Robustness of Constraint Automata for Description Logics with Concrete Domains

Stéphane Demri
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, 91190, Gif-Sur-Yvette, France

Tianwen Gu
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, 91190, Gif-Sur-Yvette, France

Abstract

Decidability or complexity issues about the consistency problem for description logics with concrete domains have already been analysed with tableaux-based or type elimination methods. Concrete domains in ontologies are essential to consider concrete objects and predefined relations. In this work, we expose an automata-based approach leading to the optimal upper bound ExpTime, that is designed by enriching the transitions with symbolic constraints. We show that the nonemptiness problem for such automata belongs to ExpTime if the concrete domains satisfy a few simple properties. Then, we provide a reduction from the consistency problem for ontologies, yielding ExpTime-membership. Thanks to the expressivity of constraint automata, the results are extended to additional ingredients such as inverse roles, functional role names and constraint assertions, while maintaining ExpTime-membership, which illustrates the robustness of the approach.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases Description logics, concrete domains, constraint automata, complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2026.42

Related Version *Full Version*: https://arxiv.org/abs/2601.19644

Acknowledgements We thank the reviewers for their numerous suggestions that help us to improve the quality of the document, in particular to have spotted an error in the submitted version.

1 Introduction

Landmark results for description logics with concrete domains. In description logics with concrete domains, the elements of the interpretation domain are enriched with tuples of values from the concrete domain, see e.g. [4, 25, 26, 27, 6], enabling reasoning about concrete information. The first decidability result for a large class of concrete domains $\mathcal{D}$ can be traced back to [27] in which the decidability of the consistency problem with ω-admissible concrete domains is established using tableaux-based decision procedures (apart from the very first decidability result in [4, Theorem 4.3]). Though this was a genuine breakthrough in understanding $\mathcal{ALC}$ with concrete domains, a few restrictions were required: the set of binary relations is finite; $\mathcal{D}$ satisfies the patchwork property [27, Definition 3] which corresponds to the amalgamation property; and it satisfies compactness, see e.g. [8, Section 6.1] and [27, Definition 4] which corresponds to homomorphism ω-compactness. Furthermore, CD-restrictions are of the form $\mathcal{Q}\, v_1 : p_1, v_2 : p_2.\ P_1(v_1, v_2) \vee \cdots \vee P_k(v_1, v_2)$ [27, Definition 6], where $\mathcal{Q} \in \{\exists, \forall\}$ and the P_i's are binary predicates. Moreover, no ABoxes in the consistency problem are handled in [27]. Among the concrete domains captured by [27], it is worth mentioning Allen's interval algebra and RCC8, see more details in [27, Section 2].

Though the works [27, 15] mainly focus on decidability, a recent remarkable breakthrough was achieved in [13], which shows that the knowledge base consistency problem for $\mathcal{ALC}$ with ω-admissible concrete domains belongs to ExpTime, refining the decidability results; see also [3]. This complexity result nicely extends the type elimination algorithm for $\mathcal{ALC}$ that leads already to ExpTime, based on the technique using elimination of Hintikka sets established in [29], see also [12, Section 6.8], [28, Section 2.3] and [30, Section 4.2].

Our motivations. The well-known automata-based approach, see e.g. [14, 35, 36], consists of reducing logical problems to automata-based problems to take advantage of results and decision procedures from automata theory. Among the expected properties, the reduction should be conceptually simple even if showing its correctness requires some substantial work. Ideally, the complexity of the automata-based target problems should be well-understood, so that the reduction yields a tight upper bound on the complexity of the logical problem. Herein, we would like to follow an automata-based approach for reasoning in the description logics $\mathcal{ALCO}$ with concrete domains and regain the ExpTime-membership from [13] (see also the recent work [3]) based on the approach from [19, 20]. Type elimination and tableaux-based techniques are definitely worth being developed but we would like to expose an alternative approach with comprehensively developed formal methods that are equally accessible to a wide audience. Our motivations definitely rest on the use of concrete domains as the automata-based approach for description logics (without concrete domains) with automata over finite alphabets is already advocated in [2, Section 3.2], see also [5] and [7, Section 3.6.1]. This is the research agenda we wish to pursue with *constraint* automata, see also [19].

Our contributions. Our first task is to introduce a class of constraint automata parameterised by concrete domains that accept *infinite data trees* and the scope of the constraints slightly extends the scope involved in the constraint automata in [19, 20] (Section 3). The constraints involved in the transitions have a simple and natural form (constraints between siblings are allowed) that shall lead to a forthcoming smooth reduction from the consistency problem, though technical difficulties still arise. In order to get k-ExpTime-membership of the nonemptiness problem, the concrete domains satisfy (C1.1) the completion property [18, 11] (resp. (C1.1.1) the amalgamation property and (C1.1.2) homomorphism ω-compactness [13]), (C2) the arity of predicates is bounded, (C3.k) the constraint satisfaction problem for the concrete domain $\mathcal{D}$ is in k-ExpTime and (C4) equality is part of the relations. We provide a parameterised complexity analysis that is helpful to characterise then the complexity of the consistency problem for the description logic $\mathcal{ALCO}(\mathcal{D})$. More precisely, we design a reduction to the nonemptiness problem for Büchi tree automata, known to be solved in quadratic time, see e.g. [16]. Along the paper, we are particularly interested in determining which conditions are really needed for each part of our investigations.

Then, we design a reduction from the consistency problem for $\mathcal{ALCO}(\mathcal{D})$ with concrete domains $\mathcal{D}$ satisfying the above conditions into the nonemptiness problem for constraint automata parameterised by $\mathcal{D}$, leading to ExpTime-membership if (C3.1) holds, which is known to be optimal, see e.g. [13, 3]. We mix standard ingredients to translate modal/temporal/description logics into tree automata with a quite natural handling of CD-restrictions. A few complications arise to handle nominals (see e.g. [34] and the notion of guess in [32, Definition 6] about the μ-calculus with converse, universality modality and nominals), and the fact that concrete features admit partial interpretations. It is notable that we allow slightly more general classes of concrete domains than those in [13] (see Section 2.1 for a brief discussion about the differences). By way of example, we allow more general concrete domain restrictions and finiteness of the set of relations is not always required. Furthermore, we are a bit more liberal with the set of CD-restrictions, which allows us to get rid of the condition JEPD (jointly exhaustive pairwise disjoint), see e.g. [13]. Often, we introduce notions that have counterparts in [13, 3] but tailored to our approach.

Constraint automata allow us to add ingredients and to adapt smoothly the translation, for instance by adding inverse roles while giving up the nominals (as hinted possible in [13, Section 5]), functional role names and predicate assertions (see Section 2.3 and Section 5).

The extension with inverse roles, though technically challenging, perfectly illustrates the versatility of our approach. In short, we revisit the complexity results about description logics with concrete domains from [13, 3] by advocating an automata-based approach, while slightly refining some results (use of the completion property) and establishing a few new ones (inverse roles). *A forthcoming arXiv report shall contain the missing proofs.*

2 Preliminaries: Towards the Consistency Problem for Ontologies

In this section, we introduce the concrete domains $\mathcal{D}$ and the few assumptions involved in this document. Then, we define the description logics $\mathcal{ALCO}(\mathcal{D})$ parameterised by $\mathcal{D}$.

2.1 Concrete Domains Under Scrutiny

A *concrete domain* is a relational structure $\mathcal{D} = (\mathbb{D}, P_1^{\mathbb{D}}, P_2^{\mathbb{D}}, \dots)$, where $\mathbb{D}$ is a nonempty set and each $P_i^{\mathbb{D}} \subseteq \mathbb{D}^{k_i}$ interprets a predicate symbol P_i of arity k_i. Elements of $\mathbb{D}$ are written $\mathbb{d}_0, \mathbb{d}_1, \dots$. Let $\mathrm{VAR} = \{v_1, v_2, \dots\}$ be a countably infinite set of variables. A *(Boolean) constraint* is an expression following the grammar below:

$$\Theta ::= P(v_1, \dots, v_k) \mid \neg\Theta \mid \Theta \wedge \Theta \mid \Theta \vee \Theta,$$

with P a predicate symbol of arity k and $v_1, \dots, v_k \in \mathrm{VAR}$ (repetitions allowed). A *valuation* is a total function $\mathfrak{v} : \mathrm{VAR} \to \mathbb{D}$. Satisfaction of atomic formulas is defined by $\mathfrak{v} \models P(v_1, \dots, v_k) \stackrel{\text{def}}{\Leftrightarrow} (\mathfrak{v}(v_1), \dots, \mathfrak{v}(v_k)) \in P^{\mathbb{D}}$, and extended to Boolean combinations in the usual way. A *constraint system* S is a set of literals of the form either $P(v_1, \dots, v_k)$ or $\neg P(v_1, \dots, v_k)$. We write $\mathrm{VAR}(S)$ to denote the set of variables occurring in S. Given $X \subseteq \mathrm{VAR}$, we write $S|_X$ to denote the subset of S made of literals using only variables from the set X. A valuation $\mathfrak{v} : \mathrm{VAR} \to \mathbb{D}$ satisfies the system S, written $\mathfrak{v} \models S$, if it satisfies every literal in S. The *constraint satisfaction problem* for $\mathcal{D}$, denoted $\mathsf{CSP}(\mathcal{D})$, takes as an input a finite constraint system S and asks whether there is a valuation $\mathfrak{v}$ such that $\mathfrak{v} \models S$.

We assume that the concrete domains satisfy a few properties to guarantee at least decidability of the logical decision problems, and k-ExpTime-membership if possible. The concrete domain $\mathcal{D}$ satisfies the *completion property* $\stackrel{\text{def}}{\Leftrightarrow}$ for every finite constraint system S and for every valuation $\mathfrak{v} : X \to \mathbb{D}$ with $X \subseteq \mathrm{VAR}(S)$, if $\mathfrak{v}$ satisfies the restriction $S|_X$ and S is satisfiable, then there is an extension $\mathfrak{v}' : \mathrm{VAR}(S) \to \mathbb{D}$ such that $\mathfrak{v}'|_X = \mathfrak{v}$ and $\mathfrak{v}' \models S$.

A constraint system S is *complete* with respect to some set $\mathcal{P}$ of predicate symbols and some set X of variables $\stackrel{\text{def}}{\Leftrightarrow}$ no other predicate symbol or variable occurs in S and for all $P \in \mathcal{P}$ of arity k and $v_1, \dots, v_k \in X$, we have either $P(v_1, \dots, v_k) \in S$ or $\neg P(v_1, \dots, v_k) \in S$ (but not both). A concrete domain $\mathcal{D}$ satisfies the *amalgamation property* $\stackrel{\text{def}}{\Leftrightarrow}$

- either $\mathcal{D}$ has a finite set $\mathcal{P}_{\mathcal{D}}$ of predicate symbols and for all finite constraint systems S, S' such that $S|_{\mathrm{VAR}(S)\cap\mathrm{VAR}(S')} = S'|_{\mathrm{VAR}(S)\cap\mathrm{VAR}(S')}$ and $S|_{\mathrm{VAR}(S)\cap\mathrm{VAR}(S')}$ is complete w.r.t. $\mathcal{P}_{\mathcal{D}}$ and $\mathrm{VAR}(S) \cap \mathrm{VAR}(S')$, we have ($\triangle$) S and S' are satisfiable (separately) iff $S \cup S'$ is satisfiable,
- or $\mathcal{D}$ has an infinite set of predicate symbols and for all finite sets $\mathcal{P} \subseteq \mathcal{P}_{\mathcal{D}}$, for all finite constraint systems S, S' built over $\mathcal{P}$ such that $S|_{\mathrm{VAR}(S)\cap\mathrm{VAR}(S')} = S'|_{\mathrm{VAR}(S)\cap\mathrm{VAR}(S')}$ and $S|_{\mathrm{VAR}(S)\cap\mathrm{VAR}(S')}$ is complete w.r.t. $\mathcal{P}$ and $\mathrm{VAR}(S) \cap \mathrm{VAR}(S')$, we have ($\triangle$).

