Register-Bounded Synthesis from Constraint LTL

Nino Dauvier
LIS, Aix Marseille Univ, CNRS, Marseille, France

Emmanuel Filiot
Université libre de Bruxelles

Pierre-Alain Reynier
LIS, Aix Marseille Univ, CNRS, Marseille, France

Abstract

We consider synthesis problems from logical specifications over infinite data domains, expressed in the logic *constraint LTL* (CLTL), which extends LTL with predicates over an infinite set of data values. We consider register-bounded synthesis, where the goal is to automatically generate, if it exists, a transducer with r registers that realizes a given CLTL formula, where r is also given as input. We prove that CLTL register-bounded synthesis is 2EXPTIME-C for various data domains such as any countable set with equality, $(\mathbb{Q}, <)$, and $(\mathbb{N}, <)$. For the latter domain, this contrasts with known undecidability results of (unbounded) register CLTL synthesis, by Bhaskar and Praveen. Lastly, we consider synthesis in a partial observation setting, by extending CLTL with invisible variables.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Modal and temporal logics

Keywords and phrases Synthesis, Data words, Constraint linear time logic, Register transducer

1 Introduction

Boolean reactive synthesis Program synthesis – the automatic generation of programs from high-level specifications – has emerged as a promising avenue to improve software reliability, by producing programs that are correct *by construction*.

A particularly rich and well-studied area within the field of program synthesis is *reactive synthesis* [2], which focuses on the design of algorithmic methods for the automatic construction of reactive systems, i.e. systems that maintain ongoing interactions with their environments. Over the past fifteen years, there has been remarkable progress in the synthesis of reactive systems modeled as finite state machines (Mealy machines), from specifications expressed in linear-time temporal logic (LTL). These developments have led to the creation of powerful tools and solvers, whose performance are continuously evaluated through the annual Reactive Synthesis Competition [14].

Beyond the Booleans Classical reactive synthesis methods focus on complex control aspects and typically assume that reactive systems process a finite set of Boolean signals, while ignoring *data*. More recent research have considered extensions of this purely Boolean setting to *data-aware* reactive systems. In particular, there is a line of work on the synthesis of infinite-state systems modeled as transducers with registers. In this setting, programs targeted by synthesis algorithms are modeled as a Mealy machine extended with registers that allow storing and comparing incoming data to produce data. In turn, the synthesis problem is parameterized by a *data domain*, i.e. a countable set of data values and a finite set of predicates over those data values.

A natural candidate for expressing temporal specifications over data domains is *constraint LTL* (CLTL), which extends LTL with constraints over data values [8]. CLTL formulas are built over a finite set of variables X, which is extended by considering for any variable $x \in X$, $k+1$ copies $x^{(0)}, \ldots, x^{(k)}$ of x. The intended meaning of $x^{(i)}$ is to denote the content of x in i steps. The syntax of CLTL formulas is defined as for LTL, except that atomic formulas are predicates over the (copied) variables. As an example, the CLTL formula $G(x^{(0)} = y^{(2)})$ is satisfied if, at any instant t, x holds the same data value as y at instant $t+2$. Over a set of data values $\mathbb{D}$, models of CLTL formulas are sequences of valuations $(X \to D)^\omega$. It is worth noting that the satisfiability of a CLTL formula depends on the data domain. For instance, $G(x^{(0)} > x^{(1)})$ is satisfiable in $\mathbb{Z}$ but not in $\mathbb{N}$. The formula $G(x^0 < x^{(1)} \leq y^{(0)} = y^{(1)})$ is satisfiable in $\mathbb{Q}$ but not in $\mathbb{Z}$.

In an attempt to classify data domains, the *completion property* has been considered in the literature [8]. For a data domain $\mathcal{D}$, it asks that for any satisfiable constraint C (defined as a conjunction of predicates over $\mathcal{D}$), any partial valuation satisfying a sub-constraint of C can be extended to a (total) valuation satisfying C. For example, the constraint $C_0 = x < y < z$ is satisfiable in $\mathbb{Q}$, and any valuation of x, z satisfying $x < z$ can be extended to a valuation of x, y, z satisfying C_0, e.g. by taking $y = (x+z)/2$. While any domain $(\mathbb{D}, =)$ and the domain $(\mathbb{Q}, <)$ satisfy the completion property, important data domains such as $(\mathbb{N}, <)$ and $(\mathbb{Z}, <)$ do not. As an example, the partial valuation $x = 0, z = 1$, which satisfies $x < z$, cannot be extended to a valuation that satisfies C_0.

While the *satisfiability* problem for CLTL (over various data domains) has been quite studied in the literature [4,8] (see also the survey [6] and the references therein), not much is known about the *synthesis* problem. In this context, the set of variables is further divided into variables $\mathbb{I}$ controlled by the environment (input variables) and variables $\mathbb{O}$ controlled by the system (output variables). The synthesis problem from specifications expressed in CLTL has been studied in [1]. It consists in deciding whether there exists a strategy that, at each step, reads data values of $\mathbb{I}$, and outputs data values for $\mathbb{O}$, resulting in an infinite data word over $\mathbb{I} \cup \mathbb{O}$ that satisfies the CLTL formula. It is shown that over data domains with the completion property, CLTL synthesis is decidable. However, synthesis is shown to be *undecidable* for $(\mathbb{Z}, <)$. To recover decidability, the authors of [1] consider some restriction, called single-sidedness, in which the power of the environment is restricted. In this paper, we take an orthogonal route to recover decidability.

Register-bounded reactive synthesis While [1] considers arbitrary strategies, we focus on strategies represented by *register transducers*. A register transducer successively receives as input a valuation $(\mathbb{I} \to \mathcal{D})$, compares it with the data values stored in its registers, assigns some of the input values to its registers (or none), and finally produces some output valuation $(\mathbb{O} \to \mathcal{D})$ by assigning to each variable in $\mathbb{O}$ the content of some register. These strategies are more amenable to implementations, as they are close to real programs. We consider the *register-bounded synthesis problem from* $CLTL(\mathcal{D})$, defined as the problem of deciding, given as input a CLTL formula Φ over $\mathcal{D}$ and some integer r, whether there exists a transducer with r registers which realizes Φ (and in that case output it). As we show, bounding the number of registers (but not the number of states) allows to recover decidability for CLTL synthesis over data domains such as $(\mathbb{Z}, <)$. Register-bounded synthesis is also desirable for synthesizing systems with a *minimal* number of registers. As registers represent some auxiliary memory, this is particularly relevant in contexts where resources are limited.

Contributions Our objective is to develop general techniques that allow us to show decidability results for *families* of data domains. To that end, we develop reductions from the register-bounded synthesis problem to a realizability problem over a finite alphabet. The key in this approach is the analysis of the set $\mathcal{S}_\mathcal{D}$ of infinite constraint sequences over $\mathcal{D}$ that are satisfiable. Our contributions are as follows.

1. We first show that if $\mathcal{S}_\mathcal{D}$ is an ω-regular language, then the register-bounded synthesis problem from $CLTL(\mathcal{D})$ is decidable (Theorem 10). In addition, we prove that if a data domain $\mathcal{D}$ meets the completion property, then $\mathcal{S}_\mathcal{D}$ is ω-regular. This allows us to prove that the register-bounded synthesis problem from CLTL is 2EXPTIME-C over any domain $(\mathbb{D}, =)$ for a countable set $\mathbb{D}$, and over the domain $(\mathbb{Q}, <)$ (Corollary 19).
2. Yet, $\mathcal{S}_{(\mathbb{Z},<)}$ is not ω-regular (see [9]), which reflects the undecidability result mentioned

earlier. As a remedy, we show that if $\mathcal{S}_\mathcal{D}$ can be over-approximated by an ω-regular language which coincides with $\mathcal{S}_\mathcal{D}$ on lasso words, then the register-bounded synthesis problem from $CLTL(\mathcal{D})$ is decidable (Theorem 22). We show that this is the case for numerous data domains including $(\mathbb{N}, <)$, $(\mathbb{Z}, <)$, $(\Sigma^*, \preceq_{pref})$ for Σ any finite alphabet and $\preceq_{pref}$ the prefix relation, as well as $(\mathbb{Z}^k, <_k)$ for any k and $<_k$ the partial order on tuples (pairwise comparison), using a reduction to [15]. This implies that the register-bounded synthesis problem from CLTL is 2ExpTime-c for all these data domains (Corollary 28).

3. Lastly, we consider a partial-observation generalization in which the CLTL formula has private and public input variables, but the system only has access to public ones. We show that our approach can be leveraged to this setting, yielding that the problem is 2ExpTime-c for any domain $(\mathbb{D}, =)$, and over the domain $(\mathbb{Q}, <)$ (Corollary 32).

Related work Synthesis problems of register transducers over data domains have already been considered in the literature, but mostly when the specification is already given as an automaton, and in particular a *register automaton*, see [10, 11, 16, 17]. Beyond specifications given by deterministic register automata, synthesis quickly turns into undecidability. That is the case, for instance, for specifications given by non-deterministic or universal register automata [10, 16], which are strictly more expressive than their deterministic restrictions. When considering the data domains $(\mathbb{Z}, <)$ and $(\mathbb{N}, <)$, synthesis is already undecidable for specifications given by deterministic register automata [9].

Register-bounded synthesis has been considered for specifications given as register automata [10, 15, 16]. In particular, for specifications given as universal register automata over $(\mathbb{Z}, =)$ and $(\mathbb{Z}, <)$, register-bounded synthesis has been shown to be decidable [11, 15, 16]. A legitimate question is whether register-bounded synthesis from CLTL could be reduced to register-bounded synthesis from register automata and directly apply the results of [11, 15, 16]. We do not know of any reduction that would preserve the number of registers needed to realize the specification. Indeed, even though CLTL formulas can be converted into deterministic register automata, this conversion is however modulo some model encoding: CLTL formulas are interpreted over $(X \to D)^\omega$ while register automata are executed on words in D^ω, and so tuples need to be encoded as chunks of $d = |X|$ consecutive data. Having access to d data at once allows for register succinctness. This is because register transducers in [11, 15, 16] have access to only one data at a time, while the register transducers of this paper receive as input a tuple of data, to match the semantics of CLTL formulas. For example, a CLTL specification might require to output an input value only if it appears twice in a tuple. To check the latter property, a register transducer in [11, 15, 16] would need $d-1$ additional registers.

On top of the related papers already cited before, we would like to mention the work on *constraint tree automata* over $(\mathbb{Z}, <)$, whose emptiness problem is proved to be decidable in [7], allowing us to prove the decidability of the branching-time logics *constraint* CTL and CTL* over data domain $(\mathbb{Z}, <)$. It is, however, not clear how register-bounded synthesis from constraint LTL could be reduced to the emptiness problem of such automata.

2 Constraint LTL and automata over data words

Finite and infinite words Given an alphabet Σ finite or not, let Σ^* (resp. Σ^ω) be the set of finite (resp. infinite) sequences, called words, over Σ. In this paper, we denote finite words as $\overline{u} = u_1 u_2 \dots u_n$ where $u_i \in \Sigma$ for all $i \in \{1, \dots, n\}$. We let $|\overline{u}| = n$ denote its length and for $1 \leq i \leq j \leq |\overline{u}|$, we let $\overline{u}_{[i,j]} = u_i u_{i+1} \dots u_j$. Similarly, if $\overline{u} = u_1 u_2 \dots$ is an infinite word, we let $\overline{u}_{[i,+\infty]} = u_i u_{i+1} \dots$ be the infinite suffix starting at position i.

Data We define a data domain $\mathcal{D}$ as a tuple $(\mathbb{D}, P)$, where $\mathbb{D}$ is an infinite set whose elements are called *data* and P is a finite set of predicates (relations over $\mathbb{D}$ of any arity). It is also required that the set of predicates always contains the equality predicate, in order to be able to distinguish data values. In this paper, we will mostly use the data domains $(\mathbb{D}, =)$ (where $\mathbb{D}$ is arbitrary), $(\mathbb{Q}, <)$ and $(\mathbb{N}, <)$. Note that $(\mathbb{N}, <)$ does not contain equality, but $x = y$ can be expressed as $\neg(x < y) \wedge \neg(y < x)$. In general, data domains are defined as interpretations of a set of predicate *symbols*, however to simplify the presentation, in this paper we confuse symbols with their (intuitive) semantics. Lastly we say that a data domain is *decidable* if its existential first-order theory is decidable.

