106, 2001.

M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.

M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994.

M. Vardi and Th. Wilke. Automata: from logics to algorithms. In *Logic and Automata: History and Perspectives*, number 2 in Texts in Logic and Games, pages 629–736. Amsterdam University Press, 2007.

F. Wolter and M. Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *KR'00*, pages 3–14, 2000.

D. Zhuk. A proof of CSP dichotomy conjecture. In *FOCS'17*, pages 331–342. IEEE Computer Society, 2017.