Similar definitions about amalgamation can be found in [27, 31, 9, 13, 3]. Note that the involved constraint systems are finite, and unlike [27, 31], we do not assume completeness of the constraint systems, see also the related notion of symbolic type in Section 3.2. Finally, $\mathcal{D}$ is *homomorphism ω-compact* (see e.g. [13]) $\stackrel{\text{def}}{\Leftrightarrow}$ if every finite $S' \subseteq S$ of a countable constraint system S is satisfiable, then S itself is satisfiable.

We are ready to define the main properties satisfied by concrete domains involved herein.
(C1) The concrete domain $\mathcal{D}$ satisfies the completion property or (it satisfies the amalgamation property and is homomorphism ω-compact). (from local to global)
(C2) There is k_0 such that all the relations in $\mathcal{D}$ have arity bounded by k_0. (bounded arity)
(C3.k) $\mathsf{CSP}(\mathcal{D})$ is in k-ExpTime ($k \geq 1$). (k-ExpTime CSP problem)
(C4) $\mathcal{D}$ contains the equality relation. (equality in the concrete domain)
The condition (C4) is a bit less general than the condition JD (jointly diagonal) from [13, 9] but simpler to check. The completion property has been already considered in the literature [10, 18, 11], sometimes under the term "global consistency", see e.g. [17, Definition 2.3] and [27]. By way of example, concrete domains satisfying the completion property include $(\mathbb{R}, <, =)$, $(\mathbb{Q}, <, =)$ and $(\mathbb{D}, =)$ for some infinite set $\mathbb{D}$. The concrete domain RCC8 with space regions in $\mathbb{R}^2$ equipped with topological relations between spatial regions [37] also satisfies the completion property, see e.g. [27, Section 2.1] and [18]. This applies also to the Allen's temporal concrete domain $\mathcal{D}_A = (I_{\mathbb{Q}}, (R_i)_{i\in[1,13]})$, where $I_{\mathbb{Q}}$ 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 [1], see also [27, Section 2]. Observe that RCC8 and Allen's interval algebra also satisfy (C2), (C3.1) and (C4), see e.g. [27, Section 2]. The completion property being one alternative in the condition (C1) is a natural condition that happens to be sufficient for our needs. However, this excludes $(\mathbb{N}, <, =)$ that is handled in [15, 23, 22, 20]. A simple property can be already established.

▶ **Lemma 1.** *If $\mathcal{D}$ satisfies the completion prop., then it satisfies the amalgamation prop.*

The converse implication is not known to hold in general; that is, whether the amalgamation property (or some variant) implies the completion property remains open. More broadly, the exact relationships between the completion property, the amalgamation property, and ω-compactness are not fully understood. By contrast with [13, 3], observe that the satisfaction of (C1), (C2), (C3.1) and (C4) allows the concrete domains to have an infinite set of relations as in $(\mathbb{Q}, <, (=_q)_{q\in\mathbb{Q}})$. The condition JEPD from [13] is not required in our framework and (C4) implies JD (jointly diagonal) from [13]. Unlike the completion property in (C1), the second disjunct of (C1) is often assumed in the literature, see e.g. [27, 31] and the ExpTime-ω-admissible concrete domains in [13, 3].

2.2 Reasoning about Ontologies with Concrete Objects

In this section, we define $\mathcal{ALCO}(\mathcal{D})$ (over the concrete domain $\mathcal{D}$) defined as the description logic $\mathcal{ALC}(\mathcal{D})$ from [13, Section 2] (see also [3]) except that the assumptions about $\mathcal{D}$ may differ and the concrete domain restrictions (a.k.a. CD-restrictions) are defined in a slightly more general way. Let $\mathsf{N_C} = \{A, B, \ldots\}$, $\mathsf{N_R} = \{r, s, \ldots\}$, $\mathsf{N_I} = \{a, b, c, \ldots\}$, and $\mathsf{N_{CF}} = \{f_0, f_1, \ldots\}$ be countable sets of *concept names*, *role names*, *individual names* and *concrete features*, respectively. Following the assumptions from [13], a *role path* p is either a concrete feature f or the concatenation $r \cdot f$ of a role name r and a concrete feature f. The set of *complex concepts* in $\mathcal{ALCO}(\mathcal{D})$ is defined as follows.

$$\begin{aligned} C \coloneqq \quad & A \mid \{a\} \mid \top \mid \bot \mid \neg A \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C \\ & \mid \exists\, v_1 : p_1, \ldots, v_k : p_k.\ \Theta(v_1, \ldots, v_k) \mid \forall\, v_1 : p_1, \ldots, v_k : p_k.\ \Theta(v_1, \ldots, v_k), \end{aligned}$$

where $A \in \mathsf{N_C}$, $a \in \mathsf{N_I}$, $r \in \mathsf{N_R}$, and $\Theta(v_1, \ldots, v_k)$ is a $\mathcal{D}$-constraint built over the variables $v_1, \ldots, v_k$ (not necessarily an atomic formula). The variables v_i bind values retrieved via the paths p_i. The expression $\{a\}$ is known as a nominal and is interpreted by a singleton, namely the one containing the interpretation of the individual name a. For instance, let

$\mathcal{D} = (\mathbb{R}, <, =)$, and suppose $\mathsf{Patient} \in \mathsf{N_C}$, $\mathsf{age} \in \mathsf{N_{CF}}$, and $\mathsf{hasBrother} \in \mathsf{N_R}$. The concept $\mathsf{Patient} \sqcap \forall\ v_1 : \mathsf{age}.\ v_1 < 18$ describes patients younger than 18, while $\exists\ v_1 : \mathsf{age},\ v_2 : \mathsf{hasBrother} \cdot \mathsf{age}.\ v_2 < v_1$ captures individuals older than at least one of their brothers.

A *general concept inclusion* (GCI) is an expression $C \sqsubseteq D$, where C, D are concepts. A *role assertion* (resp. *concept assertion*) is an expression $(a, b) : r$ (resp. $a : C$). An *ontology* is a pair $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a finite set of GCIs and $\mathcal{A}$ is a finite set of assertions. As we shall provide complexity analyses, let us briefly describe how the size of ontologies is defined. The size of $\mathcal{O}$, written $size(\mathcal{O})$, is defined as the sum of the size of $\mathcal{T}$, written $size(\mathcal{T})$, and the size of $\mathcal{A}$, written $size(\mathcal{A})$. The size of a TBox is the sum of the sizes of its GCIs, whereas the size of an ABox is defined as the sum of the sizes of its assertions. Assuming that the size of a concept C, written $size(C)$, is defined as the sum between the number of its subconcepts and the number of its subconstraints occurring in CD-restrictions, the size of a GCI is the sum of the size of its two concepts. Moreover, the size of a role assertion is defined as one, and the size of a concept assertion is defined as the size of its concept.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty (interpretation) domain $\Delta^{\mathcal{I}}$ (its elements are written $d, e, \ldots$) and an interpretation function $\cdot^{\mathcal{I}}$ such that

- $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for all $A \in \mathsf{N_C}$; $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for all $r \in \mathsf{N_R}$, $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for all $a \in \mathsf{N_I}$.
- $f^{\mathcal{I}} : \Delta^{\mathcal{I}} \to \mathbb{D}$ is a partial function, for all $f \in \mathsf{N_{CF}}$.
- Given $d \in \Delta^{\mathcal{I}}$, $p^{\mathcal{I}}(d)$ denotes either $\{f^{\mathcal{I}}(d)\}$ for $p = f$ or $\{f^{\mathcal{I}}(e) \mid (d, e) \in r^{\mathcal{I}}\}$ for $p = r \cdot f$.

  Such sets might be empty in case $f^{\mathcal{I}}$ is undefined for the involved elements of $\Delta^{\mathcal{I}}$.

The semantics below is standard, except CD-restrictions admit arbitrary constraints, see also [31, Definition 1].

$$\top^{\mathcal{I}} \stackrel{\text{def}}{=} \Delta^{\mathcal{I}},\ \bot^{\mathcal{I}} \stackrel{\text{def}}{=} \varnothing,\ \{a\}^{\mathcal{I}} \stackrel{\text{def}}{=} \{a^{\mathcal{I}}\},\ (\neg A)^{\mathcal{I}} \stackrel{\text{def}}{=} \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}, (C \sqcap D)^{\mathcal{I}} \stackrel{\text{def}}{=} C^{\mathcal{I}} \cap D^{\mathcal{I}}, (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}},$$
$$\exists r.C^{\mathcal{I}} \stackrel{\text{def}}{=} \{\, d \in \Delta^{\mathcal{I}} \mid \exists e :\ (d, e) \in r^{\mathcal{I}},\ e \in C^{\mathcal{I}} \,\}, \forall r.C^{\mathcal{I}} \stackrel{\text{def}}{=} \{\, d \in \Delta^{\mathcal{I}} \mid \forall e :\ (d, e) \in r^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}} \,\},$$
$$(\exists\ v_1 : p_1, \ldots, v_k : p_k.\ \Theta)^{\mathcal{I}} \stackrel{\text{def}}{=} \{\, d \in \Delta^{\mathcal{I}} \mid \exists\ (\mathbb{d}_1, \ldots, \mathbb{d}_k) \in \textstyle\prod p_j^{\mathcal{I}}(d) :\ [v_1 \mapsto \mathbb{d}_1, \ldots, v_k \mapsto \mathbb{d}_k] \models \Theta \,\},$$
$$(\forall\ v_1 : p_1, \ldots, v_k : p_k.\ \Theta)^{\mathcal{I}} \stackrel{\text{def}}{=} \{\, d \in \Delta^{\mathcal{I}} \mid \forall\ (\mathbb{d}_1, \ldots, \mathbb{d}_k) \in \textstyle\prod p_j^{\mathcal{I}}(d) :\ [v_1 \mapsto \mathbb{d}_1, \ldots, v_k \mapsto \mathbb{d}_k] \models \Theta \,\}.$$