Constraints Let V be a finite set of variables, and $\mathcal{D} = (\mathbb{D}, P)$ be a data domain. A *valuation* from V to $\mathbb{D}$ is a (total) mapping from V to $\mathbb{D}$. We denote by $Val_{V,\mathbb{D}}$ the set of valuations from V to $\mathbb{D}$. Given two valuations ν_1, ν_2 defined over two distinct subsets A_1, A_2 of V, we let $\nu_1 \uplus \nu_2$ be the valuation defined on $A_1 \cup A_2$ by $\nu_1 \uplus \nu_2(x) = \nu_i(x)$ if $x \in A_i$.

A literal ℓ over V is a term of the form $p(x_1, \ldots, x_k)$ or its negation $\neg p(x_1, \ldots, x_k)$, with $p \in P$ of arity k and $x_1, \ldots, x_k \in V$. It is satisfied by a valuation $w \in Val_{V,\mathbb{D}}$, written $w \models \ell$, if $(w(x_1), \ldots, w(x_k)) \in p$ if $\ell = p(x_1, \ldots, x_k)$ (and $(w(x_1), \ldots, w(x_k)) \notin p$ for a negative literal). A constraint C is a conjunction of literals. We often view C as the set of its conjuncts. We denote by $\mathcal{C}_V$ the set of constraints on variables in V. We say that a constraint $C \in \mathcal{C}_V$ is *satisfiable* if there exists a valuation $w \in Val_{V,\mathbb{D}}$ such that $w \models \ell$ for all literals $\ell \in C$. We write $w \models C$ when this holds.

A constraint C is *consistent* if it does not contain a literal and its negation. It is *maximally consistent* if it is consistent and for all predicates p of arity k and for all $(x_1, \ldots, x_k) \in V^k$, either $p(x_1, \ldots, x_k)$ or its negation occurs in C. For example over $(\mathbb{N}, <)$, the constraint $C_0 = (x < y) \wedge (y < z) \wedge (x < z) \wedge \neg(y < x) \wedge \neg(z < y) \wedge \neg(z < x)$ is not maximally consistent, but $C_1 = C_0 \wedge \neg(x < x) \wedge \neg(y < y) \wedge \neg(z < z)$ is. We denote by $\mathcal{MC}_V$ the set of maximally consistent constraints over V.

Completion property For $V' \subseteq V$, we define the restriction $C|_{V'}$ of a constraint C, as the subset of literals of C that use only variables in V'. A satisfiable constraint C is *completable* if for any $V' \subseteq V$, for all $w' \in Val_{V',\mathbb{D}}$ such that $w' \models C|_{V'}$, there exists $w \in Val_{V,\mathbb{D}}$ such that $w'(x) = w(x)$ for all $x \in V'$ and $w \models C$.

A data domain has *completion property* (we also say it is *completable*) if all satisfiable and maximally consistent constraints are completable. For example, $(\mathbb{N}, <)$ does not meet the completion property, if we take the constraint C_1 defined before, it is satisfiable and maximally consistent. Then we can build w' such that $w'(x) = 0$ and $w'(z) = 1$, it satisfies $C_1|_{\{x,z\}}$, but we cannot find $y \in \mathbb{N}$ such that $(x < y) \wedge (y < z)$. However:

▶ **Lemma 1** (Corollary 5.4 in [8]). *Data domains* $(\mathbb{D}, =)$ *and* $(\mathbb{Q}, <)$ *are completable.*

Data valuation words Let V be a finite set of variables and $\mathcal{D} = (\mathbb{D}, P)$ be a data domain, a *data valuation word* is a sequence $\overline{w} = w_1 w_2 \cdots \in (Val_{V,\mathbb{D}})^\omega$. Data valuation words are used to model the traces of the systems that we want to synthesize. As CLTL allows us to compare valuations that occur at different time points, we also introduce another notion of valuation word. Let $k \in \mathbb{N}$, and define the set of k-future variables as $V^{(k)} = \{x^{(i)} \mid x \in V, 0 \leq i \leq k\}$. A k-extended valuation word is a sequence $\overline{\nu} = \nu_1 \nu_2 \cdots \in (Val_{V^{(k)},\mathbb{D}})^\omega$. Intuitively, $\nu_i(x^{(j)})$ is intended to be the value of x at time $i + j$, and therefore it is required that $\nu_i(x^{(j)}) = \nu_{i+j}(x)$. This condition is enforced by the notion of compatible extended valuation words, defined as the sequences $\overline{\nu} \in (Val_{V^{(k)},\mathbb{D}})^\omega$ such that

$$\forall i > 0, \forall 0 < j \leq k, \forall x \in V, \nu_i(x^{(j)}) = \nu_{i+j}(x)$$

We define a bijection between valuation words and their compatible k-extended form, *via* the function $\textsc{extend}_k : (Val_{V,\mathbb{D}})^\omega \to (Val_{V^{(k)},\mathbb{D}})^\omega$ defined by $\textsc{extend}_k(\overline{w}) = \overline{\nu}$ where for all $j \geq 1$ and all $x^{(i)} \in V^{(k)}$, $\nu_j(x^{(i)}) = w_{j+i}(x)$. One can show that $\textsc{extend}_k$ is a bijection between valuation words and compatible k-extended valuation words, and we denote by COMPRESS its inverse. Last, we lift $\uplus$ from valuations to valuation words in the obvious way.

▶ **Example 2.** Let $V = \{x, y\}$ and $\mathcal{D} = (\mathbb{N}, <)$. We consider $\overline{w} \in (Val_{V,\mathbb{D}})^\omega$ given by $w_i(x) = i$ and $w_i(y) = i + 1$ for all $i \geq 1$. Then $\overline{\nu} = \text{EXTEND}_1(\overline{w})$ is given by $\nu_i(x^{(0)}) = i, \nu_i(x^{(1)}) = \nu_i(y^{(0)}) = i + 1, \nu_i(y^{(1)}) = i + 1$ for all $i \geq 1$.

Constraint words A *constraint word* is a sequence $\overline{c} = C_1C_2 \cdots \in (\mathcal{C}_V)^\omega$ of constraints over V. A valuation word $\overline{w} \in (Val_{V,\mathbb{D}})^\omega$ satisfies $\overline{c}$, written $\overline{w} \models \overline{c}$, if $\forall i \geq 1, w_i \models C_i$. We let $[\![\overline{c}]\!] = \{\overline{w} \in (Val_{V,\mathbb{D}})^\omega \mid \overline{w} \models \overline{c}\}$. Given $\overline{c'} \in (\mathcal{C}_V)^\omega$, we write $\overline{c} \subseteq \overline{c'}$ whenever for all $i \geq 1$, $C'_i \subseteq C_i$ (seen as sets of literals). In particular, $\overline{c} \subseteq \overline{c'}$ implies $[\![\overline{c'}]\!] \subseteq [\![\overline{c}]\!]$.

For clarity purposes, we write $\mathcal{C}_V^{(k)}$ the set of constraint over $V^{(k)}$ instead of $\mathcal{C}_{V^{(k)}}$, the same goes for $\mathcal{MC}_V^{(k)}$. Constraint words in $\mathcal{C}_V^{(k)}$ are said to be of *rank* k. The satisfiability notion is extended to constraint words $\overline{c} \in (\mathcal{C}_V^{(k)})^\omega$ of rank k via the function EXTEND_k. Formally, $\overline{w} \in (Val_{V,\mathbb{D}})^\omega$ satisfies $\overline{c}$ if $\text{EXTEND}_k(\overline{w})$ satisfies $\overline{c}$, and $[\![\overline{c}]\!] = \{\overline{w} \in (Val_{V,\mathbb{D}})^\omega \mid \overline{w} \models \overline{c}\}$.

CLTL Fix a data domain $\mathcal{D} = (\mathbb{D}, P)$. The logic $CLTL(\mathcal{D})$ is defined as the set of formulas of the following form. The atomic formulas of rank $k \geq 0$ are terms of the form $p(x_1^{(n_1)}, \ldots, x_\ell^{(n_\ell)})$, where $p \in P$ is a predicate of arity ℓ, and $0 \leq n_1, \ldots, n_\ell \leq k$ are integers. We denote this set by $Atom_k$. $CLTL(\mathcal{D})$-formulas of rank k are inductively defined as:

$$\Phi ::= \Phi_1 U \Phi_2 \mid \Phi_1 \Rightarrow \Phi_2 \mid \neg\Phi \mid X\Phi \mid a \in Atom_k.$$

A $CLTL(\mathcal{D})$-formula is a $CLTL(\mathcal{D})$-formula of rank k for some k. As usual, the Boolean connectives $\wedge$ and $\vee$ can be defined from $\Rightarrow$ and $\neg$, and we let $\top = a \vee \neg a$ (for some atomic formula a) and $\bot = \neg\top$. Moreover, as for LTL formulas, we define the temporal operators *globally* (G) and *eventually* (F) by $F\Phi = \top U \Phi$ and $G\Phi = \neg F \neg \Phi$. Finally, we may write x instead of $x^{(0)}$, and $CLTL$ instead of $CLTL(\mathcal{D})$ when $\mathcal{D}$ is clear from the context.

The *size* $|\Phi|$ of a formula Φ is defined as the number of symbols in Φ, where any occurrence of a variable $x^{(i)}$ is considered to be of size i.

Semantics We define the semantics of CLTL by induction on formulas. Let $\overline{\nu} \in (Val_{V^{(k)},\mathbb{D}})^\omega$ and Φ a CLTL formula over $V^{(k)}$. We say that $\overline{\nu}$ satisfies Φ, written $\overline{\nu} \models \Phi$, if the following holds (inductively):

- if $\Phi = p(\vec{x})$ then $\nu_1 \models p(\vec{x})$
- if $\Phi = \neg\Psi$ then $\overline{\nu} \not\models \Psi$
- if $\Phi = \Psi_1 \Rightarrow \Psi_2$ then $\overline{\nu} \not\models \Psi_1$ or $\overline{\nu} \models \Psi_2$
- if $\Phi = X\Psi$ then $\nu_{[2,+\infty]} \models \Psi$
- if $\Phi = \Psi_1 U \Psi_2$ then $\exists n \geq 1$ such that $\forall 1 \leq i < n$, $\overline{\nu}_{[i,+\infty]} \models \Psi_1$ and $\overline{\nu}_{[n,+\infty]} \models \Psi_2$

Given a valuation word $\overline{w} \in (Val_{V,\mathbb{D}})^\omega$, we say that $\overline{w}$ *satisfies* Φ, also written $\overline{w} \models \Phi$, if $\text{EXTEND}_k(\overline{w}) \models \Phi$ holds. We let $L(\Phi) = \{\overline{w} \in (Val_{V,\mathbb{D}})^\omega \mid \overline{w} \models \Phi\}$.

▶ **Example 3** (Example 2 continued)**.** Consider $\Phi = G\left(x \leq y \wedge y \leq y^{(1)}\right) \in CLTL(\mathcal{D})$. One can easily check that $\overline{w} \in [\![\Phi]\!]$ holds.

