First Steps Towards Taming Description Logics with Strings

Stéphane Demri  
CNRS & ENS Paris-Saclay

Joint work with  
Karin Quaas, Leipzig University

DL Seminar, June 2024
Concrete Domains in TCS

- Constraint satisfaction problems (CSP).

- Satisfiability Modulo Theory (SMT) solvers.
  String theories, arithmetical theories, array theories, etc.
  See e.g. [Barrett & Tinelli, Handbook 2018]

- Verification of database-driven systems.
  [Deutsch et al., SIGMOD 2014; Felli et al., AAAI’22]

- LTL modulo theories.
  See e.g. [Geatti et al., IJCAI’22; Rodríguez & Sánchez, CAV’23]

- Description logics with concrete domains.
  [Baader & Hanschke, IJCAI’91, Lutz, PhD 2002; Lutz, AiML’02]
Concrete Domains

• Concrete domain $\mathcal{D} = (\mathbb{D}, R_1, R_2, \ldots)$: fixed non-empty domain with a family of relations.

• $(\mathbb{N}, <, +1)$, $(\mathbb{Q}, <, =)$, $(\mathbb{N}, <, =)$, $(\{0, 1\}^*, <_{\text{pre}}, <_{\text{suf}})$.

• Temporal concrete domain $\mathcal{D}_A = (I_Q; (R_i)_{i \in [1, 13]})$ with
  - $I_Q$: set of closed intervals $[r, r'] \subseteq \mathbb{Q}$
  - $(R_i)_{i \in [1, 13]}$ is the family of 13 Allen’s relations.

  [Allen, CACM 1983]

• Concrete domain RCC8 with space regions in $\mathbb{R}^2$ contains topological relations between spatial regions.

  See e.g. [Wolter & Zakharyaschev, KR’00]
Description Logics with Concrete Domains

• Description logics are well-known logical formalisms dedicated to ontologies.
  [Baader et al, Book 2017]

• Need to express concrete properties about data in ontologies. (e.g. age, duration, name, size, etc.)

• General scheme for integrating concrete domains in DLs.
  [Baader & Hanschke, IJCAI’91]
  – declarative semantics close to the usual semantics for DLs,
  – generic extensions of DLs with various concrete domains,
  – inference algorithms combined with theory reasoning.
Enforcing an Infinite Set of Constraints

\[ \top \subseteq \exists r. \left[ x_1 = Sx_1 \land x_3 = Sx_3 \land x_1 < x_2 < Sx_2 < x_3 \right] \]

- For \((\mathbb{Q}, <)\), the sequence \( \left( \begin{array}{c} 1 \\ \frac{i+1}{i+2} \\ \vdots \end{array} \right)_{i \in \mathbb{N}} \) does the job.

- For \(\{a, b\}^*\) with the lexicographical ordering, the sequence \( \left( \begin{array}{c} b \\ a^{i+1} \\ \varepsilon \end{array} \right)_{i \in \mathbb{N}} \) does the job.

- For \((\mathbb{N}, <)\), no sequence does the job.
Methods for Handling Concrete Domains

- Tableaux-based decision procedures for \( \omega \)-admissible concrete domains. [Lutz & Miličić, JAR 2007]
  - \( \mathcal{R} = (\mathbb{R}, <, =, (=r)_{r \in \mathbb{R}}) \) is \( \omega \)-admissible.
  - \( \mathcal{N} = (\mathbb{N}, <, =, (=n)_{n \in \mathbb{N}}) \) is not \( \omega \)-admissible.
  (complexity in [Borgwardt, De Boritoli, Hoopmann, 2024])

- Translation into a decidable extension of MSO with bounding quantifier B. [Carapelle & Kartzow & Lorhey, JCSS 2016]
  (reduction method from 70’s, see e.g. [Gabbay, IML 1975])
  - EHD approach developped with \( \text{Bool}(\text{MSO}, \text{WMSO}+B) \) over infinite trees of finite branching degree.
  - Decidability of concept satisfiability problem w.r.t. general TBoxes for \( \text{ALCP}^P(\mathcal{N}) \). [Carapelle & Turhan, ECAI’16]

- Translation into Rabin tree automata over finite alphabets using approximations for satisfiable symbolic interpretations.
  - Concept satisfiability problem w.r.t. general TBoxes for \( \text{ALCF}^P(\mathcal{N}) \) in \( \text{EXP TIME} \). [Labai & Ortiz & Šimkus, KR’20]
Finite Strings with the Prefix Relation

- $\mathcal{D}_\Sigma = (\Sigma^*, \prec_{pre}, =, (=_w)_{w \in \Sigma^*})$.