An interpretation $\mathcal{I}$ *satisfies* the ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ (written $\mathcal{I} \models \mathcal{O}$) $\stackrel{\text{def}}{\Leftrightarrow}$ $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all $C \sqsubseteq D \in \mathcal{T}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all $a : C \in \mathcal{A}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for all $(a, b) : r \in \mathcal{A}$. The *consistency problem for* $\mathcal{ALCO}(\mathcal{D})$ consists in determining for some ontology whether there exists some interpretation that satisfies it. Recall that $a : C$ (resp. $(a, b) : r$) can be encoded by $\{a\} \sqsubseteq C$ (resp. $\{a\} \sqsubseteq \exists r.\{b\}$). Herein, we design an *automata-based decision procedure* for solving the fundamental consistency problem of $\mathcal{ALCO}$ with a concrete domain $\mathcal{D}$ satisfying the conditions (C1), (C2), (C3.1), (C4), and *get* ExpTime*-membership*. The consistency problem for $\mathcal{ALC}$ (without concrete domains) can take advantage of the *finite model property*, see e.g. [6, Section 4.2], but this property does not hold for $\mathcal{ALCO}(\mathcal{D})$. For instance, within $\mathcal{ALCO}(\mathbb{Q}, <)$, the GCI $(\top \sqsubseteq \exists\ v_1 : f_1, v_2 : r \cdot f_1.\ v_1 < v_2)$ can be satisfied only by interpretations with infinite domains $\Delta^{\mathcal{I}}$. As usual, we write $\mathcal{ALC}(\mathcal{D})$ to denote $\mathcal{ALCO}(\mathcal{D})$ without nominals. We find it natural to handle nominals in the language of complex concepts as ontologies admit ABoxes (see also [3]). However, there is a price to pay. It is challenging to handle nominals with an automata-based approach as only one node of the accepted trees should correspond to the interpretation of each nominal. Furthermore, infinite interpretations (see the example above for $\mathcal{ALCO}(\mathbb{Q}, <)$) and satisfying $\top \sqsubseteq \exists r.\{a\}$ lead to interpretations such that $a^{\mathcal{I}}$ has an *infinite amount* of incoming r-edges.

Given a set X of individual names, an interpretation $\mathcal{I}$ satisfies the unique name assumption (UNA) w.r.t. X iff for all distinct individual names $a \neq a' \in X$, we have $a^{\mathcal{I}} \neq a'^{\mathcal{I}}$. Below, we state a standard result that allows us to deal with interpretations satisfying (UNA) only, as we are interested in complexity classes equal or above ExpTime. Indeed, suppose

that $\mathcal{I} \models \mathcal{O}$ and $\equiv$ be the equivalence relation on $\{a_0, \ldots, a_{\eta-1}\}$ (individual names in $\mathcal{O}$) such that $a \equiv b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$. We write $[a]$ to denote the equivalence class of a. We can build a new ontology $\mathcal{O}_{\equiv} = (\mathcal{T}_{\equiv}, \mathcal{A}_{\equiv})$ as the quotient of $\mathcal{O}$ by the equivalence relation $\equiv$: in each concept, the nominal $\{a\}$ is replaced by $\{[a]\}$ and each individual name a in assertions is replaced by $[a]$. One can show that $\mathcal{O}$ is consistent iff there is some equivalence relation $\equiv$ on $\{a_0, \ldots, a_{\eta-1}\}$ such that $\mathcal{O}_{\equiv}$ can be satisfied by an interpretation satisfying (UNA) w.r.t. $\{[a_j] \mid j \in [0, \eta-1]\}$. The equivalence classes $[a_j]$ are viewed as new individual names.

2.3 Beyond $\mathcal{ALCO}$ with Concrete Domains

We briefly present several ingredients that can be found in description logics, see e.g. [6]. Section 5 is dedicated to showing how the approach developed in Section 4 can be adapted. We write $\mathcal{ALCI}(\mathcal{D})$ to denote the extension of $\mathcal{ALC}(\mathcal{D})$ (without nominals) with inverse roles. The roles are either r or r^- where $r \in \mathsf{N}_{\mathbf{R}}$ and $(r^-)^{\mathcal{I}}$ is equal to the converse of $r^{\mathcal{I}}$. Herein, inverse roles can be found in restrictions as in $\exists r^-.C$ and in CD-restrictions as in $\forall\ v_1 : f_1, v_2 : r^- f_2.\ \Theta(v_1, v_2)$. We write $\mathcal{ALCOF}(\mathcal{D})$ to denote the extension of $\mathcal{ALCO}(\mathcal{D})$ with functional role names, see e.g. [24]. We introduce the set $\mathsf{N}_{\mathbf{FR}}$ of functional role names (a.k.a. abstract features) and the roles in $\mathcal{ALCOF}(\mathcal{D})$ are the role names from $\mathsf{N}_{\mathbf{R}} \cup \mathsf{N}_{\mathbf{FR}}$. If $r \in \mathsf{N}_{\mathbf{FR}}$ and $\mathcal{I}$ is an interpretation of $\mathcal{ALCOF}(\mathcal{D})$, then $r^{\mathcal{I}}$ is weakly functional. Functional role names can be used anywhere. We may also enrich the set of assertions by allowing in ontologies, expressions $\Theta(f_1(a_1), \ldots, f_k(a_k))$ stating constraints between the concrete feature values of $a_1, \ldots, a_k$ (predicate assertions are usually restricted to atomic constraints).

3 Tree Global Constraint Automata

In this section, we introduce a class of tree constraint automata parameterised by concrete domains $\mathcal{D}$ that accept infinite data trees in which every node is labelled by a finite tuple of data values from $\mathbb{D}$. This is slightly more general than the automata models from [19], not only because the definition does not assume a fixed concrete domain but more importantly because constraints between siblings are allowed unlike what is done in [19, 20].

3.1 Definitions

Unless otherwise stated, in the rest of this section, we assume that the concrete domain $\mathcal{D}$ satisfies the conditions (C1) and (C3.k) for some $k \geq 1$ (meaning that (C2) and (C4) are not needed). We introduce the class of tree global constraint automata that accept sets of labelled trees of the form $\mathbb{t} : [0, \mathtt{d}-1]^* \to \mathbf{\Sigma} \times \mathbb{D}^{\beta}$ for some finite alphabet $\mathbf{\Sigma}$, for some $\mathtt{d} \geq 1$ and $\beta \geq 0$. The transition relation of such automata expresses constraints between the β values at a node and the values at its children. The automaton is equipped with a Büchi acceptance condition. Formally, a *tree global constraint automaton* (in short, TGCA) over $\mathcal{D}$ is a tuple $\mathbb{A} = (Q, \mathbf{\Sigma}, \mathtt{d}, \beta, Q_{in}, \delta, F)$, where:

- Q is a finite set of locations, $Q_{in} \subseteq Q$ are initial locations, $F \subseteq Q$ are accepting locations, $\mathbf{\Sigma}$ is a finite alphabet;
- $\mathtt{d} \geq 1$ is the branching degree, and $\beta \in \mathbb{N}$ the number of registers (a.k.a. variables);
- $\delta \subseteq Q \times \mathbf{\Sigma} \times \mathrm{Cons}(\beta, \mathtt{d}) \times Q^{\mathtt{d}}$ is the transition relation where $\mathrm{Cons}(\beta, \mathtt{d})$ denotes the set of $\mathcal{D}$-constraints built over the distinguished registers in $\mathrm{REG}(\beta, \mathtt{d}) \stackrel{\text{def}}{=} \{x_1, \ldots, x_\beta\} \cup \{x_j^i \mid 1 \leq j \leq \beta,\ 0 \leq i < \mathtt{d}\}$. The expression x_j^i refers to the jth register of the ith child. Arbitrary registers in $\mathrm{REG}(\beta, \mathtt{d})$ are also referred to by $y_0, y_1, y_2, \ldots$.

Runs. Let $\mathbb{t} : [0, \mathtt{d}-1]^* \to \mathbf{\Sigma} \times \mathbb{D}^\beta$ be a data tree, with $\mathbb{t}(n) = (\mathtt{a}_n, \vec{z}_n)$ at each node n. A *run* of $\mathbb{A}$ on $\mathbb{t}$ is a mapping $\rho : [0, \mathtt{d}-1]^* \to \delta$ that maps each node n to a transition $(q_n, \mathtt{a}_n, \Theta_n, q_{n\cdot 0}, \ldots, q_{n\cdot(\mathtt{d}-1)})$ such that:

- **(i)** The source location of $\rho(n \cdot i)$ is $q_{n\cdot i}$ for all $0 \leq i < \mathtt{d}$.
- **(ii)** Let $\mathfrak{v}_n : \mathrm{REG}(\beta, \mathtt{d}) \to \mathbb{D}$ be the valuation defined by $\mathfrak{v}_n(x_j) = \vec{z}_n[j]$ and $\mathfrak{v}_n(x_j^i) = \vec{z}_{n\cdot i}[j]$, for all $1 \leq j \leq \beta,\ 0 \leq i < \mathtt{d}$. Then $\mathfrak{v}_n \models \Theta_n$.
- **(iii)** The source location of $\rho(\varepsilon)$ is in Q_{in}.

Given a path $\pi = j_1 \cdot j_2 \ldots$ in ρ starting from ε, we define $\inf(\rho, \pi)$ to be the set of locations that appear infinitely often as the source locations of the transitions in $\rho(\varepsilon)\rho(j_1)\rho(j_2)\ldots$ A run ρ is *accepting* if all paths π in ρ starting from ε, we have $\inf(\rho, \pi) \cap F \neq \emptyset$. We write $\mathrm{L}(\mathbb{A})$ to denote the set of data trees $\mathbb{t}$ for which there exists some accepting run of $\mathbb{A}$ on $\mathbb{t}$. The *nonemptiness problem for TGCA*, written NE(TGCA), asks whether $\mathrm{L}(\mathbb{A}) \neq \emptyset$ for some TGCA $\mathbb{A}$. Herein, we use Büchi acceptance conditions but such conditions play no essential role in our investigations because we shall always assume that $F = Q$ (as in looping tree automata, see e.g. [7, Section 3.6.1]).

3.2 Reduction to Nonemptiness for Büchi Tree Automata

To analyse the complexity of NE(TGCA), we construct a reduction to the nonemptiness problem for Büchi tree automata (BTA), known to be in PTIME [35]. Büchi tree automata are TGCA with $\beta = 0$ and no $\mathcal{D}$-constraints. Given $\mathbb{A} = (Q, \mathbf{\Sigma}, \mathtt{d}, \beta, Q_{in}, \delta, F)$, we write $\mathcal{P}_\mathbb{A} = \{P_1, \ldots, P_m\}$ to denote either the full set of predicate symbols of $\mathcal{D}$ if finite, or the *finite* set of predicate symbols from $\mathcal{D}$ occurring in $\mathbb{A}$.