▶ **Definition 4.** *A constraint automaton (CA) $\mathcal{A}$ over $\mathcal{D}$ is a tuple (Q, Δ, I, χ, k) where Q is a finite set of states, $k \in \mathbb{N}$ is the maximal rank for the constraints, $\Delta \subseteq Q \times \mathcal{C}_V^{(k)} \times Q$ is a transition relation, $I \subseteq Q$ is a set of initial states and $\chi \in Val_{Q,\mathbb{N}}$ a coloring of the states.*

A run of a CA over a valuation word $\overline{w} \in (Val_{V,\mathbb{D}})^\omega$ is a sequence of transitions $\overline{t} \in \Delta^\omega$ such that for all $i \geq 1$, $t_i = (q_{i-1}, C_i, q_i)$, with $w_i \models C_i$ and $q_0 \in I$. A run is *accepting* if the highest color seen infinitely often along the run is even. Under the non-deterministic semantics, the word $\overline{w}$ is accepted if there exists an accepting run. This yields the model of non-deterministic parity CA. The number of priorities of $\mathcal{A}$ is the size of $\chi(Q)$. We will also consider a universal semantics: in this case, a word $\overline{w}$ is accepted if *all* runs on $\overline{w}$ are accepting. An automaton is *deterministic* if for any two transitions of the form (q, C_1, p_1) and (q, C_2, p_2) such that $p_1 \neq p_2$, for all valuation $w \in Val_{V,\mathbb{D}}$, either $w \not\models C_1$ or $w \not\models C_2$. We also consider subclasses of CA: we say that an automaton has a Büchi acceptance condition (resp co-Büchi) if all its states have color 1 or 2, that is, $\chi(Q) \subseteq \{1, 2\}$ (resp $\chi(Q) \subseteq \{0, 1\}$). It has a reachability (resp safety) acceptance condition if it has a Büchi (resp co-Büchi) acceptance condition and there exists no transition from a state colored 2 to a 1 (resp from 1 to 0). Last, the language of a CA $\mathcal{A}$ is the set of accepted words, and is denoted $L(\mathcal{A})$.

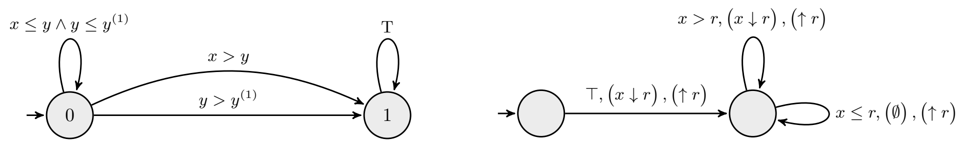
We call *syntactic automaton* $\mathcal{A}_{stx}$ the automaton $\mathcal{A}$ seen as a finite automaton over the finite alphabet $\mathcal{C}_V^{(k)}$ of constraints of rank k and let $L_{stx}(\mathcal{A}) = L(\mathcal{A}_{stx})$ denote its language.

From CLTL to constraint automata It is well-known any LTL sentence can be converted into a Büchi automaton. This result carries over to CLTL and constraint automata.

▶ **Proposition 5.** *Let Φ be a CLTL formula. One can construct in* ExpTime *a universal co-Büchi CA $\mathcal{A}$ such that $L(\Phi) = L(\mathcal{A})$ whose size is exponential in the size of Φ.*

Proof. The logic CLTL can be seen as LTL over the finite alphabet $\mathcal{C}_V^{(k)}$. As such we can use the classical exponential procedure that converts any LTL formula into an equivalent non-deterministic Büchi automaton, to build a non-deterministic Büchi constraint automaton equivalent to the CLTL formula (see Section 3.2 in [6]). The result follows by applying the latter construction on the negation of the formula and interpreting the resulting automaton with a universal co-Büchi condition. ◀

▶ **Example 6** (Example 3 continued)**.** A constraint automaton equivalent to the CLTL formula considered above is depicted in Figure 1a.



(a) A universal co-Büchi CA. The colors are indicated in the states.

(b) A register transducer

Figure 1 A constraint automaton and a register transducer.

3 Register-bounded synthesis problem from CLTL

3.1 Synthesis problem over finite alphabets

We first recall the synthesis problem for ω-regular specifications over a finite alphabet. In this setting, the goal is to synthesize (if it exists) a finite transducer over a finite input alphabet Σ_{in} and a finite output alphabet Σ_{out} that realizes a specification $S \subseteq (\Sigma_{in}\Sigma_{out})^\omega$, given, for example, as a non-deterministic Büchi automaton. Let us formally define this problem. A *finite transducer* (FT for short) is a tuple $T = (\Sigma_{in}, \Sigma_{out}, Q, q_0, \delta)$ such that Q is a finite set of states with initial state $q_0 \in Q$, and $\delta : Q \times \Sigma_{in} \to \Sigma_{out} \times Q$ is a (total) transition function. The semantics of T is a language, denoted $L(T) \subseteq (\Sigma_{in}\Sigma_{out})^\omega$, defined as the set of words $i_1 o_1 i_2 o_2 \cdots \in (\Sigma_{in}\Sigma_{out})^\omega$ such that there exists a sequence of states $p_0 p_1 \cdots \in Q^\omega$ with $p_0 = q_0$ and for all $j \geq 1$, $\delta(q_j, i_j) = (o_j, q_{j+1})$.

A specification $S \subseteq (\Sigma_{in}\Sigma_{out})^\omega$ is said to be *realizable* if there exists a finite transducer T such that $L(T) \subseteq S$. Büchi-Landweber's Theorem states that when S is ω-regular, the latter problem is decidable [3]. More precisely, this problem can be solved in ExpTime when

S is given as a universal co-büchi automaton [19], and is 2ExpTime-c when S is given as an LTL formula over some sets of input and output atomic propositions [20].

3.2 Register transducers over data words

We now want to extend this to infinite alphabets and specifications given as CLTL formulas. We need to adapt the model of implementation, *i.e.* transducers. To this end, we first extend the notion of finite transducer to transducers over infinite alphabets that act over a finite set of registers. Let $\mathcal{D} = (\mathbb{D}, P)$ be a data domain.

Action words Let $V = \mathbb{I} \uplus \mathbb{O}$ be a finite set of variables partitioned into $\mathbb{I}$ (input variables) and $\mathbb{O}$ (output variables). They are also called system and environment variables in the context of synthesis. Let R be a finite set of elements called registers. An *action* over R and V is a tuple in $Test_{\mathbb{I},R} \times Assign_{\mathbb{I},R} \times Output_{\mathbb{O},R}$ where:

- $Test_{\mathbb{I},R} = \mathcal{MC}_{\mathbb{I}\cup R}$ is the set of maximally consistent constraints over $\mathbb{I} \cup R$.
- $Assign_{\mathbb{I},R}$ is the set of partial functions $\rho^{ass} : R \hookrightarrow \mathbb{I}$ from R to $\mathbb{I}$.
- $Output_{\mathbb{O},R} = Val_{\mathbb{O},R}$ is the set of (total) functions $\rho^{out} : \mathbb{O} \to R$ from $\mathbb{O}$ to R.

Assignment are to be understood as a way to store data that we receive in input to compare them later in tests or output them. We denote by $Act_{\mathbb{I},\mathbb{O},R}$ the set of actions. An *action word* is an infinite word $\overline{a} \in (Act_{\mathbb{I},\mathbb{O},R})^\omega$. We denote by $(Act_{\mathbb{I},\mathbb{O},R})^\omega$ the set of action words.

The semantics of an action word $\overline{a} = (C_1, \rho_1^{ass}, \rho_1^{out})(C_2, \rho_1^{ass}, \rho_1^{out})\dots$ is a language $[\![\overline{a}]\!] \subseteq (Val_{V,\mathbb{D}})^\omega$ that we now define. Assume $\overline{w} = (w_1^V = w_1^{\mathbb{I}} \uplus w_1^{\mathbb{O}})(w_2^V = w_2^{\mathbb{I}} \uplus w_2^{\mathbb{O}})\dots$ be a sequence of valuations. We "execute" the action word $\overline{a}$ on $\overline{w}$. Registers are used to store data from the input valuations in $\overline{w}$, and the actions on registers (tests and assignments) are dictated by $\overline{a}$. Let $d_0 \in \mathbb{D}$ be some data value. Registers in R are all initialized with the value d_0 (valuation w_1^R). Then, test C_1 is performed on $w_1^{\mathbb{I}} \cup w_1^R$. If it succeeds, registers are updated according to ρ_1^{ass} (giving valuation w_2^R), and the valuation $w_1^{\mathbb{O}} = w_2^R \circ \rho_1^{out}$ is output. The execution proceeds with the new action $(C_2, \rho_2^{ass}, \rho_2^{out})$ and register valuation w_2^R. This is formally captured by the following compatibility notion.

We say that a valuation word $\overline{w} = (w_1 = w_1^{\mathbb{I}} \uplus w_1^{\mathbb{O}})(w_2 = w_2^{\mathbb{I}} \uplus w_2^{\mathbb{O}})\dots \in (Val_{V,\mathbb{D}})^\omega$ is *compatible with an action word* $\overline{a} \in (Act_{\mathbb{I},\mathbb{O},R})^\omega$ if there exists a register valuation word $\overline{w^R} = w_1^R w_2^R \dots \in (Val_{R,\mathbb{D}})^\omega$ such that $w_1^R(r) = d_0$ for all $r \in R$ and for all $i \in \mathbb{N}_{>0}$:

- $w_i^{\mathbb{I}} \uplus w_i^R \models C_i$;
- $w_{i+1}^R(r) = w_i^{\mathbb{I}} \circ \rho_i^{ass}(r)$ if $\rho_i^{ass}(r)$ is defined, otherwise $w_{i+1}^R(r) = w_i^R(r)$;
- $w_i^{\mathbb{O}} = w_{i+1}^R \circ \rho_i^{out}$.

The *language of an action word* $\overline{a}$ is the set $[\![\overline{a}]\!] \subseteq (Val_{V,\mathbb{D}})^\omega$ of valuation words compatible with $\overline{a}$. We write $\overline{w} \models \overline{a}$ whenever $\overline{w} \in [\![\overline{a}]\!]$. It is worth observing that if $\overline{w} \in [\![\overline{a}]\!]$, then the associated register valuation word $\overline{w^R}$ is unique and depends on d_0.

In order to remove this dependency regarding d_0, which has been arbitrarily chosen, we consider the ω-regular language $\mathsf{AW}^\omega_{\mathbb{I},\mathbb{O},R} \subseteq (Act_{\mathbb{I},\mathbb{O},R})^\omega$ composed of action words that only test or output a register once it has been assigned with some input data value. Formally, we relax the definition of action words by also considering tests that are not maximally consistent. Then, given $\overline{a} \in \mathsf{AW}^\omega_{\mathbb{I},\mathbb{O},R}$, the language $[\![\overline{a}]\!]$ does not depend on d_0.

▶ **Definition 7** (Register transducer). *A register transducer (RT for short) over $\mathcal{D}$ is a tuple $\mathbb{T} = (Q, q_0, R, V, \delta)$ where:*

- *R is a finite set of registers*
- *$V = \mathbb{I} \uplus \mathbb{O}$ is a finite set of variables partitioned into input variables and output variables such that $V \cap R = \varnothing$*
- *$(Test_{\mathbb{I},R}, Assign_{\mathbb{I},R} \times Output_{\mathbb{O},R}, Q, q_0, \delta)$ is a finite transducer, denoted $\mathbb{T}_{stx}$.*

In order to rule out transducers that could test a register before it has been assigned, we will only consider *well-behaved* transducers that satisfy the following property. W.l.o.g., by enriching states, we assume that there exists a mapping $\chi : Q \to 2^R$ that indicates the registers that have been assigned. Then we assume that for every pair of transitions of the form $(p, C_1, (\rho_1^{ass}, \rho_1^{out}), q_1)$, $(p, C_2, (\rho_2^{ass}, \rho_2^{out}), q_2)$, if $C_1|_{\mathbb{I}\cup\chi(p)} = C_2|_{\mathbb{I}\cup\chi(p)}$, then $\rho_1^{ass} = \rho_2^{ass}$, $\rho_1^{out} = \rho_2^{out}$ and $q_1 = q_2$. This states that if two transitions perform the same test on the input variables and registers that have been assigned, then the two transitions lead to the same state and realize the same assignments and outputs. This way, we can merge transitions, resulting in a transducer whose tests are maximally consistent constraints over registers that have already been assigned. We can then show that $L(\mathbb{T}_{stx}) \subseteq \mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$.