- $(\mathbb{N}, <, =, (=_n)_{n \in \mathbb{N}})$ corresponds to $\mathcal{D}_\Sigma$ with singleton $\Sigma$.

- Concept satisfiability problem w.r.t. general TBoxes for $\mathcal{ALC}(\mathcal{W})$ with $\mathcal{W} = (\Sigma^*, \cdot, =, (=_w)_{w \in \Sigma^*})$ is undecidable. [Lutz, PhD 2002]

- No known decidability results for description logics with $\mathcal{D}_\Sigma$. 

\[(\exists r_0 r_1 r_2 \cdot (\text{name} \prec_{pre} SSS \text{name})) \land \forall r_0 \cdot (\text{name} \prec_{pre} S \text{name})\]

\('S' similar to neXt in temporal logics\)
**ALC in a Nutshell**

- **Complex concepts.**

  \[ C ::= \top | \bot | A | \neg C | C \sqcap C | C \sqcup C | \exists r.C | \forall r.C, \]

  with concept names \( A \) and role names \( r \).

- **Terminological Box (TBox) \( \mathcal{T} \):** finite collection of GCIs.

- **Concept satisfiability problem w.r.t. general TBoxes (TSAT(\( ALC \))):**

  **Input:** A concept \( C_0 \) and a TBox \( \mathcal{T} \).

  **Question:** Is there an interpretation \( \mathcal{I} \) such that \( \mathcal{I} \models \mathcal{T} \) and \( C_0^\mathcal{I} \neq \emptyset \)?

- **TSAT(\( ALC \))** is \( \text{EXP\:TIME\:complete} \).
Description Logic $\mathcal{ALCFP}(\mathcal{D}_\Sigma)$

$\exists r_0 r_1 r_2. [\text{name} <_{\text{pre}} SSS \text{name}]$

- New (atomic) concepts of the form $\exists P.[[\Theta]]$ and $\forall P.[[\Theta]]$:
  - non-empty sequence $P$ of role names (role path),
  - Boolean constraint $\Theta$ built over terms of the form $S^jx$ with $j \leq |P|$ and atomic constraints of the form
    
    $t <_{\text{pre}} t' \quad t = t' \quad =_w (t)$ (also written $t = w$)

- Interpretation $\mathcal{I} = (\Delta \mathcal{I}, \cdot \mathcal{I}, v)$ with $v : \Delta \mathcal{I} \times \text{VAR} \rightarrow \Sigma^*$.

- $(r_1 r_2 \ldots r_n)\mathcal{I} \overset{\text{def}}{=} \text{set of tuples } (a_0, \ldots, a_n) \text{ in } (\Delta \mathcal{I})^{n+1} \text{ such that } (a_{i-1}, a_i) \in r_i \mathcal{I} \text{ for all } i \in [1, n].$

$$a_0 \mathrel{r_1} a_1 \ldots \mathrel{r_n} a_n$$
Clauses for Interpreting the Concepts Involving $\mathcal{D}_\Sigma$

- Satisfaction relation $\mathcal{I}, \pi = (a_0, a_1, \ldots, a_n) \models \Theta$:

  - $\mathcal{I}, \pi \models S^i x <_{\text{pre}} S^j y \overset{\text{def}}{\iff} v(a_i, x) <_{\text{pre}} v(a_j, y)$, (similarly for $=$)
  - $\mathcal{I}, \pi \models S^i x = w \overset{\text{def}}{\iff} v(a_i, x) = w$

  \[
  (\exists P. [\Theta])^\mathcal{I} \overset{\text{def}}{=} \{ a_0 \in \Delta^\mathcal{I} \mid \exists a_1, \ldots, a_n \in \Delta^\mathcal{I} \text{ s.t. } \pi = (a_0, a_1, \ldots, a_n) \in P^\mathcal{I} \& \mathcal{I}, \pi \models \Theta \}
  \]

  \[
  (\forall P. [\Theta])^\mathcal{I} \overset{\text{def}}{=} \{ a_0 \in \Delta^\mathcal{I} \mid \forall a_1, \ldots, a_n \in \Delta^\mathcal{I}, \pi = (a_0, a_1, \ldots, a_n) \in P^\mathcal{I} \text{ implies } \mathcal{I}, \pi \models \Theta \}
  \]

- $ALCF \mathcal{F}^P(\mathcal{D}_\Sigma)$ has also functional role names.

  (a.k.a. abstract features)

- No finite interpretation property, e.g. with

  \[
  \mathcal{I} = \{ \top \subseteq \exists r. [x <_{\text{pre}} Sx] \}
  \]
Main Steps Leading to ExpTime-easiness of TSAT($ALCF^P(\Sigma)$)

- Automata-based approach with tree constraint automata (TCA) accepting infinite data trees with domain $\Sigma^*$.
  