Symbolic types. We abstract concrete values by *symbolic types*, following the standard notion of types in first-order languages, see e.g. [30, Section 3.1] and [33]. These are sets of literals that capture all atomic constraints relevant to the current node and its children. A forthcoming notion of projection compatibility ensures the coherence between parent and child types across the tree. Let $\mathsf{Atoms}(\mathbb{A})$ be the set of atomic formulae of the form $P(y_1, \ldots, y_k)$ for some $y_1, \ldots, y_k \in \mathrm{REG}(\beta, \mathtt{d})$ and $P \in \mathcal{P}_\mathbb{A}$. Similarly, let $\mathsf{Literals}(\mathbb{A})$ be the set of all literals over $\mathsf{Atoms}(\mathbb{A})$, i.e., atomic formulae or their negations. A *symbolic type* τ is a set $\tau \subseteq \mathsf{Literals}(\mathbb{A})$ s.t. for every $\Theta \in \mathsf{Atoms}(\mathbb{A})$, either $\Theta \in \tau$ or $\neg\Theta \in \tau$, but not both. Let $\mathsf{STypes}(\mathbb{A})$ denote the set of all symbolic types. Symbolic types can be understood as constraints made of conjunctions of its elements built over the registers in $\mathrm{REG}(\beta, \mathtt{d})$. A symbolic type τ is *satisfiable* if there is $\mathfrak{v} : \mathrm{REG}(\beta, \mathtt{d}) \to \mathbb{D}$ such that $\mathfrak{v} \models \tau$. Let $\mathsf{SatSTypes}(\mathbb{A})$ denote the set of all satisfiable symbolic types. The properties of satisfiable types that we mainly use are stated in Lemma 2 where $\mathcal{D} \models \tau(\vec{z}, \vec{z}_0, \ldots, \vec{z}_{\mathtt{d}-1})$ means that $\mathfrak{v} \models \bigwedge_{\Theta \in \tau} \Theta$ for the valuation $\mathfrak{v}$ such that $\mathfrak{v}(x_j) = \vec{z}[j]$ and $\mathfrak{v}(x_j^i) = \vec{z}_i[j]$, for all $1 \leq j \leq \beta,\ 0 \leq i < \mathtt{d}$.

▶ **Lemma 2** (Three properties about types). *(I) Let $\vec{z}, \vec{z}_0, \ldots, \vec{z}_{\mathtt{d}-1} \in \mathbb{D}^\beta$. There is a unique satisfiable type $\tau \in \mathsf{SatSTypes}(\mathbb{A})$ such that $\mathcal{D} \models \tau(\vec{z}, \vec{z}_0, \ldots, \vec{z}_{\mathtt{d}-1})$. (II) For every constraint Θ built over the predicate symbols in $\mathcal{P}_\mathbb{A}$ and the registers $\mathrm{REG}(\beta, \mathtt{d})$, there is a disjunction $\tau_1 \vee \cdots \vee \tau_\gamma$ logically equivalent to Θ and each τ_i belongs to $\mathsf{SatSTypes}(\mathbb{A})$ (empty disjunction stands for $\bot$). (III) For all distinct types $\tau \neq \tau' \in \mathsf{SatSTypes}(\mathbb{A})$, $\tau \wedge \tau'$ is not satisfiable.*

The proof of Lemma 2 is by an easy verification. We write $\tau \models \Theta$ to denote that for all valuations $\mathfrak{v}$, if $\mathfrak{v} \models \tau$, then $\mathfrak{v} \models \Theta$. If $\tau \in \mathsf{SatSTypes}(\mathbb{A})$ and Θ satisfies the properties of Lemma 2(II), then $\tau \models \Theta$ can be checked in polynomial time. Let $\tau, \tau_0, \ldots, \tau_{\mathtt{d}-1} \in \mathsf{SatSTypes}(\mathbb{A})$. We say that τ is *projection-compatible* with $(\tau_0, \ldots, \tau_{\mathtt{d}-1})$ $\overset{\text{def}}{\Leftrightarrow}$ for each $i < \mathtt{d}$ and every predicate symbol $P \in \mathcal{P}_\mathbb{A}$ of arity k, and for all tuples $(j_1, \ldots, j_k) \in [1, \beta]^k$, we have:

$P(x^i_{j_1}, \ldots, x^i_{j_k}) \in \tau$ iff $P(x_{j_1}, \ldots, x_{j_k}) \in \tau_i$. We recall that in a type τ, registers associated to the values of the children nodes are also present. Hence, projection-compatibility simply reflects the fact that the constraints for such registers in τ must be compatible with the constraints of the registers in the children types.

Construction of Büchi tree automata. We now define the Büchi tree automaton $\mathbb{B} = (Q', \mathbf{\Sigma}, \delta', \mathtt{d}, Q'_{\mathsf{in}}, F')$ that simulates the TGCA $\mathbb{A}$ using symbolic types. The basic idea consists in abstracting data values in $\mathbb{D}$ by satisfiable types from $\mathsf{SatSTypes}(\mathbb{A})$. Moreover, the type at a node needs to be compatible with the children's types since a type expresses constraints between the values at a node and those at its children. This is where projection compatibility plays a crucial role. In this way, an infinite tree accepted by the Büchi tree automaton can be viewed as an infinite constraint system (each node has its own variables coming from β local registers) for which satisfiability is required. Generally, additional conditions are needed to guarantee satisfiability, see e.g. [23, 22, 19]. In the case of concrete domains satisfying the condition (C1), importantly, this is all we need as shown below. First, let us define the Büchi tree automaton $\mathbb{B}$ with the remarkably simple construction below.

- $Q' \stackrel{\text{def}}{=} Q \times \mathsf{SatSTypes}(\mathbb{A})$, $Q'_{in} \stackrel{\text{def}}{=} Q_{in} \times \mathsf{SatSTypes}(\mathbb{A})$, $F' \stackrel{\text{def}}{=} F \times \mathsf{SatSTypes}(\mathbb{A})$,
- δ' contains the transition $((q,\tau), \mathtt{a}, (q_0,\tau_0), \ldots, (q_{\mathtt{d}-1}, \tau_{\mathtt{d}-1}))$ if (a) $(q, \mathtt{a}, \Theta, q_0, \ldots, q_{\mathtt{d}-1}) \in \delta$, (b) $\tau \models \Theta$ and (c) τ is projection-compatible with $(\tau_0, \ldots, \tau_{\mathtt{d}-1})$.

To build effectively $\mathbb{B}$ from $\mathbb{A}$, we only need the decidability of $\mathsf{CSP}(\mathcal{D})$ to determine the satisfiable symbolic types. We can establish that this construction preserves nonemptiness. Again, apart from the decidability of $\mathsf{CSP}(\mathcal{D})$, only the condition (C1) is required.

▶ **Lemma 3** (Soundness). $\mathrm{L}(\mathbb{A}) \neq \emptyset$ *implies* $\mathrm{L}(\mathbb{B}) \neq \emptyset$.

The proof does not require much about the concrete domain $\mathcal{D}$ since it mainly rests on the construction of satisfiable types from tuples made of data values in $\mathbb{D}$.

▶ **Lemma 4** (Completeness). $\mathrm{L}(\mathbb{B}) \neq \emptyset$ *implies* $\mathrm{L}(\mathbb{A}) \neq \emptyset$.

In its proof, essential for our investigations, each disjunct of (C1) leads to a way to construct the valuation $\mathfrak{v}$ on which the data values for the data trees accepted by $\mathbb{A}$ are defined. If $\mathcal{D}$ satisfies the completion property, then the valuation $\mathfrak{v}$ is built incrementally while visiting $[0, \mathtt{d}-1]^*$ with a breadth-first search. By contrast, assuming the second disjunct of (C1) allows us to define an infinite chain of constraint systems (thanks to the amalgamation property) and then to obtain the existence of $\mathfrak{v}$ as $\mathcal{D}$ is homomorphism ω-compact. Today, it is unclear to us what are the exact relationships between these two options; it is not excluded that model-theoretical developments from [31, Chapter 4] could be helpful but we were unable to draw conclusive arguments.

It remains to perform a parameterised analysis of the complexity of the nonemptiness problem for TGCA. We warn the reader that the analysis is rather standard but this needs to be performed with care. Assuming that $\mathsf{CSP}(\mathcal{D})$ can be solved in time $\mathcal{O}(2^{\mathtt{p}_1(n)-n})$ and the nonemptiness problem for Büchi tree automata can be solved in time $\mathcal{O}(\mathtt{p}_2(n)-n)$ for some polynomials $\mathtt{p}_1$ and $\mathtt{p}_2$, the nonemptiness problem for TGCA can be solved in time

$$\mathcal{O}\left(2^{\mathtt{p}_1(m\times(\beta(\mathtt{d}+1))^{k_0})} + \mathtt{p}_2(|Q|^{\mathtt{d}+1} \times |\mathbf{\Sigma}| \times 2^{(\mathtt{d}+1)\times m\times(\beta(\mathtt{d}+1))^{k_0}})|\delta|\left(\mathtt{MCS}(\mathbb{A}) + \mathtt{d}\times m\times(\beta(\mathtt{d}+1))^{k_0}\right)\right),$$

where k_0 is the maximal arity of predicates in $\mathbb{A}$ and $\mathtt{MCS}(\mathbb{A})$ is the maximal size of constraints occurring in $\mathbb{A}$. The use of the polynomials $(\mathtt{p}_i(n)-n)$'s is convenient for the above expression.

▶ **Theorem 5.** *Let $\mathcal{D}$ be a concrete domain satisfying (C1), (C2) and (C3.1). The nonemptiness problem for TGCA over $\mathcal{D}$ is in* ExpTime.

As a slight digression, ExpTime-hardness already holds with the concrete domain $(\mathbb{D}, =)$ with infinite domain $\mathbb{D}$ as a consequence of [20, Appendix B.1]. If we assume (C3.k) for some $k \geq 2$ instead of (C3.1) in Theorem 5, we get k-ExpTime-membership.

4 Constraint Automata for Checking Consistency

Below, we reduce the consistency problem for $\mathcal{ALCO}(\mathcal{D})$ to the nonemptiness problem for TGCA over $\mathcal{D}$. Correctness of the reduction only requires that the concrete domain $\mathcal{D}$ satisfies the condition (C4). As shown in Section 3, the translation from TGCA to BTA requires the satisfaction of (C1) and $\mathsf{CSP}(\mathcal{D})$ is decidable. In order to get ExpTime-membership of the consistency problem for $\mathcal{ALCO}(\mathcal{D})$, the condition (C2) and (C3.1) are further assumed.

4.1 Preliminaries

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, we shall build a TGCA $\mathbb{A} = (Q, \mathbf{\Sigma}, \mathtt{d}, \beta, Q_{in}, \delta, F)$ over $\mathcal{D}$, such that $\mathcal{O}$ is consistent iff $\mathrm{L}(\mathbb{A}) \neq \emptyset$. Below, we introduce the notions of concept types (similar to Hintikka sets), and (local/global/contextual) abstractions that are finite pieces of information about elements in some interpretation domain $\Delta^{\mathcal{I}}$. The construction of $\mathbb{A}$ is finalised in Section 4.2 and the correctness is postponed to Section 4.3.