We define two semantics for T. The first is more syntactic and is useful to define finite abstractions of languages defined by register transducers, and is simply defined as $L_{stx}(\mathbb{T}) = L(\mathbb{T}_{stx})$. The second one, over valuations of V, is defined as $L(\mathbb{T}) = \bigcup_{\overline{a}\in L_{stx}(\mathbb{T})} [\![\overline{a}]\!]$.

3.3 Synthesis problem over data domains

When moving from a finite alphabet to data words over some data domain $\mathcal{D}$, we lift specifications from ω-regular languages over a finite alphabet to languages $L \subseteq (Val_{V,\mathbb{D}})^\omega$, for some set of variables V. Such a specification can, of course, be described by a constraint automaton, as well as by a CLTL formula, which is the purpose of this work. Regarding implementations, we consider well-behaved register transducers, leading to the following problem:

Register-bounded Synthesis Problem from CLTL($\mathcal{D}$)
Input: A CLTL($\mathcal{D}$) formula Φ over $V = \mathbb{I} \uplus \mathbb{O}$ and an integer r
Output: A well-behaved register transducer T over V and with r registers, if it exists, which *realizes* Φ, i.e. such that $L(T) \subseteq L(\Phi)$.

The decision problem associated with the synthesis problem is called *the realizability problem*, and only asks whether such a transducer T exists. In this paper, when stating hardness results with respect to the complexity of the synthesis problem, they refer to the realizability problem. Moreover, when stating upper-bounds, they also include the cost of constructing a solution (a register transducer).

We assume that r is given in unary. This is reasonable as the expected register transducer has r registers, which means that the configuration of the registers is already of size r.

▶ **Example 8** (Example 3 continued)**.** We consider again $\Phi = G\left(x \leq y \wedge y \leq y^{(1)}\right)$ with $\mathbb{I} = \{x\}$ and $\mathbb{O} = \{y\}$. In our context, since RT can only output data they received before, a transducer realizing this specification must output the largest input seen so far. The (well-behaved) RT with a single register depicted in Figure 1b realizes this specification. The transitions are to be read as follows: $x \downarrow r$ corresponds to an assignment (the valuation of x in stored in register r) and $\uparrow r$ corresponds to an output.

4 Data domains with ω-regular satisfiability

Let $\mathcal{D}$ be a data domain. For all integer $k \geq 1$ and set of variables X, let

$$\mathsf{SAT}_{k,X} = \{\overline{c'} \in (\mathcal{C}_X^{(k)})^\omega \mid \overline{c'} \text{ is a word of maximally consistent constraints and } [\![\overline{c'}]\!] \neq \emptyset\}$$

be the set of satisfiable constraint words in $\mathcal{D}$.

▶ **Definition 9.** *We say that $\mathcal{D}$ has* effective ω-regular satisfiability *if, for every k, X, one can construct an automaton recognizing* $\mathsf{SAT}_{k,X}$.

In this section, we will show the following theorem.

▶ **Theorem 10.** *Let $\mathcal{D}$ be a data domain with effective ω-regular satisfiability. Then register-bounded synthesis from specifications expressed as universal co-Büchi CA over $\mathcal{D}$ is decidable.*

The main idea is to reduce the problem to a synthesis problem for an ω-regular specification over a finite alphabet, using some sound and complete finite abstraction stated in Lemma 13. Since the goal is to synthesize a register transducer, the finite alphabet is the set of actions $Act_{\mathbb{I},\mathbb{O},R}$ of the transducer, which is partitioned into input actions (constraints over $\mathbb{I}$ and R) and output actions (register assignments and output function). The next two lemmas are technical results towards proving Lemma 13. From now on, we fix $\mathcal{D}$ some data domain.

From action words to constraint words First, in order to compare actions of register transducers and constraints of constraint automata, sequences of actions are canonically mapped to sequences of constraints, via a mapping $\mathsf{cstr} : (Act_{\mathbb{I},\mathbb{O},R})^\omega \to (\mathcal{C}^{(1)}_{V\cup R})^\omega$. Remember that an action is a triple $(C, \rho^{ass}, \rho^{out})$ where C is a constraint over $\mathbb{I} \cup R$, $\rho^{ass} : R \hookrightarrow \mathbb{I}$ is a partial function and $\rho^{out} : \mathbb{O} \to R$ a function. The latter two assignments induce constraints between V and R. It is also worth noting that the register valuation at step i is the one used for the test, while the one at step $i+1$ is the one obtained after the assignment, which is used for producing the output. For example, the assignment ρ^{ass} implies that the *next* content of register r is equal to the value of variable $\rho^{ass}(r)$. Similarly, any output variable $y \in \mathbb{O}$ holds a value equal to the *next* content of register $\rho^{out}(y)$. Formally, for all $\overline{a} = (C_1, \rho^{ass}_1, \rho^{out}_1)\ldots$, we let $\mathsf{cstr}(\overline{a}) = C'_1 C'_2 \ldots$ where:

$$C'_i = C_i \wedge \bigwedge_{\substack{r \in R \text{ s.t.} \\ \rho^{ass}_i(r) \text{ defined}}} (r^{(1)} = \rho^{ass}_i(r)) \wedge \bigwedge_{\substack{r \in R \text{ s.t.} \\ \rho^{ass}_i(r) \text{ not defined}}} (r^{(1)} = r) \wedge \bigwedge_{y \in \mathbb{O}} (y = (\rho^{out}_i(y))^{(1)})$$

The latter transformation is correct in the following sense, which is based on the previous observation that states that when $\overline{a} \in \mathsf{AW}^\omega_{\mathbb{I},\mathbb{O},R}$, the language $[\![\overline{a}]\!]$ does not depend on d_0:

▶ **Lemma 11.** *Let $\overline{a} \in \mathsf{AW}^\omega_{\mathbb{I},\mathbb{O},R}$ then $[\![\overline{a}]\!] = [\![\mathsf{cstr}(\overline{a})]\!]|_V$.*

Since a specification has constraints over $V^{(k)}$, and a register transducer expresses properties of action words, which themselves hold constraints over $V = \mathbb{I} \uplus \mathbb{O}$ and R, one needs a mechanism to synchronize these two types of constraints. This is done via the following function called *extension*, denoted $\mathsf{join} : (Act_{\mathbb{I},\mathbb{O},R})^\omega \times (\mathcal{C}^{(k)}_V)^\omega \to \mathcal{MC}^{(k)}_{V\cup R}$, defined for any $\overline{a} \in (Act_{\mathbb{I},\mathbb{O},R})^\omega$ and $\overline{c} \in (\mathcal{C}^{(k)}_V)^\omega$ as

$$\mathsf{join}(\overline{a}, \overline{c}) = \{\overline{c'} \in \mathcal{MC}^{(k)}_{V\cup R} \mid \mathsf{cstr}(\overline{a}) \subseteq \overline{c'} \text{ and } \overline{c} \subseteq \overline{c'}\}$$

Valuations satisfying constraint words in $\mathsf{join}(\overline{a}, \overline{c})$, when restricted to variables in V, satisfy both $\overline{a}$ and $\overline{c}$. Using Lemma 11, we prove:

▶ **Lemma 12** (Adequation). *Let $\overline{a} \in \mathsf{AW}^\omega_{\mathbb{I},\mathbb{O},R}$, $\overline{c} \in (\mathcal{C}^{(k)}_V)^\omega$, $\overline{w} \in Val_{V\cup R,\mathbb{D}}$ and $\overline{c'} \in (\mathcal{MC}^{(k)}_{V\cup R})^\omega$ such that $\overline{w} \models \overline{c'}$. Then $\overline{c'} \in \mathsf{join}(\overline{a}, \overline{c})$ iff $\overline{w}|_V \models \overline{a}$ and $\overline{w}|_V \models \overline{c}$.*

The next lemma formalizes the reduction of synthesis from infinite to finite alphabets.

▶ **Lemma 13** (Transfer Lemma). *Let $\mathcal{A}$ a universal co-Büchi CA and R a set of registers. Let*

$$\mathsf{W}_{\mathcal{A},R} = \{\overline{a} \in \mathsf{AW}^\omega_{\mathbb{I},\mathbb{O},R} \mid \forall \overline{c} \in (\mathcal{C}^{(k)}_V)^\omega, (\exists \overline{c'} \in \mathsf{join}(\overline{a}, \overline{c}), \overline{c'} \in \mathsf{SAT}_{k,V\cup R}) \Rightarrow \overline{c} \in L_{stx}(\mathcal{A})\}$$

$L(\mathcal{A})$ is realizable by an RT with $|R|$ registers iff $\mathsf{W}_{\mathcal{A},R}$ is realizable by an FT.

Proof. $\Rightarrow$ Suppose $\mathcal{A}$ is realized by some well-behaved RT $\mathbb{T}$, i.e. $L(\mathbb{T}) \subseteq L(\mathcal{A})$. We show that $\mathbb{T}_{stx}$ realizes $\mathsf{W}_{\mathcal{A},R}$, i.e., $L(\mathbb{T}_{stx}) \subseteq \mathsf{W}_{\mathcal{A},R}$. Let $\overline{a} \in L(\mathbb{T}_{stx})$. Let $\overline{c} \in (\mathcal{C}^{(k)}_V)^\omega$ such that

$\exists \overline{c'} \in \mathsf{join}(\overline{a}, \overline{c})$ and $\overline{c'} \in \mathsf{SAT}_{k,V\cup R}$. By definition of $\mathsf{SAT}_{k,V\cup R}$, there exists $\overline{w} \in (Val_{V\cup R,\mathbb{D}})^\omega$, such that $\overline{w} \models \overline{c'}$. By Lemma 12, $\overline{w}|_V \models \overline{a}$ and $\overline{w}|_V \models \overline{c}$. As $\overline{a} \in L(\mathbb{T}_{stx})$ and $\overline{w}|_V \models \overline{a}$, we get $\overline{w}|_V \in L(\mathbb{T})$, and therefore $\overline{w}|_V \in L(\mathcal{A})$. Let ρ be a run of $\mathcal{A}_{stx}$ on $\overline{c}$. Since $\overline{w}|_V \models \overline{c}$, we have that ρ is a run of $\mathcal{A}$ on $\overline{w}|_V$. From $\overline{w}|_V \in L(\mathcal{A})$ we find that ρ is accepting. Since this is true for all runs ρ and $\mathcal{A}$ has a universal acceptance condition, we finally get $\overline{c} \in L(\mathcal{A}_{stx})$. So, $\overline{a} \in \mathsf{W}_{\mathcal{A},R}$, by definition of $\mathsf{W}_{\mathcal{A},R}$. Finally, we have shown that $L(\mathbb{T}_{stx}) \subseteq \mathsf{W}_{\mathcal{A},R}$.

$\Leftarrow$ Suppose $\mathsf{W}_{\mathcal{A},R}$ is realized by a finite transducer $\mathbb{T}_f = (Q, I, \Sigma_i, \Sigma_o, \delta)$ where $\Sigma_i = Test_{\mathbb{I},R}$, $\Sigma_o = Assign_{\mathbb{I},R} \times Output_{\mathbb{O},R}$ and $\delta : Q \times Test_{\mathbb{I},R} \to Assign_{\mathbb{I},R} \times Output_{\mathbb{O},R} \times Q$. We have $L(\mathbb{T}_f) \subseteq \mathsf{W}_{\mathcal{A},R}$. The finite transducer $\mathbb{T}_f$ can be seen as an R-register transducer $\mathbb{T}$ over $\mathcal{D}$, i.e. such that $\mathbb{T}_{stx} = \mathbb{T}_f$. In addition, as we have $\mathsf{W}_{\mathcal{A},R} \subseteq \mathsf{AW}^\omega_{\mathbb{I},\mathbb{O},R}$, $\mathbb{T}$ is well-behaved. We show that $\mathbb{T}$ realizes $\mathcal{A}$, i.e. $L(\mathbb{T}) \subseteq L(\mathcal{A})$.