  (first principles from [Vardi & Wolper, IC 1994])

- Step 0: to transform the instance so that every concept is in *simple form*, proper form to perform Step 1.

- Step 1: $C_0, T \rightarrow A$ s.t. $C_0, T$ is a positive instance of TSAT iff $L(A) \neq \emptyset$.

- Step 2: $A$ (for $\mathcal{D}_\Sigma$) $\rightarrow A'$ (for $\mathcal{N}$) such that $L(A) \neq \emptyset$ iff $L(A') \neq \emptyset$

- Step 3: complexity analysis to get $\text{ExpTime}$-completeness of TSAT. (based on [Demri & Quaas, CONCUR’23]).
Concepts in Simple Form

- A concept is in simple form if it is in NNF and all its role paths are of length at most one.

- \( \exists rr'.[S^2y <_{pre} x] \) not in simple form but concepts below in simple form:

\[
\exists r.\exists r'.\exists \varepsilon.[y <_{pre} x^{++}] \quad \top \sqsubseteq \forall r.[x = Sx^+] \quad \top \sqsubseteq \forall r'.[x^+ = Sx^{++}]
\]

- Given \( C_0, \mathcal{T} \), one can construct in polynomial-time \( C'_0, \mathcal{T}' \) in simple form s.t. \( C_0, \mathcal{T} \) positive instance of TSAT iff \( C'_0, \mathcal{T}' \) positive instance of TSAT.
Tree Interpretation Property Needed!
Constraint Automata on Words

- Constraint automata accept data words.

\[
\begin{array}{ccccccc}
\text{a} & \text{b} & \text{c} & \text{a} & \text{a} & \text{a} & \text{b} \\
0 & 0.3 & 1 & 2.3 & 3.5 & 3.51
\end{array}
\]

- Concrete domains \( \mathcal{D} = (\mathbb{D}, <, P_1, \ldots, P_l, =_\vartheta_1, \ldots, =_\vartheta_m) \), where \((\mathbb{D}, <)\) is a linear ordering and the \( P_i \)'s are unary relations.

[Segoufin & Toruńczyk, STACS'11]

- \( \text{PSPACE} \)-completeness with the concrete domains
  - \( \mathcal{D}_{Q^*} = (Q^*; <_{\text{pre}}, \leq_{\text{lex}}, =_\vartheta_1, \ldots, =_\vartheta_m) \).
  - \( \mathcal{D}_{[1,\alpha]^*} = ([1, \alpha]^*; <_{\text{pre}}, \leq_{\text{lex}}, =_\vartheta_1, \ldots, =_\vartheta_m), \ \alpha \geq 2. \)

[Kartzow & Weidner, arXiv 2015; Weidner, PhD 2016]

- \( \text{NLOGSPACE} \)-completeness with \((\{0,1\}^*, <_{\text{pre}}, <_{\text{suf}})\) restricted to a single variable.

[Peteler & Quaas, MFCS'22]
Tree Constraint Automata on Strings
(or how to recognize infinite data trees)

A accepts infinite trees \( t : [0, d - 1]^* \rightarrow (\Sigma \times (\Sigma^*)^\beta) \).

Transitions \((q, a, (\Theta_1, q_1), \ldots, (\Theta_d, q_d))\) put constraints on values of current node and children nodes.

Büchi and Rabin acceptance conditions.

Nonemptiness problem:

**Input:** Tree constraint automaton \( \mathcal{A} \).

**Question:** \( L(\mathcal{A}) \neq \emptyset \)?
From TSAT to Nonemptiness
(or how to apply the standard automata-based approach)

• Technically involved construction following a standard pattern.
  – \( C_0, \mathcal{T} \) in simple form positive instance of TSAT iff \( L(\mathcal{A}) \neq \emptyset \).
  – Locations are propositionally \( \mathcal{T} \)-consistent set of subconcepts.
  – Each role name has dedicated directions in \([1, d]\).
  – Constraints at the level of concepts translated at the level of transitions.

(see e.g. [Vardi & Wolper, JCSS 1986; Lutz, AI 2004; Baader, 2009])

• Postponing the actual problem to the nonemptiness problem.

• Great advantage: same construction for other concrete domains.
From String Constraints to Integer Constraints

• Intuition: encoding of prefix of strings $w$ and $w'$ by length of common prefix (which is a nonnegative integer).

• $\text{clen}(w, w')$: length of longest common prefix btw. $w$ and $w'$. E.g. $\text{clen}(aba, abbbab) = 2$.

• $w <_{\text{pre}} w'$ iff $\text{clen}(w, w') < \text{clen}(w', w') = |w'|$.

• There exist integer constraints that capture the string constraints.