Normalisation. A concept is in *negation normal form* (NNF) if the negation occurs only in front of concept names or nominals. Every concept is logically equivalent to a concept in NNF. In particular, $\neg(\mathcal{Q}\ v_1 : p_1, \ldots, v_k : p_k.\ \Theta) \equiv \overline{\mathcal{Q}}\ v_1 : p_1, \ldots, v_k : p_k.\ \neg\Theta$, where $\mathcal{Q} \in \{\exists, \forall\}$, $\overline{\forall} = \exists$ and $\overline{\exists} = \forall$. This is valid since the constraints in CD-restrictions are *closed under negations*. Also, every GCI $C \sqsubseteq D$ is equivalent to $\top \sqsubseteq \neg C \sqcup D$. Hence, we assume a fixed ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ such that all the concepts are in NNF and the GCIs are of the form $\top \sqsubseteq C$. This normalisation is standard and harmless for complexity. We write $\mathsf{Sub}(\mathcal{O})$ to denote the set of subconcepts occurring in the ontology $\mathcal{O}$ augmented with the concepts $\{a\}$ and $\neg\{a\}$ for all individual names a occurring in $\mathcal{O}$ in assertions and nominals (details omitted). We introduce the following sets: $\mathsf{N_I}(\mathcal{O}) \stackrel{\text{def}}{=} \{a_0, \ldots, a_{\eta-1}\}$ is the set of individual names occurring in assertions/nominals in $\mathcal{O}$, $\mathsf{N_{CF}}(\mathcal{O}) \stackrel{\text{def}}{=} \{f_1, \ldots, f_\alpha\}$ is the set of concrete features in $\mathsf{Sub}(\mathcal{O})$ and $\mathsf{N_R}(\mathcal{O}) \stackrel{\text{def}}{=} \{r_1, \ldots, r_\kappa\}$ the set of role names in $\mathsf{Sub}(\mathcal{O})$.

Abstractions. Consistency of the ontology $\mathcal{O}$ asks for the existence of an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{O}$. In the automata-based approach, this reduces to the existence of an infinite data tree $\mathbb{t}$ such that $\mathbb{t} \in \mathrm{L}(\mathbb{A})$. Such a witness tree $\mathbb{t}$ represents an interpretation $\mathcal{I}$ (nodes in $[0, \mathtt{d}-1]^*$ roughly correspond to elements of $\Delta^{\mathcal{I}}$), so $\mathrm{L}(\mathbb{A})$ is designed to accept such representations. An accepting run on $\mathbb{t}$ thus ensures that $\mathbb{t}$, viewed as an $\mathcal{ALCO}(\mathcal{D})$ interpretation, satisfies $\mathcal{O}$. In an accepting run ρ on $\mathbb{t}$, each node n is labelled by a transition $(q, \mathtt{a}, \Theta, q_0, \ldots, q_{\mathtt{d}-1})$, where the location q contains some *finite symbolic abstraction* about the node n viewed as an element from the interpretation domain. This is where *local, global and contextual abstractions* enter into play.

Given an element $d \in \Delta^{\mathcal{I}}$ for some interpretation $\mathcal{I}$, the *local abstraction* of the element d in $\mathcal{I}$, written $\mathit{locabs}(d, \mathcal{I})$, is a triple (T, sl, act) defined as follows.

- T is the *concept type* and $T \stackrel{\text{def}}{=} \{C \in \mathsf{Sub}(\mathcal{O}) \mid d \in C^{\mathcal{I}}\}$,
- act is the *activity vector* in $\{\bot, \top\}^\alpha$ s.t. $act[i] = \top \stackrel{\text{def}}{\Leftrightarrow} f_i^{\mathcal{I}}(d)$ is defined, for all $i \in [1, \alpha]$,
- sl is the set of *symbolic links* in $\mathsf{N_R}(\mathcal{O}) \times \mathsf{N_I}(\mathcal{O})$ such that $(r, a) \in sl \stackrel{\text{def}}{\Leftrightarrow} (d, a^{\mathcal{I}}) \in r^{\mathcal{I}}$.

It is not immediate that this notion of local abstraction suffices, but we shall show that it does. Of course, in the forthcoming TGCA $\mathbb{A}$, we also make use of the registers to encode the concrete features from interpretations of $\mathcal{ALCO}(\mathcal{D})$. By contrast, the *global abstractions* act as persistent memories relevant for the whole interpretation. Unsurprisingly, they assign to each individual name in $\mathsf{N_I}(\mathcal{O})$ a local abstraction; due to a technical reason, we can restrict these to just the concept type and activity vector. Since runs of the automaton $\mathbb{A}$ cannot revisit nodes, the global abstraction enables to recover the local abstraction of the individual names from $\mathsf{N_I}(\mathcal{O})$ from any node. We use a standard notion of *types* to represent locally consistent sets of subconcepts, similarly to Hintikka sets to prove completeness of propositional logic for tableaux-style calculi, see also [6, Section 5.1.2]. The set of *concept types*, denoted $\mathsf{CTypes}(\mathcal{O})$, consists of all subsets $T \subseteq \mathsf{Sub}(\mathcal{O})$ such that

1. $\bot \notin T$; for all concept names $A \in \mathsf{CTypes}(\mathcal{O})$, if $A \in T$ then $\neg A \notin T$,
2. for all $j \in [0, \eta - 1]$, either $\{a_j\} \in T$ or $\neg\{a_j\} \in T$ (but never both),
3. if $C \sqcap D \in T$, then $C \in T$ and $D \in T$; if $C \sqcup D \in T$, then $C \in T$ or $D \in T$,
4. for every GCI $\top \sqsubseteq C \in \mathcal{T}$, we have $C \in T$.

An *anonymous* concept type T is such that $\{\neg\{a_0\}, \ldots, \neg\{a_{\eta-1}\}\} \subseteq T$. An a-type T is such that $\{a\} \in T$ and for all $\{b\}$ with $b \in \mathsf{N_I}(\mathcal{O})$ distinct from a, we have $\neg\{b\} \in T$ (see e.g. the named states in [21, Section 5] and the named types in [13, Section 4]).

An *activity vector act* is a tuple in $\{\top, \bot\}^\alpha$ where the i-th component $act[i]$ indicates whether the concrete feature f_i (from $\mathsf{N_{CF}}(\mathcal{O})$) is defined ($\top$) or not ($\bot$). The set of activity vectors is written ACT. Since the interpretation of each concrete feature is a partial function, only the registers corresponding to positions where $act[i] = \top$ represent meaningful feature values. While the data tree assigns values to all registers uniformly, entries with $act[i] = \bot$ are ignored when encoding $\mathcal{ALCO}(\mathcal{D})$ interpretations. A *symbolic link* is a pair $(r, a) \in \mathsf{N_R}(\mathcal{O}) \times \mathsf{N_I}(\mathcal{O})$ intended to represent that the current element is related to the interpretation of a via the interpretation of r. The set SLinks of symbolic link sets is the powerset $\mathcal{P}(\mathsf{N_R}(\mathcal{O}) \times \mathsf{N_I}(\mathcal{O}))$. A *local abstraction* LA is a triple in $\mathsf{CTypes}(\mathcal{O}) \times \mathsf{SLinks} \times \mathsf{ACT}$. A *global abstraction* GA is a function $\mathsf{GA} : \mathsf{N_I}(\mathcal{O}) \to \mathsf{CTypes}(\mathcal{O}) \times \mathsf{ACT}$. A location of the forthcoming automaton $\mathbb{A}$ is structured in such a way that it contains a local abstraction LA (related to the associated node) and a global abstraction GA that records a fixed knowledge about the individual names. Such pairs are called *contextual abstractions* $\mathsf{CA} = (\mathsf{GA}, \mathsf{LA})$.

Branching degree $\mathtt{d}$. We need to determine a branching degree $\mathtt{d}$ that allows us to satisfy all kinds of existential restrictions. To do so, we introduce the following values.

- $N_{\mathsf{ex}} \stackrel{\text{def}}{=} |\{\exists r.C \mid \exists r.C \in \mathsf{Sub}(\mathcal{O})\}|$.
- $N_{\mathsf{cd}} \stackrel{\text{def}}{=} |\{\exists\ v_1 : p_1, \ldots, v_k : p_k.\ \Theta \mid \exists\ v_1 : p_1, \ldots, v_k : p_k.\ \Theta \in \mathsf{Sub}(\mathcal{O})\}|$.
- $N_{\mathsf{var}} \stackrel{\text{def}}{=} \max(\{0\} \cup \{k \mid \exists\ v_1 : p_1, \ldots, v_k : p_k.\ \Theta \in \mathsf{Sub}(\mathcal{O})\})$.

The rationale to define $\mathtt{d}$ is based on the following requirements. For each $\exists r.C \in \mathsf{Sub}(\mathcal{O})$, one needs one direction to have a witness individual. Similarly, for each $\exists\ v_1 : p_1, \ldots, v_k : p_k.\ \Theta \in \mathsf{Sub}(\mathcal{O})$, k directions are sufficient to get k witness values in $\mathbb{D}$. Finally, the tree interpretation property for $\mathcal{ALCO}(\mathcal{D})$ is based on the representation of a forest such that each of its trees corresponds to the unfolding of the interpretation from some individual name in $\mathsf{N_I}(\mathcal{O})$. As this is encoded at the root of the data tree accepted by $\mathbb{A}$, the degree should be also at least $|\mathsf{N_I}(\mathcal{O})|$. So, it makes sense to define $\mathtt{d}$ as the value $\max\left\{N_{\mathsf{ex}} + N_{\mathsf{cd}} \times N_{\mathsf{var}},\ |\mathsf{N_I}(\mathcal{O})|\right\}$; the soundness proof of Lemma 6 shall confirm this formally.

Correspondence between directions, role names and restrictions. Once the degree $\mathtt{d}$ is defined, one needs to establish a discipline to put in correspondence directions in $[0, \mathtt{d} - 1]$, role names and restrictions. To do so, below, we define the maps λ and dir. Indeed, a

standard approach consists in booking directions in $[0, \mathtt{d}-1]$ for each role name in $\mathsf{N}_{\mathbf{R}}(\mathcal{O})$. The map λ takes care of assigning directions to existential restrictions and to role paths in existential CD-restrictions. By contrast, the map *dir* is uniquely determined by λ and returns a set of directions among $[0, \mathtt{d}-1]$ to each role name. Such a discipline is required to construct the transitions in $\mathbb{A}$ as one needs to agree on the directions that are relevant for getting witnesses of a given restriction. We introduce the following sets.

- $\mathcal{E}_r \stackrel{\text{def}}{=} \{\exists r_{i_1}.C_1, \ldots, \exists r_{i_{N_{\mathsf{ex}}}}.C_{N_{\mathsf{ex}}}\}$ contains all the existential restrictions from $\mathsf{Sub}(\mathcal{O})$.
- $\mathcal{E}_{\mathsf{cd}} \stackrel{\text{def}}{=} \{C^\star_1, \ldots, C^\star_{N_{\mathsf{cd}}}\}$ contains all the existential CD-restrictions from $\mathsf{Sub}(\mathcal{O})$.

We define the labeling domain as: $\mathcal{B} \stackrel{\text{def}}{=} \mathcal{E}_r \cup \{(C^\star_i, j) \mid 1 \leq i \leq N_{\mathsf{cd}},\ 1 \leq j \leq N_{\mathsf{var}}\}$. The global labeling function $\lambda : \mathcal{B} \to [0, \mathtt{d}-1]$, that is designed as a bijection, is defined as:

- $\lambda(\exists r_{i_k}.C_k) \stackrel{\text{def}}{=} k-1$, for $1 \leq k \leq N_{\mathsf{ex}}$,
- $\lambda(C^\star_i, j) \stackrel{\text{def}}{=} N_{\mathsf{ex}} + (i-1) \times N_{\mathsf{var}} + (j-1)$, for $1 \leq i \leq N_{\mathsf{cd}},\ 1 \leq j \leq N_{\mathsf{var}}$.