Let $\overline{w} \in L(\mathbb{T})$. The run of $\mathbb{T}$ on $\overline{w}$ induces an action word that we write $\overline{a}$, as well as a word of valuations $\overline{w'} \in (Val_{V\cup R,\mathbb{D}})^\omega$ such that $\overline{w'}|_V = \overline{w}$. We have $\overline{w} \models \overline{a}$ by semantics of register transducer. In particular $\overline{a} \in L(\mathbb{T}_f)$. Remember that we want to show $\overline{w} \in L(\mathcal{A})$. Let ρ be a run of $\mathcal{A}$ on $\overline{w}$. We let $\overline{c}$ be the constraint word induced by ρ, by semantics of constraint word and automata $\overline{w} \models \overline{c}$. Let $\overline{c'} \in \mathcal{MC}^{(k)}_{V\cup R}$ the maximally consistent constraint word such that $\overline{w'} \models \overline{c'}$ (it is obtained by choosing for every literal and its negation the one that $\overline{w}$ satisfies). As both $\overline{w} \models \overline{a}$ and $\overline{w} \models \overline{c}$, by Lemma 12 we have $\overline{c'} \in \mathsf{join}(\overline{a}, \overline{c})$. Moreover, since $\overline{w'} \models \overline{c'}$, we have $\overline{c'} \in \mathsf{SAT}_{k,V\cup R}$. As $\overline{a} \in L(\mathbb{T}_f)$, so $\overline{a} \in \mathsf{W}_{\mathcal{A},R}$ by hypothesis. By definition of $\mathsf{W}_{\mathcal{A},R}$, we have $\overline{c} \in L(\mathcal{A}_{stx})$. By definition of $\overline{c}$ and since $\mathcal{A}_{stx}$ is universal, ρ is accepting. As this is true for all runs ρ of $\mathcal{A}$ on $\overline{w}$, we get $\overline{w} \in L(\mathcal{A})$. ◀

The next lemma states that $\mathsf{W}_{\mathcal{A},R}$ is ω-regular whenever $\mathcal{D}$ has regular satisfiability and establishes some complexity bounds on the size of the representation of $\mathsf{W}_{\mathcal{A},R}$.

▶ **Lemma 14.** *Let $\mathcal{A}$ be a universal co-Büchi CA with n states. If $\mathcal{D}$ has effective ω-regular satisfiability, then $\mathsf{W}_{\mathcal{A},R}$ is effectively ω-regular: given $k \in \mathbb{N}$ and registers R, if $\mathsf{SAT}_{k,V\cup R}$ is recognizable by a non-deterministic parity automaton with n_s states and c_s colors, then $\mathsf{W}_{\mathcal{A},R}$ is accepted by a universal co-Büchi automaton with $O(n_s.c_s.n.2^{|R|})$ states.*

Proof sketch. We first consider the following definitions/equalities:

$$\begin{array}{lcl}
\mathsf{A} & = & (Act_{\mathbb{I},\mathbb{O},R})^\omega \times (\mathcal{C}^{(k)}_V)^\omega \times (\mathcal{MC}^{(k)}_{V\cup R})^\omega \\
L_{\mathsf{join}} & = & \{(\overline{a}, \overline{c}, \overline{c'}) \in \mathsf{A} \mid \overline{c'} \in \mathsf{join}(\overline{a}, \overline{c})\} \\
\mathsf{W}' & = & \{(\overline{a}, \overline{c}, \overline{c'}) \in \mathsf{A} \mid (\overline{a}, \overline{c}, \overline{c'}) \notin L_{\mathsf{join}} \vee \overline{c'} \notin \mathsf{SAT}_{k,V\cup R} \vee \overline{c} \in L(\mathcal{A}_{stx})\} \\
\mathsf{W}_{\mathcal{A},R} & = & \{\overline{a} \in \mathsf{AW}^\omega_{\mathbb{I},\mathbb{O},R} \mid \forall \overline{c} \in (\mathcal{C}^{(k)}_V)^\omega, \forall \overline{c'} \in (\mathcal{MC}^{(k)}_{V\cup R})^\omega, (\overline{a}, \overline{c}, \overline{c'}) \in \mathsf{W}'\}
\end{array}$$

As a consequence, a universal co-Büchi automaton for $\mathsf{W}_{\mathcal{A},R}$ can be derived from one for W'. Let us explain how we build it. First, ensuring that an action word is in $\mathsf{AW}^\omega_{\mathbb{I},\mathbb{O},R}$ can be done by a deterministic automaton with $O(2^{|R|})$ states. Then, L_{join} is recognizable by a deterministic (safety) automaton with two states. Then, the *complement* of W' roughly corresponds to the intersection of L_{join} with $\mathsf{SAT}_{k,V\cup R}$ and with the complement of $L(\mathcal{A}_{stx})$. The non-deterministic parity automaton for $\mathsf{SAT}_{k,V\cup R}$ can be translated into a non-deterministic Büchi one in polynomial time. In addition, as $\mathcal{A}$ is a universal co-Büchi CA, the complement of $L(\mathcal{A}_{stx})$ is also recognized by a non-deterministic Büchi automaton. Last, the intersection of two non-deterministic Büchi automata is known to be a non-deterministic Büchi automaton of polynomial size. ◀

Proof sketch of Theorem 10 The Transfer Lemma (Lemma 13) reduces the register-bounded synthesis problem to a synthesis problem over a finite alphabet, whose specification is ω-regular by Lemma 14. This problem is decidable by Büchi-Landweber's Theorem.

Data domains with the completion property Theorem 10 is established for data domains with effective ω-regular satisfiability. We prove that any data domain satisfying the completion property has ω-regular satisfiability. Intuitively, we build a safety automaton that stores the last constraint read and checks that two consecutive constraints are compatible.

▶ **Lemma 15.** *Let $\mathcal{D}$ be a decidable completable data domain (resp. decidable in* ExpTime*), then $\mathcal{D}$ has ω-regular satisfiability (resp. a deterministic parity automaton recognizing $\mathsf{SAT}_{k,X}$ can be constructed that has $exp(k.|X|)$ states and 2 priorities.*

▶ Remark 16. In the statement, we require that the existential first-order theory of $\mathcal{D}$ is decidable. In fact we only require decidability of the satisfiability of conjunctions of constraints and not any formula, this problem is generally easier.

We can now state the main result of this section:

▶ **Theorem 17.** *Let $\mathcal{D}$ be a decidable completable data domain (resp. decidable in* ExpTime*). Then register-bounded synthesis from CLTL($\mathcal{D}$) is decidable (resp. 2*ExpTime*-c). It is 2*ExpTime*-hard for any fixed number of registers $r \geq 2$.*

Proof sketch. To prove the upper bound, we first build from a CLTL formula Φ an equivalent CA $\mathcal{A}$, using Proposition 5. Then, using Lemmas 13, 14, 15, realisability of Φ reduces to that of $\mathsf{W}_{\mathcal{A},R}$, which can be recognized by a universal co-Büchi automaton whose size is exponential in $|\Phi|$, k and $|R|$. Last, realizability for specifications expressed by universal co-Büchi automata can be decided in ExpTime [19].

For the lower bound, we reduce the problem of LTL synthesis (over finite alphabets), which is already 2ExpTime-c [20], to bounded register synthesis with two registers, over domain $\mathcal{D}$. Intuitively, we use two distinct data values in order to encode the two boolean values, and add constraints in the formula in order to ensure that the register transducer uses two registers to store these data values all along the execution. ◀

▶ Remark 18. In Theorem 17, the lower bound already holds for two registers, and also if $\mathcal{D}$ has an existential first-order theory decidable in PTime. Moreover, this lower bound holds for *any* data domain, as long as you are able to express the equality.

We have seen in Lemma 1 that $(\mathbb{D}, =)$ and $(\mathbb{Q}, <)$ are completable data domains. In addition, it is easily seen that the existential first-order theory of $(\mathbb{D}, =)$ is decidable in NP, as well as for $(\mathbb{Q}, <)$. As a consequence of Theorem 17, we obtain:

▶ **Corollary 19.** *Register-bounded synthesis from CLTL$(\mathbb{D}, =)$ and CLTL$(\mathbb{Q}, <)$ is 2*ExpTime*-complete.*

5 ω-Regularly approximable data domains

Another important setting is the data domain $(\mathbb{N}, <)$. As said before, it is not completable, but worse than that, its set of satisfiable constraint words $\mathsf{SAT}_{k,X}$ is not regular. Actually, when considering only finite words, this set is regular, but it turns out that it is not regular when considering infinite words. Here is an example to illustrate this discrepancy. We define $\mathsf{decrease} = (x^{(1)} < x^{(0)} \wedge y^{(1)} = y^{(0)})$ and $\mathsf{reset} = (x^{(0)} = y^{(0)} \wedge y^{(1)} = y^{(0)})$. Then we look at the family of constraint words $\mathsf{reset}.\mathsf{decrease}^{i_1}.\mathsf{reset}.\mathsf{decrease}^{i_2}.\mathsf{reset}.\mathsf{decrease}^{i_3}\ldots$. Depending on the sequence $(i_n)_{n\geq 0}$, the constraint word will or will not be satisfiable: if $(i_n)_{n\geq 0}$ has an upper bound then by picking the first value for y big enough we can build a satisfying valuation, but otherwise the constraint word is not satisfiable. In this section, we will show an extension of the previous framework that allows to capture such data domains.

5.1 General approach

Let X be a set of variables. We let $\mathsf{LASSO} \subseteq (\mathcal{MC}_X^{(k)})^\omega$ denote the set of ultimately periodic constraint words (or lasso-shaped word).

▶ **Definition 20.** *We say that $\mathcal{D}$ is* effectively ω-regularly approximable *if for every k, X, we can build an ω-regular language $\mathsf{QSAT}_{k,X} \subseteq (\mathcal{MC}_X^{(k)})^\omega$ such that $\mathsf{SAT}_{k,X} \subseteq \mathsf{QSAT}_{k,X}$ and $\mathsf{SAT}_{k,X} \cap \mathsf{LASSO} = \mathsf{QSAT}_{k,X} \cap \mathsf{LASSO}$.*

▶ Remark 21. $\mathsf{QSAT}_{k,X}$ can be thought of as an over-approximation of $\mathsf{SAT}_{k,X}$ which is exact on lasso-shaped words.

▶ **Theorem 22.** *Let $\mathcal{D}$ be an effectively ω-regularly approximable data domain. Then register-bounded synthesis from specifications expressed as universal co-Büchi CA is decidable.*

Let $\mathcal{A}$ be a universal co-Büchi CA over $\mathcal{D}$, an ω-regularly approximable data domain, and R be a set of registers. Fix $\mathsf{QSAT}_{k,V\cup R} \subseteq (\mathcal{MC}_{V\cup R}^{(k)})^\omega$ as given in Definition 20. We define the following language:

$$\mathsf{W}_{\mathcal{A},R}^Q = \{\overline{a} \in \mathsf{AW}_{\mathbb{I},\mathbb{O},R}^\omega \mid \forall \overline{c} \in (\mathcal{C}_V^{(k)})^\omega, (\exists \overline{c'} \in \mathsf{join}(\overline{a},\overline{c}), \overline{c'} \in \mathsf{QSAT}_{k,V\cup R}) \Rightarrow \overline{c} \in L_{stx}(\mathcal{A})\}$$

▶ **Lemma 23.** $\mathsf{W}_{\mathcal{A},R}$ *is realizable a FT iff* $\mathsf{W}_{\mathcal{A},R}^Q$ *is realizable by a FT.*

Proof sketch. For the reverse direction, it is easy to verify that the inclusion $\mathsf{SAT}_{k,V\cup R} \subseteq \mathsf{QSAT}_{k,V\cup R}$ entails $\mathsf{W}_{\mathcal{A},R} \supseteq \mathsf{W}_{\mathcal{A},R}^Q$. Thus, if there exists a FT $\mathbb{T}$ that realizes $\mathsf{W}_{\mathcal{A},R}^Q$, *i.e.* $L(\mathbb{T}) \subseteq \mathsf{W}_{\mathcal{A},R}^Q$, then it also realizes $\mathsf{W}_{\mathcal{A},R}$.