   (interesting only if CCA on $\mathcal{N}$ can be handled)
Integer Constraints

- Properties (I)–(III) are “complete” to recover string values in a greedy way \((k = \text{card}(\Sigma))\). [Demri & Deters, JLC 2015]

\((\text{some analogy with the patchwork property})\)

(I) For \(w, w' \in \Sigma^*, |w| = \text{clen}(w, w) \geq \text{clen}(w, w')\).

(II) For all \(w_0, w_1, \ldots, w_k \in \Sigma^*\) such that
- \(\text{clen}(w_0, w_1) = \cdots = \text{clen}(w_0, w_k)\) and,
- for all \(i \in [0, k]\), \(\text{clen}(w_0, w_1) < |w_i|\),
there are \(i \neq j \in [1, k]\) such that \(\text{clen}(w_0, w_1) < \text{clen}(w_i, w_j)\).

(III) For all \(w_0, w_1, w_2 \in \Sigma^*\),
\(\text{clen}(w_0, w_1) < \text{clen}(w_1, w_2)\) implies \(\text{clen}(w_0, w_1) = \text{clen}(w_0, w_2)\).
From TCA on Strings to TCA on Natural Numbers

- TCA $A = (Q, \Sigma, d, \beta, Q_{in}, \delta, F)$ on $\mathcal{D}_\Sigma$ translated into $A' = (Q, \Sigma, d, \beta', Q_{in}, \delta', F)$ on $\mathcal{N}$.

- $L(A) \neq \emptyset$ iff $L(A') \neq \emptyset$.

- “Variables” in $A'$ of the form $\text{clen}(t_1, t_2)$ with

$$t_1, t_2 \in \{ x_i \mid i \in [1, \beta + \alpha] \} \cup \{ S^{-1}x_i \mid i \in [1, \beta + \alpha] \}$$

(constant strings $w_1, \ldots, w_\alpha$ in $A$)

- Translation of a constraint from $A$ is divided in three parts.
  1. Constraints related to (I)–(III), e.g. $\text{clen}(x, x) \geq \text{clen}(x, y)$.
  2. Constraints related to the understanding of $S^{-1}x$, e.g. $\text{clen}(S^{-1}x, S^{-1}y)' = \text{clen}(x, y)$.

$$\left( S \text{ clen}(t_1, t_2) \text{ also written } \text{clen}(t_1, t_2)' \right)$$

  3. Encoding of string constraints, e.g. $x_i' <_{\text{pre}} x_j$ becomes

$$\text{clen}(x_i, x_i)' = \text{clen}(x_i, S^{-1}x_j)' \land \text{clen}(x_i, x_i)' < \text{clen}(S^{-1}x_j, S^{-1}x_j)'$$
Complexity Analysis

- TCA $\mathbb{A} = (Q, \Sigma, d, \beta, Q_{in}, \delta, F)$ on $\mathcal{D}_\Sigma$ translated into $\mathbb{A}' = (Q, \Sigma, d, \beta', Q_{in}, \delta', F)$ on $\mathcal{N}$.

- $L(\mathbb{A}') \neq \emptyset$ checked in time

$$R_1 \left( \text{card}(Q) \times \text{card}(\delta') \times \text{MCS}(\mathbb{A}') \times \text{card}(\Sigma) \times R_2(\beta') \right)^{O(R_2(\beta') \times R_3(d))}$$

[Demri & Quaas, CONCUR'23]

- the $R_i$'s are polynomials,

- $\text{MCS}(\mathbb{A}')$: maximal size of a constraint in $\mathbb{A}'$,

- $\text{MCS}(\mathbb{A}')$ in $(\beta + \text{MCS}(\mathbb{A}) \times \text{card}(\delta) \times d)^{O(\text{card}(\Sigma)+3)}$,

- $\beta'$ polynomial in $\beta$ and in the number of constant strings in $\mathbb{A}$.
Concluding Remarks

\[
\text{TSAT}(\mathcal{ALCF}^P(\mathcal{D}_\Sigma)) \xrightarrow{} \text{TSAT}(\mathcal{ALCF}^P(\mathcal{D}_\Sigma)) \xrightarrow{} \text{NE(TCA}(\mathcal{D}_\Sigma)) \xrightarrow{} \text{NE(TCA}(\mathbb{N}))
\]

in simple form

- First steps towards taming description logics over strings.
- Automata-based approach with tree constraint automata.
- Reuse or adaptations of several results from literature with new insights to combine them.
- How to extend the results with suffix relation \(<_{\text{suf}}\), with infix relation or with regularity constraints?
- Other types of constraints on values, i.e. by considering different paths. See e.g. [Lutz, IJCAR’01; Figueira, ToCL 2012]