The definition of the map *dir* is provided below. Let $r \in \mathsf{N}_{\mathbf{R}}(\mathcal{O})$.

$$dir(r) \stackrel{\text{def}}{=} \left\{ i \in [0, \mathtt{d}-1] \;\middle|\; \begin{array}{l} \lambda^{-1}(i) = \exists r.C \text{ for some } C, \text{ or} \\ \lambda^{-1}(i) = (C^\star, j) \text{ where } C^\star = \exists\ v_1{:}p_1, \ldots, v_k{:}p_k.\ \Theta, \\ \quad \text{with } 1 \leq j \leq k \text{ and } p_j = r \cdot f \end{array} \right\}.$$

Dimension β and conventions about registers. The forthcoming automaton $\mathbb{A}$ accepts infinite data trees $\mathbb{t} : [0, \mathtt{d}-1] \to \mathbf{\Sigma} \times \mathbb{D}^\beta$ and it remains to define β before moving to the final steps of the construction. Since $\mathsf{N}_{\mathbf{CF}}(\mathcal{O})$ contains α concrete features, we require at least α registers. Though global abstractions are persistent memories, this is not sufficient to handle the values returned by the concrete features for the individual names. This is useful to consider the satisfaction of CD-restrictions requiring as witness individuals those interpreting some individual name in $\mathsf{N}_{\mathbf{I}}(\mathcal{O})$. That is why, we augment the dimension by $|\mathsf{N}_{\mathbf{I}}(\mathcal{O})| \times \alpha$ so that it is possible to carry all over the runs the values of the concrete features for such individuals. Consequently, we set $\beta \stackrel{\text{def}}{=} (\eta+1) \times \alpha$. To use the registers in $\mathrm{REG}(\beta, \mathtt{d})$, we follow conventions that happen to be handy and that diverge slightly from the general notations introduced in Section 3. The distinguished registers $x_1, \ldots, x_\beta$ referring to values at the current node shall be represented by $x_1, \ldots, x_\alpha, x_{a_0,1}, \ldots, x_{a_0,\alpha}, \ldots, x_{a_{\eta-1},1}, \ldots, x_{a_{\eta-1},\alpha}$. Hence, implicitly x_j refers to the value of the concrete feature f_j, if any. Similarly, $x_{a_\ell,j}$ refers to the value of f_j for the interpretation of the individual name a_ℓ. Furthermore, we use x^i_j (resp. $x^i_{a_\ell,j}$) to refer to the value of f_j for the ith child (resp. for the individual name a_ℓ). However, by construction of $\mathbb{A}$, we require that $x_{a_\ell,j}$ is enforced to be equal to $x^i_{a_\ell,j}$ as such registers are dedicated to store global values.

4.2 Definition of the Constraint Automaton $\mathbb{A}$

Now, we define the TGCA $\mathbb{A} = (Q, \mathbf{\Sigma}, \mathtt{d}, \beta, Q_{in}, \delta, F)$. Each non-root node $n \in [0, \mathtt{d}-1]^+$ in a run of the automaton may represent an element of the interpretation domain, but this is not systematic. A node represents a domain element if its location is not the designated *sink location* – a special marker to identify *inactive* nodes. In the active case, the location encodes a contextual abstraction as previously defined. By contrast, the root node ε does not correspond to any domain element. It serves as a technical device to assemble a forest-shaped model into a single data tree: its immediate children within $[0, \eta-1]$ encode the interpretations of the individual names in the ABox (recall we can safely assume (UNA)). The transitions are designed to enforce the semantical properties of the description logic $\mathcal{ALCO}(\mathcal{D})$ based on the contextual abstractions. Below, we present the definition of the constraint automaton $\mathbb{A}$ based on the definitions in Section 4.1.

Set of locations and alphabet. The set Q is defined as $\mathcal{CA} \cup \{ⓔ, \square\}$ and $F \stackrel{\text{def}}{=} Q$, where

- $\mathcal{CA} \stackrel{\text{def}}{=} \mathcal{GA} \times \mathcal{LA}$ is the set of *contextual abstractions*, where $\mathcal{GA}$ is the set of *global abstractions* and $\mathcal{LA}$ the set of *local abstractions*;
- ⓔ is a distinguished location used only at the root (*root location*) and $Q_{in} \stackrel{\text{def}}{=} \{ⓔ\}$,
- $\square$ is a *sink location* used to label nodes that represent no domain elements.

The alphabet $\mathbf{\Sigma}$ is $\{\mathtt{a}\}$; the automaton $\mathbb{A}$ does not rely on input symbols – only the structure of the tree and the tuples in $\mathbb{D}^\beta$ matter.

Transition relation. Before providing the formal definition, let us briefly explain its main principles. Each non-root node in a run is either associated with a contextual abstraction representing a candidate element of the interpretation domain, or is labelled with the sink location $\square$. If a node is associated with a contextual abstraction, its concept type encodes the concepts it must belong to in any corresponding interpretation: if a concept appears in the concept type of a node, then the interpreted element is required to satisfy that concept and this creates obligations (see the conditions for general transitions). Role names are interpreted with the help of two kinds of links: (i) *direct links* to the children, where each direction in $[0, \mathtt{d}-1]$ is possibly attached to a role name; and (ii) *symbolic links* connecting a node to the interpretation of individual names from the ABox. These links are used to satisfy CD-restrictions, existential restrictions and value restrictions.

Constraints are evaluated using register values drawn from the current node and its children (as determined by λ). The current node stores its own concrete feature values via the registers x_j and also the concrete feature values for the interpretation of the individual names via the registers $x_{a,j}$. These latter values must remain constant during the run, and this is explicitly enforced by the transition relation with the help of the constraints. Activity vectors determine which concrete feature values are defined and guarantee that only the defined values are referred to in the constraints. The global abstraction – common to all nodes labelled by non-sink locations – must remain constant too. Formally, the transition relation δ is a subset of $Q \times \mathbf{\Sigma} \times \mathrm{Cons}(\beta, \mathtt{d}) \times Q^{\mathtt{d}}$ and contains three kinds of transitions.

(a) Padding transition. There is a unique padding transition $\mathtt{t}_\square \stackrel{\text{def}}{=} (\square, \mathtt{a}, \top, \square, \ldots, \square)$.

(b) Root transitions. From the root location ⓔ, the automaton performs an initialization step by assigning contextual abstractions to the first η children, each of which represents the interpretation of an individual name from the ABox. Each such contextual abstraction is of the form $(\mathsf{GA}_i, (T_i, sl_i, act_i))$. For each individual name a_i, the local abstraction assigned to the i-th child must match the corresponding entry in the global abstraction: specifically, the concept type and activity vector must coincide with $\mathsf{GA}(a_i)$. In addition, the value of each concrete feature f_j for a_i must be consistently stored in the register $x_{a_i,j}$ at each node, which is treated as a constant throughout the run. This value originates from the local register x_j of the i-th child, which represents the interpretation of a_i. Therefore, the consistency condition $x_{a_i,j} = x_j$ must hold at that node. These requirements together form the *global information consistency check*. Beyond this, the local abstraction must correctly reflect the ABox assertions involving a_i.

Formally, root transitions are of the form $(ⓔ, \mathtt{a}, \Theta_{\mathsf{root}}, \mathsf{CA}_0, \ldots, \mathsf{CA}_{\eta-1}, \square, \ldots, \square)$, where each $\mathsf{CA}_i = (\mathsf{GA}_i, (T_i, sl_i, act_i))$ is a contextual abstraction and T_i is an a_i-type. The constraint Θ_{root} is equal to the expression $\bigwedge_{i,i' \in [0,\eta-1], j \in [1,\alpha], act_{a_i}[j]=\top} x_j^i = x_{a_i,j}^{i'}$.

The transition must also satisfy the following structural conditions: $\mathsf{GA}_0 = \cdots = \mathsf{GA}_{\eta-1}$ and for all $i \in [0, \eta-1]$, $(T_i, act_i) = \mathsf{GA}(a_i)$, $\{C \mid a_i : C \in \mathcal{A}\} \subseteq T_i$, and $\{(r, b) \mid (a_i, b) : r \in \mathcal{A}\} \subseteq sl_i$. From now on, for each individual name $a \in \mathsf{N_I}(\mathcal{O})$, we let

$\mathsf{GA}(a) = (T_a, act_a)$, where T_a denotes the concept type and act_a the activity vector associated to a via the global abstraction GA. This notation is well-defined due to the global information consistency check performed by the root transition.

(c) General transitions. From a location $\mathsf{CA} = (\mathsf{GA}, (T, sl, act))$, we need to find witness individuals for the existential restrictions and CD–restrictions occurring in T. Each witness can be realized in two ways: either through a child, determined via the branch labeling function λ, or via a symbolic link in sl pointing to an individual name. As for value restrictions, the transition must verify that *all* possible witnesses satisfy the required concept or constraint. For universal CD–restrictions, the constraint must hold across all applicable combinations of registers extracted from those directions and links. General transitions also maintain some global information and are of the form $\mathtt{t} = (\mathsf{CA}, \mathtt{a}, \Theta_\mathtt{t}, q_0, \ldots, q_{\mathtt{d}-1})$, where $\mathsf{CA} = (\mathsf{GA}, (T, sl, act))$ satisfies the conditions below.

1. **Existential restrictions.** For all $\exists r.C \in T$, there is $(r, a) \in sl$ such that $C \in T_a$ or $i = \lambda(\exists r.C)$, q_i is of the form $(\mathsf{GA}_i, (T_i, sl_i, act_i))$ and $C \in T_i$.
2. **Value restrictions.** For all $\forall r.C \in T$, for all $i \in dir(r)$ with q_i of the form $(\mathsf{GA}_i, (T_i, sl_i, act_i))$, we have $C \in T_i$ and for all $(r, a) \in sl$, we have $C \in T_a$.
3. **CD-restrictions.** Let $\mathrm{CD}_\exists(T)$ and $\mathrm{CD}_\forall(T)$ be the sets of existential and universal CD-restrictions in T. Let $C = \mathcal{Q}\, v_1 : p_1, \ldots, v_k : p_k.\ \Theta(v_1, \ldots, v_k)$ be such a CD-restriction with $\mathcal{Q} \in \{\exists, \forall\}$. For each p_j, we define a set Z_j of registers as follows:
   - If $p_j = f_l$, then if $act[l] = \top$, then $Z_j \stackrel{\text{def}}{=} \{x_l\}$, otherwise $Z_j \stackrel{\text{def}}{=} \emptyset$.
   - If $p_j = r \cdot f_l$, then Z_j consists of the values of the feature f_l taken from nodes reachable via the role name r, provided the corresponding feature is defined.

   $$Z_j \stackrel{\text{def}}{=} \{x^i_l \mid i \in dir(r),\ q_i = (\mathsf{GA}_i, (T_i, sl_i, act_i)), act_i[l] = \top\} \cup \{x_{a,l} \mid (r, a) \in sl,\ act_a[l] = \top\}.$$