To prove the direct implication, we will show that for any FT $\mathbb{T}$, $L(\mathbb{T}) \not\subseteq \mathsf{W}_{\mathcal{A},R}^Q$ entails $L(\mathbb{T}) \not\subseteq \mathsf{W}_{\mathcal{A},R}$. Following the lines of the proof of Lemma 14, and using the same notations, we consider the following definitions/equalities:

$$\begin{array}{lcl} \mathsf{W}'_Q & = & \{(\overline{a},\overline{c},\overline{c'}) \in \mathsf{A} \mid (\overline{a},\overline{c},\overline{c'}) \notin L_{\mathsf{join}} \vee \overline{c'} \notin \mathsf{QSAT}_{k,V\cup R} \vee \overline{c} \in L(\mathcal{A}_{stx})\} \\ \mathsf{W}_{\mathcal{A},R}^Q & = & \{\overline{a} \in \mathsf{AW}_{\mathbb{I},\mathbb{O},R}^\omega \mid \forall \overline{c} \in (\mathcal{C}_V^{(k)})^\omega, \forall \overline{c'} \in (\mathcal{MC}_{V\cup R}^{(k)})^\omega, (\overline{a},\overline{c},\overline{c'}) \in \mathsf{W}'_Q\} \end{array}$$

In addition, as $\mathsf{QSAT}_{k,V\cup R}$ is ω-regular, we can build a non-deterministic Büchi automaton B accepting the complement $\overline{\mathsf{W}'_Q}$ of W'_Q. Observe that by definition of W' and W'_Q, the equality $\mathsf{SAT}_{k,V\cup R} \cap \mathsf{LASSO} = \mathsf{QSAT}_{k,V\cup R} \cap \mathsf{LASSO}$ entails $\overline{\mathsf{W}'} \cap \mathsf{LASSO} = \overline{\mathsf{W}'_Q} \cap \mathsf{LASSO}$ $(\star)$.

Let $\mathbb{T}$ be an FT such that $L(\mathbb{T}) \not\subseteq \mathsf{W}_{\mathcal{A},R}^Q$: there exist $\overline{a} \in L(\mathbb{T})$, $\overline{c} \in (\mathcal{C}_V^{(k)})^\omega$, $\overline{c'} \in (\mathcal{MC}_{V\cup R}^{(k)})^\omega$ such that $(\overline{a},\overline{c},\overline{c'}) \notin \mathsf{W}'_Q$, and thus $(\overline{a},\overline{c},\overline{c'}) \in L(B)$. Considering the product $\mathbb{T} \times B$, we can exhibit a lasso shaped word $(\overline{b},\overline{d},\overline{d'}) \in L(B)$ such that $\overline{b} \in L(\mathbb{T})$. Property $(\star)$ entails $(\overline{b},\overline{d},\overline{d'}) \notin \mathsf{W}'$, hence $\overline{b} \notin \mathsf{W}_{\mathcal{A},R}$, and thus $L(\mathbb{T}) \not\subseteq \mathsf{W}_{\mathcal{A},R}$. ◀

▶ Remark 24. Observe that if $\mathsf{W}_{\mathcal{A},R}^Q$ is realizable a FT $\mathbb{T}$, $\mathsf{W}_{\mathcal{A},R}$ is also realizable by $\mathbb{T}$.

Proof sketch of Theorem 22

By Lemmas 13 and 23, we have that $L(\mathcal{A})$ is realizable iff $\mathsf{W}_{\mathcal{A},R}^Q$ is realizable by a FT. The definition of $\mathsf{W}_{\mathcal{A},R}^Q$ is the same as that of $\mathsf{W}_{\mathcal{A},R}$, while substituting $\mathsf{SAT}_{k,V\cup R}$ with $\mathsf{QSAT}_{k,V\cup R}$. As a consequence, Lemma 14 can be adapted to prove that $\mathsf{W}_{\mathcal{A},R}^Q$ is effectively ω-regular, and we conclude using Büchi-Landweber's Theorem.

5.2 Proving ω-regular approximability

In [15], a similar notion of ω-regular approximability is considered, but their setup is different as they do not start from logic but from register automata. As such, their syntactic input languages are not over constraint words as we do, but over action words. One of the differences is that they only speak of the future one time step ahead and that they receive values one at a time. Still, we will show that it is possible to transfer ω-regular approximability results from their setting to ours.

In the setting of [15], there is a single input, hence $\mathbb{I}$ should be a singleton ($\mathbb{I} = \{\star\}$) and there is no output ($\mathbb{O} = \emptyset$). Last, we let R denote some set of registers. With these choices, we denote by $\mathsf{FEAS}_R \subseteq (Act_{\{\star\},\emptyset,R})^\omega$ the set of action words $\overline{a}$ such that $[\![\overline{a}]\!] \neq \emptyset$. We will also denote by LASSO_{AW} the adaptation of LASSO to the set $(Act_{\{\star\},\emptyset,R})^\omega$.

▶ **Definition 25.** *Let $\mathcal{D}$ be a data domain. We say that $\mathcal{D}$ is effectively AW regularly approximable (AW-RA) if, for every R, we can build an ω-regular language $\mathsf{QFEAS}_R \subseteq \mathsf{AW}^\omega_{\mathbb{I},\mathbb{O},R}$ such that $\mathsf{FEAS}_R \subseteq \mathsf{QFEAS}_R$ and $\mathsf{FEAS}_R \cap \mathsf{LASSO}_{AW} = \mathsf{QFEAS}_R \cap \mathsf{LASSO}_{AW}$.*

Now we can state the following result:

▶ **Proposition 26** ([15]). *Each of the following data domain is effectively AW regularly approximable:*

- $(\mathbb{N}, <)$*: natural numbers with linear order*
- $(\mathbb{Z}, <)$*: integers with linear order*
- $(\mathbb{Z}^d, =^d, <^d)$*: tuples of integers, with pointwise linear order ($d \in \mathbb{N}$ is fixed)*
- $(\Sigma^*, \prec)$*: finite words over Σ with the prefix relation*

In addition, for each of these domains, given a set of registers R, the set QFEAS_R is recognized by a non-deterministic parity automaton with $exp(|R|)$ states and $poly(|R|)$ priorities.

This follows from different results proven in [15]. First, it is shown in Section 3.2 that for all R, $(\mathbb{N}, <)$ has a witness QFEAS_R of ω-regular approximability recognized by a non-deterministic parity automaton with $exp(|R|)$ states and $poly(|R|)$ priorities. Then, it is shown in Sections 4.2 and 4.3 that the other data domains reduce to $(\mathbb{N}, <)$. In Remark 18, it is explained why these reductions induce a construction for QFEAS_R that preserves its size and number of priorities (only a polynomial blowup occurs).

The next result allows us to transfer these positive results to our setting:

▶ **Lemma 27.** *If a data domain $\mathcal{D}$ is effectively AW-RA, then it is effectively ω-regularly approximable: if QFEAS_R can be recognized by a non-deterministic parity automaton with $exp(|R|)$ states and $poly(|R|)$ priorities, then $\mathsf{QSAT}_{k,X}$ can be recognized by a non-deterministic parity automaton with $exp((k+1).|X|)$ states and $poly((k+1).|X|)$ priorities.*

Proof sketch. The intuitive idea of the construction is to translate what happens in the setting of constraint sequences into that of action words over inputs of dimension 1. To that end, intuitively, we need to trade the higher dimension of input variables ($|X|$) and the possibility to look at horizon k with longer executions. Each step in the setting of constraint sequences will be simulated by $(k+1).|X|$ steps in the action words setting.

Let $\mathcal{D}$ be an effectively AW regularly approximable data domain. Let $X = \{x_1, \dots, x_{|X|}\}$ be a set of variables and $k \in \mathbb{N}$. We define the following set of registers:

$$R = \{x_{i,j} \mid 1 \leq i \leq |X|, 0 \leq j \leq k\}$$

In particular, we have $|R| = (k+1).|X|$

Formally, we define a mapping $\Psi : (\mathcal{MC}^{(k)}_X)^\omega \to (Act_{\{\star\},\emptyset,R})^\omega$ from constraint sequences over $X^{(k)}$ to action words over R (with the same conditions as above, *i.e.* singleton input variables, and empty set of output variables). We describe how it works on an example. Assume that $X = \{x, y, z\}$ and $k = 1$. We thus have access to six data values, which we will store in six registers. Each step in the constraint sequence setting is simulated by six steps in the action word setting. The way we convert $\overline{w} = c_1 c_2 \dots$ is depicted on Figure 2.

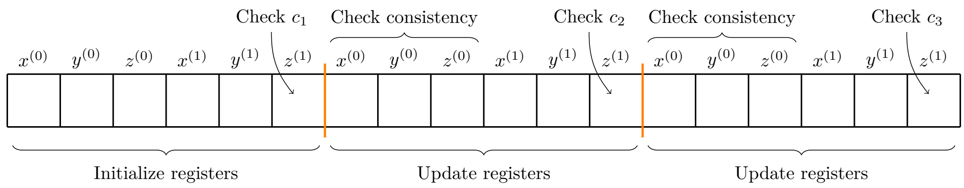


Figure 2 Illustration of the construction of Lemma 27.

During the first six steps, we initialize the registers with the data values that we read. At the end of these steps, we are able to check the first constraint c_1. Then, the data are processed six by six as follows: the first three data should correspond to data seen previously (as $x^{(0)}$ corresponds to $x^{(1)}$ one step before), so we have to check consistency. Still, we update the registers, and at the end of these six steps, we are able to check the constraint c_2.

We claim that this mapping fulfills the two following properties:

(i) $\forall \overline{w} \in (\mathcal{MC}_X^{(k)})^\omega, \overline{w} \in \mathsf{SAT}_{k,X} \Leftrightarrow \Psi(\overline{w}) \in \mathsf{FEAS}_R$

(ii) $\forall \overline{w} \in (\mathcal{MC}_X^{(k)})^\omega, \overline{w} \in \mathsf{LASSO} \Rightarrow \Psi(\overline{w}) \in \mathsf{LASSO}_{AW}$

We let $\mathsf{QSAT}_{k,X} = \Psi^{-1}(\mathsf{QFEAS}_R)$. Property (i) gives $\mathsf{SAT}_{k,X} = \Psi^{-1}(\mathsf{FEAS}_R)$. Thus, $\mathsf{FEAS}_R \subseteq \mathsf{QFEAS}_R$ entails, by monotonicity of the inverse image, $\mathsf{SAT}_{k,X} \subseteq \mathsf{QSAT}_{k,X}$. In addition, Property (ii) easily gives $\mathsf{QSAT}_{k,X} \cap \mathsf{LASSO} \subseteq \mathsf{SAT}_{k,X}$, which implies $\mathsf{SAT}_{k,X} \cap \mathsf{LASSO} = \mathsf{QSAT}_{k,X} \cap \mathsf{LASSO}$ as expected.

Regarding complexity, one can observe that Ψ is realized by an FT $\mathbb{T}_\Psi$ with two states. As $\mathsf{QSAT}_{k,X} = \Psi^{-1}(\mathsf{QFEAS}_R)$, we can build an automaton accepting $\mathsf{QSAT}_{k,X}$ by doing a wreath product between $\mathbb{T}_\Psi$ and an automaton recognizing QFEAS_R, yielding the result, as we have $|R| = (k+1).|X|$. ◀

▶ **Corollary 28.** *For $\mathcal{D} \in \{(\mathbb{N}, <), (\mathbb{Z}, <), (\mathbb{Z}^d, <^d), (\Sigma^*, \prec)\}$, register-bounded synthesis from $CLTL(\mathcal{D})$ is 2*ExpTime*-c.*

Proof sketch. Let $\mathcal{D}$ be one of these data domains and $\Phi \in CLTL(\mathcal{D})$. Let $\mathcal{A}$ be a universal co-Büchi CA built from Φ. By Proposition 26 and Lemma 27, a bound on the size of a non-deterministic parity automaton recognizing $\mathsf{QSAT}_{k,V \cup R}$ can be derived. Then, the complexity analysis done in the proof sketch of Theorem 17 can be adapted to show the upper bound. The lower bound follows from the one of Theorem 17 as it does not depend on the data domain (Remark 18). ◀

6 CLTL register-bounded synthesis with partial observation

Partial observation aims to improve the modeling capabilities. While a system may contain numerous variables, the controller usually has access to only a few of them [18]. In this section, we study an extension of CLTL that features partial observation: we split our set of input variables into two subsets, public (visible) inputs $\mathbb{I}_v$ and private (hidden) inputs $\mathbb{I}_h$.