   Let $Z \stackrel{\text{def}}{=} Z_1 \times \cdots \times Z_k$. If $C \in \mathrm{CD}_\exists(T)$, then $\Theta_C \stackrel{\text{def}}{=} \bigvee_{(y_1,\ldots,y_k)\in Z} \Theta(y_1, \ldots, y_k)$, otherwise $\Theta_C \stackrel{\text{def}}{=} \bigwedge_{(y_1,\ldots,y_k)\in Z} \Theta(y_1, \ldots, y_k)$. The constraint $\Theta_\mathtt{t}$ is defined below:

   $$\Theta_\mathtt{t} \stackrel{\text{def}}{=} \overbrace{\bigwedge_{C\in \mathrm{CD}_\exists(T)\cup\mathrm{CD}_\forall(T)} \Theta_C}^{\text{dedication to CD-restrictions}} \wedge \overbrace{\bigwedge_{\substack{i\in[0,\mathtt{d}-1]\\ q_i\neq\Box}} \bigwedge_{j=1}^{\alpha} \bigwedge_{j'\in[0,\eta-1],\ act_{a_{j'}}[j]=\top} x_{a_{j'},j} = x^i_{a_{j'},j}}^{\text{propagation of global values}}.$$

   The constraint $\Theta_\mathtt{t}$ is uniquely determined by the sequence $q, q_0, \ldots, q_{\mathtt{d}-1}$.
4. **Global abstraction preservation.** For each $i < \mathtt{d}$ with $q_i \neq \Box$, we require $\mathsf{GA}_i = \mathsf{GA}$.
5. **Inactive children.** For $i \in [0, \mathtt{d}-1]$, if $\lambda^{-1}(i) = \exists r.C$ and $\lambda^{-1}(i) \notin T$ or $\lambda^{-1}(i) = (\exists\ v_1 : p_1, \ldots, v_k : p_k.\ \Theta, j)$ and ($j > k$ or $\exists\ v_1 : p_1, \ldots, v_k : p_k.\ \Theta \notin T$), then $q_i = \Box$.
6. **Anonymity.** For $i < \mathtt{d}$ such that q_i is of the form $(\mathsf{GA}_i, (T_i, sl_i, act_i))$, T_i is an anonymous concept type.

Such general transitions are reminiscent of the augmented types in [13, Section 3].

4.3 Correctness of the Reduction

It remains to establish that $\mathrm{L}(\mathbb{A}) \neq \emptyset$ iff the ontology $\mathcal{O}$ is consistent. Forthcoming Lemma 6 states the left-to-right direction and its proof amounts to design an interpretation from a data tree in $\mathrm{L}(\mathbb{A})$ with its accepting run. Elements of the interpretation domain are non-root nodes that are not labelled by the sink location, interpretation of the concept names can be read from the concept types, and the interpretation of the concrete features are read from values in the tuples from $\mathbb{D}^\beta$ in the data tree. Requirements on the transitions complete the analysis to guarantee that an interpretation satisfying the ontology is indeed produced.

▶ **Lemma 6.** $\mathrm{L}(\mathbb{A}) \neq \emptyset$ *implies* $\mathcal{O}$ *is consistent.*

▶ **Lemma 7.** *$\mathcal{O}$ is consistent with an interpretation satisfying (UNA) implies* $\mathrm{L}(\mathbb{A}) \neq \emptyset$.

The construction of the TGCA $\mathbb{A}$ can be done in exponential-time in $size(\mathcal{O})$, since all components – locations, transitions, and constraints – can be generated and assembled in exponential time. By substituting these parameters into the general complexity expression for the nonemptiness check of TGCA yields the main result below.

▶ **Theorem 8.** *The consistency problem for $\mathcal{ALCO}(\mathcal{D})$ is in k-ExpTime if $\mathcal{D}$ satisfies the conditions (C1), (C2), (C3.k) and (C4), for any $k \geq 1$.*

As a corollary of the combination of Lemma 6 and Lemma 7, if $\mathcal{O}$ is consistent, then it admits a forest-like interpretation satisfying it with branching degree bounded by $\max\{N_{\mathsf{ex}} + N_{\mathsf{cd}} \times N_{\mathsf{var}},\ |\mathsf{N_I}(\mathcal{O})|\}$. This provides an alternative to the tree unfolding of the interpretations. Besides, the concrete domain $\mathcal{D} = (\mathbb{Z}, (+_k)_{k\in\mathbb{N}})$ for which the consistency problem $\mathcal{ALC}(\mathcal{D})$ is shown undecidable in [31, Proposition 1], has an infinite set of predicate symbols but does not satisfy the notion of amalgamation used in Section 2.1 (see also [31, Example 3]).

5 Extensions

In this section, we consider the additional ingredients described in Section 2.3 and we show how our automata-based approach can be adapted, preserving the ExpTime-membership. Due to lack of space, we only state their main results about functional role names (resp. constraint assertions).

▶ **Theorem 9.** *The consistency problem for $\mathcal{ALCO}(\mathcal{D})$ augmented either with constraint assertions or with functional role names is in k-ExpTime if $\mathcal{D}$ satisfies the conditions (C1), (C2), (C3.k) and (C4), for any $k \geq 1$.*

Note that nothing prevents us from using constraint assertions and functional role names in the same logic. Now, let us consider in more detail $\mathcal{ALCI}(\mathcal{D})$, see Section 2.3. The roles R in $\mathcal{ALCI}(\mathcal{D})$ are either role names r or their inverses r^-. Role assertions involve only role names because inverse role names add no expressivity in such assertions. By convention, we write $(a, b) : r^- \in \mathcal{A}$ to denote that the role assertion $(b, a) : r$ belongs to $\mathcal{A}$. Given a role R, we write R^- to denote its inverse, i.e. $R^- \stackrel{\text{def}}{=} r^-$ if $R = r$ and $R^- \stackrel{\text{def}}{=} r$ if $R = r^-$. Given an ontology $\mathcal{O}$ for $\mathcal{ALCI}(\mathcal{D})$, we write $\mathsf{Roles}(\mathcal{O})$ to denote the set $\{R_1, \ldots, R_\kappa\}$ of roles occurring in $\mathcal{O}$. In the definition of $\mathsf{Sub}(\mathcal{O})$, we remove the parts about nominals. To extend the automaton construction to $\mathcal{ALCI}(\mathcal{D})$, the transitions for $\mathbb{A}$ need to encode access to new witnesses by considering also *the edge* to the parent node.

Parent abstractions and new contextual abstractions. A *parent abstraction* PA is either a pair $(\gamma, \mathfrak{f})$ with $\gamma \in [0, \eta - 1]$ and $\mathfrak{f} : \mathsf{N_I}(\mathcal{O}) \to \mathsf{ACT}$ (encoding unknown parent, incoming direction and activity vectors of the individual names) or a triple $(R^\uparrow, T^\uparrow, act^\uparrow) \in \mathsf{Roles}(\mathcal{O}) \times \mathsf{CTypes}(\mathcal{O}) \times \mathsf{ACT}$. As expected, $R^\uparrow$ encodes the role to reach a node from its parent node, $T^\uparrow$ is the parent's concept type and $act^\uparrow$ is the parent's activity vector. The set $\mathcal{CA}$ of contextual abstractions is now defined as the product $\mathcal{PA} \times \mathcal{LA}$, where $\mathcal{PA}$ denotes the set of parent abstractions. Contextual abstractions no longer include global abstractions since we gave up the nominals (but the pair $(\gamma, \mathfrak{f})$ contains a bit of global information). Similarly, the local abstractions have no more symbolic links.

Automaton construction for $\mathcal{ALCI}(\mathcal{D})$. We define the TGCA $\mathbb{A} = (Q, \mathbf{\Sigma}, \mathtt{d}, \beta, Q_{in}, \delta, F)$ from some ontology $\mathcal{O}$ as follows, extending the construction in Section 4.2.

- $\mathtt{d} \stackrel{\text{def}}{=} \max\{N_{\mathsf{ex}} + N_{\mathsf{cd}} \times N_{\mathsf{var}}, |\mathsf{N_I}(\mathcal{O})|\}$, where N_{ex} is the number of existential restrictions in $\mathsf{Sub}(\mathcal{O})$. To support the reasoning on the concrete feature values of the parent nodes, the new registers $x_1^{\uparrow}, \ldots, x_\alpha^{\uparrow}$ are dedicated to the parent nodes. Consequently, $\beta \stackrel{\text{def}}{=} \alpha \times (\eta + 2)$. Unlike what happens in the automata for $\mathcal{ALCO}(\mathcal{D})$, the registers dedicated to the individual names are not maintained constant along the run but only used at the nodes $0, \ldots, \eta - 1$ (labelled by some location of the form $((\gamma, \mathfrak{f}), \mathsf{LA})$).
- $Q \stackrel{\text{def}}{=} \mathcal{CA} \cup \{\textcircled{\scriptsize E}, \Box\}$ (suitable definition for $\mathcal{CA}$), $F \stackrel{\text{def}}{=} Q$, $Q_{in} \stackrel{\text{def}}{=} \{\textcircled{\scriptsize E}\}$ and $\mathbf{\Sigma} \stackrel{\text{def}}{=} \{\mathtt{a}\}$.
- The relation δ is a subset of $Q \times \mathbf{\Sigma} \times \mathrm{Cons}(\beta, \mathtt{d}) \times Q^{\mathtt{d}}$. As for $\mathcal{ALCO}(\mathcal{D})$, we distinguish three kinds of transitions but the general transitions have now two categories. There is still a unique padding transition $\mathtt{t}_\Box \stackrel{\text{def}}{=} (\Box, \mathtt{a}, \top, \Box, \ldots, \Box)$. In the root transitions $(\textcircled{\scriptsize E}, \mathtt{a}, \Theta_{\mathsf{root}}, \mathsf{CA}_0, \ldots, \mathsf{CA}_{\eta-1}, \Box, \ldots, \Box)$, each CA_i is of the form $((i, \mathfrak{f}_i), (T_i, act_i))$ with the same requirements as for $\mathcal{ALCO}(\mathcal{D})$ (except that there is no more global abstractions) but we require that $\mathfrak{f}_0 = \cdots = \mathfrak{f}_{\eta-1}$. Furthermore, Θ_{root} keeps the same value. There is nevertheless an additional requirement due to the absence of global abstractions: for all $(a_i, a_{i'}) : r \in \mathcal{A}$, for all $\forall r.C \in T_i$ and $\forall r^-.C' \in T_{i'}$, we have $C \in T_{i'}$ and $C' \in T_i$. The main changes are for the general transitions $\mathtt{t} = (\mathsf{CA}, \mathtt{a}, \Theta_{\mathtt{t}}, q_0, \ldots, q_{\mathtt{d}-1})$ satisfying the following conditions with $\mathsf{CA} = (\mathsf{PA}, (T, act))$.
  - **(Existential restrictions)** For all $\exists R.C \in T$, $i = \lambda(\exists R.C)$, q_i of the form $(\mathsf{PA}_i, (T_i, act_i))$ and $C \in T_i$ or $R^- = R^{\uparrow}$ and $C \in T^{\uparrow}$ if $\mathsf{PA} = (R^{\uparrow}, T^{\uparrow}, act^{\uparrow})$ (new option).
  - **(Value restrictions)** For all $\forall R.C \in T$, for all $i \in dir(R)$ with $q_i = (\mathsf{PA}_i, (T_i, act_i))$, we have $C \in T_i$, and if $R^- = R^{\uparrow}$, then $C \in T^{\uparrow}$ if $\mathsf{PA} = (R^{\uparrow}, T^{\uparrow}, act^{\uparrow})$ (new requirement).
  - **(Back-propagation)** If $q_i = (\mathsf{PA}_i, (T_i, act_i))$, $\forall R.C \in T_i$ and $i \in dir(R^-)$, then $C \in T$. This is a new condition due to inverse roles.
  - **(CD-restrictions)** Let $\mathrm{CD}_\exists(T)$ and $\mathrm{CD}_\forall(T)$ be the sets of existential and universal CD-restrictions in T. Let $C = \mathcal{Q}\, v_1 : p_1, \ldots, v_k : p_k.\ \Theta$ be such a CD-restriction with $\mathcal{Q} \in \{\exists, \forall\}$. For each role path p_j, we define $Z_j \subseteq \mathrm{REG}(\beta, \mathtt{d})$ as follows.
    - If $p_j = f_l$, then if $act[l] = \top$, then $Z_j \stackrel{\text{def}}{=} \{x_l\}$ else $Z_j \stackrel{\text{def}}{=} \emptyset$.
    - If $p_j = R \cdot f_l$, then Z_j is defined as follows. If $\mathsf{PA} = (R^{\uparrow}, T^{\uparrow}, act^{\uparrow})$, then
    $$Z_j \stackrel{\text{def}}{=} \{x_l^i \mid i \in dir(R),\ q_i = (\mathsf{PA}_i, (T_i, act_i)),\ act_i[l] = \top\} \cup \left\{x_l^{\uparrow} \;\middle|\; R^{\uparrow} = R^-,\ act^{\uparrow}[l] = \top\right\}$$
    Otherwise PA is of the form $(\gamma, \mathfrak{f})$, we need to take advantage of the registers dedicated to individual names: $Z_j \stackrel{\text{def}}{=} \{x_l^i \mid i \in dir(R),\ q_i = (\mathsf{PA}_i, (T_i, act_i)),\ act_i[l] = \top\} \cup \{x_{a_{\gamma'}, l} \mid (a_\gamma, a_{\gamma'}) : R \in \mathcal{A}, \mathfrak{f}(a_{\gamma'})[l] = \top\}$.