▶ **Example 29.** To illustrate this setting, consider an environment that, at each turn, outputs two public values $\mathsf{in}_1, \mathsf{in}_2$. One of them must be equal to some private (hidden) variable t (target), that the controller aims at identifying infinitely often, using some variable g (guess) that it outputs at each turn. Such a setting can be captured as follows. Let $\mathbb{I}_v = \{\mathsf{in}_1, \mathsf{in}_2\}$, $\mathbb{I}_h = \{\mathsf{t}\}$ and $\mathbb{O} = \{\mathsf{g}\}$ and consider the following formula:

$$\Phi = G\left(\bigvee_{i\in\{1,2\}} \mathsf{in}_i^{(0)} = \mathsf{t}^{(0)}\right) \Rightarrow GF\left(\mathsf{g}^{(0)} = \mathsf{t}^{(0)}\right)$$

This formula is not realizable, as $\mathbb{D}$ is infinite and the way t alternates between in_1 and in_2 is arbitrary. However, if we assume a periodic behavior of the environment, then we obtain the following formula (here with period p):

$$\Phi_{\mathsf{per}} = \left(G\left(\mathsf{t}^{(0)} = \mathsf{t}^{(p)}\right) \wedge G\left(\bigvee_{i\in\{1,2\}} \mathsf{in}_i^{(0)} = \mathsf{t}^{(0)}\right)\right) \Rightarrow GF\left(\mathsf{g}^{(0)} = \mathsf{t}^{(0)}\right)$$

Now, we can show that this formula is realizable by a register transducer with 2 registers, which stores the two first inputs to identify which one repeats after p rounds.

Let $V = \mathbb{I}_v \uplus \mathbb{I}_h \uplus \mathbb{O}$ be a set of variables. We say that a transducer $\mathbb{T}$ with input alphabet $\mathbb{I}_v$ and output alphabet $\mathbb{O}$ *PO-realizes* a specification $L \subseteq (Val_{V,\mathbb{D}})^\omega$ iff $\forall \overline{w_h} \in (Val_{\mathbb{I}_h,\mathbb{D}})^\omega$, $\forall \overline{w} \in L(\mathbb{T})$, $\overline{w} \uplus \overline{w_h} \in L$.

Register-bounded Partial Observation Synthesis Problem from CLTL($\mathcal{D}$)
Input: A CLTL($\mathcal{D}$) formula Φ over $V = \mathbb{I}_v \uplus \mathbb{I}_h \uplus \mathbb{O}$ and an integer r
Output: A register transducer $\mathbb{T}$ over $(\mathbb{I}_v, \mathbb{O})$ with r registers that PO-realizes Φ, if it exists.

As the specification deals with input variables $\mathbb{I}_v \cup \mathbb{I}_h$, while the transducer only reads inputs in $\mathbb{I}_v$, we need to adapt the transfer lemma. To that end, given an action word $\overline{a}$ over $\mathbb{I}_h$, and an action word $\overline{a'}$ over $\mathbb{I}_v \cup \mathbb{I}_h$, we say that $\overline{a'}$ is a *completion* of $\overline{a}$ if for all $i > 0$, if $a_i = (C_i, \rho_i^{ass}, \rho_i^{out})$ then $a'_i = (C'_i, \rho_i^{ass}, \rho_i^{out})$, with $C'_i \in (\mathcal{MC}^{(k)}_{V\cup R})^\omega$ such that $C_i = C'_i|_{\mathbb{I}_v \cup R}$. We let $\mathsf{compl}_{\mathbb{I}_h}(\overline{a})$ denote the set of completions of $\overline{a}$.

Then, given a universal co-Büchi CA $\mathcal{A}$, we consider the following set:

$$\mathsf{W}^{PO}_{\mathcal{A},R} = \left\{\begin{array}{l} \overline{a} \in (Act_{\mathbb{I}_v,\mathbb{O},R})^\omega \mid \forall \overline{a'} \in \mathsf{compl}_{\mathbb{I}_h}(\overline{a}), \forall \overline{c} \in (\mathcal{C}_V^{(k)})^\omega, \\ (\exists \overline{c'} \in \mathsf{join}(\overline{a'}, \overline{c}), \overline{c'} \in \mathsf{SAT}_{k,V\cup R}) \Rightarrow \overline{c} \in L_{stx}(\mathcal{A}) \end{array}\right\}$$

We can then adapt the transfer lemma and prove:

▶ **Lemma 30** (Partial observation transfer Lemma). *Let $\mathcal{A}$ be a universal co-Büchi CA. Then $L(\mathcal{A})$ is PO-realizable by a RT with $|R|$ registers iff $\mathsf{W}^{PO}_{\mathcal{A},R}$ is realizable by a FT.*

Following the same lines as in Section 4, we prove:

▶ **Theorem 31.** *Let $\mathcal{D}$ be a data domain with effective ω-regular satisfiability. Register-bounded partial observation synthesis from CLTL($\mathcal{D}$) is decidable. If, in addition, $\mathcal{D}$ is completable and decidable in* ExpTime*, then it is in 2*ExpTime*.*

▶ **Corollary 32.** *Register-bounded partial observation synthesis from CLTL($\mathbb{D}, =$) and CLTL($\mathbb{Q}, <$) is 2*ExpTime*-c.*

7 Conclusion

We have shown that when the set of satisfiable constraint sequences over a data domain $\mathcal{D}$ is ω-regular, or ω-regularly approximable, then the register-bounded synthesis problem from $CLTL(\mathcal{D})$ is decidable. In addition, we have provided detailed complexity analysis to obtain optimal complexity results for most of the classical data domains studied in the literature. Last, we have also proven that our approach can be generalized to partial observation.

This work opens several perspectives. First, one could investigate natural extensions of this work, for instance by targeting other data domains (*e.g.* sets of natural numbers with inclusion [13]), or other logics over data words (*e.g.* freeze LTL [5]). Another direction consists in trying to lift successful approaches developed for reactive synthesis from the boolean to the data-aware setting. For instance, one could investigate compositional approaches, as well as heuristics based on antichains, as proposed in [12] to develop more efficient symbolic algorithms.

References

1 Ashwin Bhaskar and M. Praveen. Realizability problem for constraint LTL. *Information and Computation*, 2024. URL: https://arxiv.org/pdf/2207.06708.

2 Roderick Bloem, Krishnendu Chatterjee, and Barbara Jobstmann. Graph games and reactive synthesis. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 921–962. Springer, 2018. doi:10.1007/978-3-319-10575-8_27.

3 J. Richard Buchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969. URL: http://www.jstor.org/stable/1994916.

4 Stéphane Demri. LTL over integer periodicity constraints. *Theor. Comput. Sci.*, 360(1-3):96–123, 2006. URL: https://doi.org/10.1016/j.tcs.2006.02.019, doi:10.1016/J.TCS.2006.02.019.

5 Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009. doi:10.1145/1507244.1507246.

6 Stéphane Demri and Karin Quaas. Concrete domains in logics: a survey. *ACM SIGLOG News*, 8(3):6–29, July 2021. URL: https://hal.science/hal-03313291, doi:10.1145/3477986.3477988.

7 Stéphane Demri and Karin Quaas. Constraint automata on infinite data trees: from CTL(Z)/CTL*(Z) to decision procedures. In Guillermo A. Pérez and Jean-François Raskin, editors, *34th International Conference on Concurrency Theory, CONCUR 2023, September 18-23, 2023, Antwerp, Belgium*, volume 279 of *LIPIcs*, pages 29:1–29:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2023.29, doi:10.4230/LIPICS.CONCUR.2023.29.

8 Stéphane Demri and Deepak D'Souza. An automata-theoretic approach to constraint LTL. *Information and Computation*, 205(3):380–415, 2007. URL: https://www.sciencedirect.com/science/article/pii/S0890540106001076, doi:10.1016/j.ic.2006.09.006.

9 Léo Exibard, Emmanuel Filiot, and Ayrat Khalimov. Church synthesis on register automata over linearly ordered data domains. *Formal Methods Syst. Des.*, 61(2):290–337, 2022. URL: https://doi.org/10.1007/s10703-023-00435-w, doi:10.1007/S10703-023-00435-W.

10 Léo Exibard, Emmanuel Filiot, and Pierre-Alain Reynier. Synthesis of data word transducers. In Wan J. Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, volume 140 of *LIPIcs*, pages 24:1–24:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2019.24, doi:10.4230/LIPICS.CONCUR.2019.24.

11 Léo Exibard, Emmanuel Filiot, and Pierre-Alain Reynier. Synthesis of data word transducers. *Log. Methods Comput. Sci.*, 17(1), 2021. URL: https://lmcs.episciences.org/7279.

12 Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. Antichains and compositional algorithms for LTL synthesis. *Formal Methods Syst. Des.*, 39(3):261–296, 2011. URL: https://doi.org/10.1007/s10703-011-0115-3, doi:10.1007/S10703-011-0115-3.

13 Sabína Gulcíková and Ondrej Lengál. Register set automata (technical report). *CoRR*, abs/2205.12114, 2022. URL: https://doi.org/10.48550/arXiv.2205.12114, arXiv:2205.12114, doi:10.48550/ARXIV.2205.12114.

14 Swen Jacobs, Guillermo A. Pérez, Remco Abraham, Véronique Bruyère, Michaël Cadilhac, Maximilien Colange, Charly Delfosse, Tom van Dijk, Alexandre Duret-Lutz, Peter Faymonville, Bernd Finkbeiner, Ayrat Khalimov, Felix Klein, Michael Luttenberger, Klara J. Meyer, Thibaud Michaud, Adrien Pommellet, Florian Renkin, Philipp Schlehuber-Caissier, Mouhammad Sakr, Salomon Sickert, Gaëtan Staquet, Clément Tamines, Leander Tentrup, and Adam Walker. The reactive synthesis competition (SYNTCOMP): 2018-2021. *Int. J. Softw. Tools Technol. Transf.*, 26(5):551–567, 2024. URL: `https://doi.org/10.1007/s10009-024-00754-1`, `doi:10.1007/S10009-024-00754-1`.

15 Ayrat Khalimov, Emmanuel Filiot, and Léo Exibard. A generic solution to register-bounded synthesis with an application to discrete orders. *CoRR*, abs/2105.09978, 2021. URL: `https://arxiv.org/abs/2105.09978`, `arXiv:2105.09978`.

16 Ayrat Khalimov and Orna Kupferman. Register-bounded synthesis. In Wan J. Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, volume 140 of *LIPIcs*, pages 25:1–25:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: `https://doi.org/10.4230/LIPIcs.CONCUR.2019.25`, `doi:10.4230/LIPICS.CONCUR.2019.25`.

17 Ayrat Khalimov, Benedikt Maderbacher, and Roderick Bloem. Bounded synthesis of register transducers. In Shuvendu K. Lahiri and Chao Wang, editors, *Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings*, volume 11138 of *Lecture Notes in Computer Science*, pages 494–510. Springer, 2018. `doi:10.1007/978-3-030-01090-4\_29`.

18 Orna Kupferman and Moshe Y. Vardi. *Synthesis with Incomplete Information*, pages 109–127. Springer Netherlands, Dordrecht, 2000. `doi:10.1007/978-94-015-9586-5_6`.