  Let $Z \stackrel{\text{def}}{=} \prod Z_j$. If $C \in \mathrm{CD}_\exists(T)$, then $\Theta_C \stackrel{\text{def}}{=} \bigvee_{(y_1, \ldots, y_k) \in Z} \Theta(y_1, \ldots, y_k)$, otherwise $\Theta_C \stackrel{\text{def}}{=} \bigwedge_{(y_1, \ldots, y_k) \in Z} \Theta(y_1, \ldots, y_k)$. The constraint $\Theta_{\mathtt{t}}$ is defined below, where the new part expresses the consistency of the registers dedicated to the parent node ($x_j^{\uparrow\, i}$ denotes the register at the ith child for the parent's value of f_j).
  $$\Theta_{\mathtt{t}} = \bigwedge_{C \in \mathrm{CD}_\exists(T) \cup \mathrm{CD}_\forall(T)} \Theta_C \wedge \bigwedge_{\substack{i \in [0, \mathtt{d}-1] \\ q_i \neq \Box}} \ \bigwedge_{j \in [1, \alpha], act[j] = \top} x_j^{\uparrow\, i} = x_j.$$
  - **(Inactive children)** Same condition as for $\mathcal{ALCO}(\mathcal{D})$.
  - **(Parent information propagation)** For each i such that q_i is a contextual abstraction $((R_i^{\uparrow}, T_i^{\uparrow}, act_i^{\uparrow}), (T_i, act_i))$, we require $i \in dir(R_i^{\uparrow})$ and $(T, act) = (T_i^{\uparrow}, act_i^{\uparrow})$.

Soundness is shown as for Lemma 10 by extending the interpretation of role names.

▶ **Lemma 10.** $\mathrm{L}(\mathbb{A}) \neq \emptyset$ *implies* $\mathcal{O}$ *is consistent.*

Similarly, completeness is also shown along the lines of the proof of Lemma 7.

▶ **Lemma 11.** *$\mathcal{O}$ is consistent with an interpretation satisfying (UNA) implies* $\mathrm{L}(\mathbb{A}) \neq \emptyset$.

As far as complexity is concerned, the main differences with the construction of $\mathbb{A}$ for $\mathcal{ALCO}(\mathcal{D})$ include β still bilinear in α and η, and the presence of parent abstractions. All of this causes no harm and we can establish Theorem 12.

▶ **Theorem 12.** *The consistency problem for $\mathcal{ALCI}(\mathcal{D})$ is in k-ExpTime if $\mathcal{D}$ satisfies the conditions (C1), (C2), (C3.k) and (C4), for any $k \geq 1$.*

As shown herein, our developments for $\mathcal{ALCO}(\mathcal{D})$ can be naturally adapted to $\mathcal{ALCI}(\mathcal{D})$, witnessing the robustness of our approach. It is open whether there is an adaptation for the extension $\mathcal{ALCOI}(\mathcal{D})$ with inverse role names *and* nominals. Indeed, in $\mathcal{ALCOI}(\mathbb{Q}, <)$, the ontology $\mathcal{O} = (\{\top \sqsubseteq \exists\, v_1 : f_1, v_2 : r \cdot f_1.\ v_1 < v_2, \top \sqsubseteq \exists r^-.\{a\}\}, \{a : \forall\, v_1 : f_1, v_2 : r \cdot f_1.\ \Theta\})$, can be made consistent and for any interpretation $\mathcal{I}$ satisfying it, $a^{\mathcal{I}}$ has an *infinite amount* of r-successors, which cannot be captured currently with the finite sets Z_j. It is not so much the combination of inverse roles and nominals that is problematic, see e.g. the handling of the μ-calculus with converse and nominals in [32], but the additional presence of CD-restrictions.

6 Conclusion

We have considered a class of concrete domains $\mathcal{D}$ for which the consistency problem for $\mathcal{ALCO}(\mathcal{D})$ can be reduced to the nonemptiness problem for constraint automata parameterised by $\mathcal{D}$, leading to the ExpTime-membership of the consistency problem (Theorem 8). This follows an automata-based approach, whose versatility allowed us to add ingredients such as inverse roles (while giving up nominals) and functional role names (Section 5). Furthermore, we carefully examined the conditions on the concrete domain $\mathcal{D}$ to get the ExpTime upper bound. In particular, we have shown that the nonemptiness problem for constraint automata is in ExpTime whenever $\mathcal{D}$ satisfies the conditions (C1), (C2) and (C3.1) (Theorem 5). The decidability/complexity status of the logics $\mathcal{ALCOI}(\mathcal{D})$ is left open (also evoked in [3, Section 7] for some extension).

References

1 J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983. doi:10.1145/182.358434.

2 F. Baader. Description Logics. In *Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Tutorial Lectures*, volume 5689 of *LNCS*, pages 1–39. Springer, 2009. doi:10.1007/978-3-642-03754-2_1.

3 F. Baader, S. Borgwardt, F. De Bortoli, and P. Koopmann. Concrete domains meet expressive cardinality restrictions in description logics. In *CADE'25*, volume 15943 of *LNCS*, pages 676–695. Springer, 2025. doi:10.1007/978-3-031-99984-0_35.

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

5 F. Baader, J. Hladik, C. Lutz, and F. Wolter. From Tableaux to Automata for Description Logics. *Fundamenta Informaticae*, 57(2–4):247–279, 2003. URL: http://content.iospress.com/articles/fundamenta-informaticae/fi57-2-4-08.

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

7 F. Baader, I. Horrocks, and U. Sattler. Description logics. In F. van Harmelen, V. Lifschitz, and B.W. Porter, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 135–179. Elsevier, 2008. doi:10.1016/S1574-6526(07)03003-9.

8 F. Baader and C. Lutz. Description logic. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, pages 757–819. Elsevier, 2006.

9 F. Baader and J. Rydval. Using model theory to find decidable and tractable description logics with concrete domains. *Journal of Automated Reasoning*, 66(3):357–407, 2022. doi:10.1007/S10817-022-09626-2.

10 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 *LNAI*, pages 162–173. Springer, 2002.

11 A. Bhaskar and M. Praveen. Realizability problem for constraint LTL. *Information & Computation*, 296:105126, 2024. doi:10.1016/J.IC.2023.105126.

12 P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

13 S. Borgwardt, F. De Bortoli, and P. Koopmann. The precise complexity of reasoning in $\mathcal{ALC}$ with ω-admissible concrete domains. In *DL'24*, volume 3739 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2024.

14 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.

15 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. doi:10.3233/978-1-61499-672-9-1440.

16 K. Chatterjee and M. Henzinger. An $\mathcal{O}(n^2)$ time algorithm for alternating Büchi games. In *SODA'12*, pages 1386–1399. SIAM, 2012.

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

18 S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. *Information & Computation*, 205(3):380–415, 2007. doi:10.1016/J.IC.2006.09.006.

19 S. Demri and K. Quaas. First steps towards taming description logics with strings. In *JELIA'23*, volume 14281 of *LNCS*, pages 322–337. Springer, 2023. doi:10.1007/978-3-031-43619-2_23.

20 S. Demri and K. Quaas. Constraint automata on infinite data trees: From CTL(Z)/CTL*(Z) to decision procedures. *Logical Methods in Computer Science*, 21(2):16:1–16:79, 2025. doi:10.46298/LMCS-21(2:16)2025.

21 S. Demri and U. Sattler. Automata-theoretic decision procedures for information logics. *Fundamenta Informaticae*, 53(1):1–22, 2002. URL: http://content.iospress.com/articles/fundamenta-informaticae/fi53-1-01.

22 N. Labai. *Automata-based reasoning for decidable logics with data values*. PhD thesis, TU Wien, May 2021.

23 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.

24 C. Lutz. NEXPTIME-complete description logics with concrete domains. In *IJCAR'01*, volume 2083 of *LNCS*, pages 46–60. Springer, 2001. doi:10.1007/3-540-45744-5_5.

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

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

27 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. doi:10.1007/S10817-006-9049-7.

28 M. Marx. Complexity of modal logic. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, pages 139–179. Elsevier, 2006.

29 V. Pratt. Models of program logics. In *FoCS'79*, pages 115–122, 1979.

30 I. Pratt-Hartmann. *Fragments of First-Logic*. Oxford University Press, 2023.

31 J. Rydval. *Using Model Theory to Find Decidable and Tractable Description Logics with Concrete Domains*. PhD thesis, Dresden University, 2022.

32 U. Sattler and M. Vardi. The hybrid mu-calculus. In *IJCAR'01*, volume 2083 of *LNAI*, pages 76–91. Springer, 2001.

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

34 S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000. doi:10.1613/JAIR.705.

35 M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986. doi:10.1016/0022-0000(86)90026-7.

36 M.Y. Vardi. Automata-theoretic techniques for temporal reasoning. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, pages 971–989. Elsevier, 2006.

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