19 Orna Kupferman and Moshe Y. Vardi. Safraless decision procedures. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 531–542. IEEE Computer Society, 2005. `doi:10.1109/SFCS.2005.66`.

20 A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '89, page 179–190, New York, NY, USA, 1989. Association for Computing Machinery. `doi:10.1145/75277.75293`.

A Omitted proofs of Section 4

Proof of Lemma 11

▶ **Lemma 11.** *Let* $\overline{a} \in \mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$ *then* $[\![\overline{a}]\!] = [\![\mathsf{cstr}(\overline{a})]\!]|_V$.

Proof. Intuitively, this property follows from the fact that when $\overline{a} \in \mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$, the language $[\![\overline{a}]\!]$ does not depend on d_0. Let $\overline{a} = (C_1, \rho_1^{ass}, \rho_1^{out}) \cdots \in \mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$ and $\mathsf{cstr}(\overline{a}) = \overline{C'}$. We proceed by double inclusion.

$\subseteq$. Let $\overline{w^V} \in (Val_{V,\mathbb{D}})^{\omega}$ be such that $\overline{w^V} \in [\![\overline{a}]\!]$. First by semantics of $[\![\overline{a}]\!]$, there is a valuation of the register $\overline{w^R}$ along the run of $\overline{a}$ on $\overline{w^V}$.

We call $\overline{w} = \overline{w^V} \uplus \overline{w^R}$. We claim that $\overline{w} \in [\![\mathsf{cstr}(\overline{a})]\!]$ which immediately yields the result. This follows from the definition of the semantics of $[\![\overline{a}]\!]$ and of the constraint word $\mathsf{cstr}(\overline{a})$, as we have simply encoded the semantics of $\overline{a}$ into $\mathsf{cstr}(\overline{a})$.

$\supseteq$. Let $\overline{w} \in (Val_{V \cup R,\mathbb{D}})^{\omega}$, we suppose $\overline{w} \in [\![\mathsf{cstr}(\overline{a})]\!]$. Then we know that for all $i > 0$, $w_i \models C'_i$ that is:

- $w_i \models C_i$.
- $\forall r \in R$ such that $\rho_i^{ass}(r)$ is defined, $w_i \models (r^{(1)} = \rho_i^{ass}(r))$.
- $\forall r \in R$ such that $\rho_i^{ass}(r)$ is not defined, $w_i \models (r^{(1)} = r)$.
- $\forall y \in \mathbb{O}$, $w_i \models (y = (\rho_i^{out}(y))^{(1)})$

We want to show $\overline{w}|_V \in [\![a]\!]$ that is, there exists a word $\overline{w^R} \in (Val_{R,\mathbb{D}})$ that satisfies the semantics of action words.

Let $\overline{reg} \in (Val_{R,\mathbb{D}})^{\omega}$ such that $\forall i \geq 1$, $\forall r \in R$:

$$reg_i(r) = \begin{cases} w_i(r) & \text{if } \exists j < i,\ \rho_j^{ass}(r) \text{ is defined} \\ d_0 & \text{otherwise} \end{cases}$$

Now we show that $\overline{w}|_V \in [\![\overline{a}]\!]$, using this valuation word for registers. Let $i > 0$:

- $reg_1(r) = d_0$ for all $r \in R$ as there is no $j < 1$ such that ρ_j^{ass} is defined (ρ^{ass} starts at one).
- Then we show, $w_i|_{\mathbb{I}} \uplus reg_i \models C_i$. by hypothesis we have $w_i \models C_i$, but test in action word are defined only on input variables and registers. So $w_i|_{\mathbb{I}} \uplus w_i|_R \models C_i$. Since $\overline{a} \in \mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$, we know that all registers that appear in C_i have been assigned strictly before step i. Hence, $w_i|_{\mathbb{I}} \uplus reg_i \models C_i$.
- We show for all $r \in R$, $reg_{i+1}(r) = w_i|_V \circ \rho_i^{ass}(r)$ if $\rho_i^{ass}(r)$ is defined. Let $r \in R$ such that $\rho_i^{ass}(r)$ is defined, by hypothesis $w_i \models (r^{(1)} = \rho_i^{ass}(r))$, so $w_{i+1}(r) = w_i(\rho_i^{ass}(r))$ and by definition of assignment $\rho_i^{ass}(r) \in V$, hence $w_{i+1}(r) = w_i|_V(\rho_i^{ass}(r))$. Also since $\rho_i^{ass}(r)$ is defined, $w_{i+1}(r) = reg_{i+1}(r)$, So finally $reg_{i+1}(r) = w_i|_V \circ \rho_i^{ass}(r)$
- Then we show for all $r \in R$, $reg_{i+1}(r) = reg_i(r)$ if $\rho_i^{ass}(r)$ is not defined. Let $r \in R$ such that $\rho_i^{ass}(r)$ is not defined, by hypothesis $w_i \models (r^{(1)} = r)$ so $w_{i+1}(r) = w_i(r)$. There are two cases:
  - if there exists $j < i$ such that $\rho_j^{ass}(r)$ is defined, then $reg_i(r) = w_i(r)$. In addition, we also have $j < i + 1$, hence $reg_{i+1}(r) = w_{i+1}(r)$. Together with $w_{i+1}(r) = w_i(r)$, we obtain $reg_{i+1}(r) = reg_i(r)$.
  - otherwise, there is no $j < i$ such that $\rho_j^{ass}(r)$ is defined. Then there is also no such $j < i + 1$ and thus both $reg_i(r) = d_0$ and $reg_{i+1}(r) = d_0$. We can conclude $reg_i(r) = reg_{i+1}(r)$ as well.
- We show $w_i^V = reg_{i+1} \circ \rho_i^{out}$. Let $y \in \mathbb{O}$. As $\overline{a} \in \mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$, we know that the register $r = \rho_i^{out}(y)$ has already received an assignment, that is, there exist $j \leq i$ such that $\rho_j^{ass}(r)$ is defined and thus $reg_{i+1}(r) = w_{i+1}(r)$. By hypothesis $w_i \models (y = (\rho_i^{out}(y))^{(1)})$, so $w_i(y) = w_{i+1}(\rho_i^{out}(y)) = reg_{i+1}(\rho_i^{out}(y)) = reg_{i+1} \circ \rho_i^{out}(y)$. So $w_i^V = reg_{i+1} \circ \rho_i^{out}$.

Finally $[\![\overline{a}]\!] = [\![\mathsf{cstr}(\overline{a})]\!]|_V$ ◀

Proof of Lemma 12

▶ **Lemma 12** (Adequation). *Let $\overline{a} \in \mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$, $\overline{c} \in (\mathcal{C}_V^{(k)})^{\omega}$, $\overline{w} \in Val_{V\cup R,\mathbb{D}}$ and $\overline{c'} \in (\mathcal{MC}^{(k)}_{V\cup R})^{\omega}$ such that $\overline{w} \models \overline{c'}$. Then $\overline{c'} \in \mathsf{join}(\overline{a},\overline{c})$ iff $\overline{w}|_V \models \overline{a}$ and $\overline{w}|_V \models \overline{c}$.*

Proof. Let $\overline{c} \in (\mathcal{C}_V^{(k)})^{\omega}$, $\overline{a} \in \mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$, $\overline{w} \in Val_{V\cup R,\mathbb{D}}$, $\overline{c'} \in (\mathcal{MC}^{(k)}_{V})^{\omega}$ such that $\overline{w} \models \overline{c'}$.

$\Rightarrow$

Suppose that $\overline{c'} \in \mathsf{join}(\overline{a},\overline{c})$. We first show $\overline{w}|_V \models \overline{c}$. As $\overline{c} \subseteq \overline{c'}$, we have for all $i \geq 0$ for all literal $p \in c_i$, $p \in c'_i$, but $\overline{w} \models \overline{c'}$, hence $w, i \models c'_i$ and directly $w, i \models p$, but as p is a predicate whose variable are all in V, $w|_V, i \models p$. So $\overline{w}|_V \models \overline{c}$.

Then for $\overline{a}$, with the same argument as above we can get $\overline{w} \models \mathsf{cstr}(\overline{a})$ and by Lemma 11, we have $\overline{w}|_V \models \overline{a}$.

$\Leftarrow$

We suppose $\overline{w}|_V \models \overline{a}$ and $\overline{w}|_V \models \overline{c}$. We first show $[\![\overline{c'}]\!] \subseteq [\![\overline{c}]\!]$. As $\overline{w} \in [\![\overline{c'}]\!]$, for all $i \geq 0$ for all literal $p \in c_i$, as $\overline{c'}$ is maximally consistent and $\overline{w}, i \models p$, $p \in c'_i$, hence $\overline{c} \subseteq \overline{c'}$.

Then for $\overline{a}$, by Lemma 11, we have $\overline{w} \models \mathsf{cstr}(\overline{a})$ and then with the same argument as above $\mathsf{cstr}(\overline{a}) \subseteq \overline{c'}$. And by definition of join, $\overline{c'} \in \mathsf{join}(\overline{a},\overline{c})$. ◀

Proof of Lemma 14

▶ **Lemma 14.** *Let $\mathcal{A}$ be a universal co-Büchi CA with n states. If $\mathcal{D}$ has effective ω-regular satisfiability, then $\mathsf{W}_{\mathcal{A},R}$ is effectively ω-regular: given $k \in \mathbb{N}$ and registers R, if $\mathsf{SAT}_{k,V\cup R}$ is recognizable by a non-deterministic parity automaton with n_s states and c_s colors, then $\mathsf{W}_{\mathcal{A},R}$ is accepted by a universal co-Büchi automaton with $O(n_s.c_s.n.2^{|R|})$ states.*

Proof. Given two alphabets A and B, we say that a set $L \subseteq A^{\omega} \times B^{\omega}$ is recognizable by some automaton if the set of words $w = (a_1, b_1)(a_2, b_2) \cdots \in (A \times B)^{\omega}$ such that $(a_1 a_2 \ldots, b_1 b_2 \ldots) \in L$, is recognizable by some automaton over alphabet $A \times B$. This notion is naturally generalized to sets $L \subseteq A_1^{\omega} \times \cdots \times A_n^{\omega}$, for $A_1, \ldots, A_n$ arbitrary alphabets.

We define the following objects:

$$\begin{array}{lcl}
\mathsf{A} & = & (Act_{\mathbb{I},\mathbb{O},R})^{\omega} \times (\mathcal{C}_V^{(k)})^{\omega} \times (\mathcal{MC}^{(k)}_{V\cup R})^{\omega} \\
L_{\mathsf{join}} & = & \{(\overline{a},\overline{c},\overline{c'}) \in \mathsf{A} \mid \overline{c'} \in \mathsf{join}(\overline{a},\overline{c})\} \\
\mathsf{W}' & = & \{(\overline{a},\overline{c},\overline{c'}) \in \mathsf{A} \mid (\overline{a},\overline{c},\overline{c'}) \notin L_{\mathsf{join}} \vee \overline{c'} \notin \mathsf{SAT}_{k,V\cup R} \vee \overline{c} \in L(\mathcal{A}_{stx})\} \\
\mathsf{W}_{\mathcal{A},R} & = & \{\overline{a} \in \mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R} \mid \forall \overline{c} \in (\mathcal{C}_V^{(k)})^{\omega}, \forall \overline{c'} \in (\mathcal{MC}^{(k)}_{V\cup R})^{\omega}, (\overline{a},\overline{c},\overline{c'}) \in \mathsf{W}'\}
\end{array}$$

First, one can easily verify that an action word belongs to $\mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$ by storing in the state the set of registers assigned so far and verifying that the test only considers initialized registers. The number of states is exponential in the number of registers.

Secondly, L_{join} is recognizable by a deterministic (safety) automaton A_{join} with two states, one accepting and one rejecting sink. Let us describe its transitions. Remind the construction of Lemma 11, which from the constraint of an action a over V, constructs a constraint that we denote $\mathsf{cstr}(\overline{a})$, over $V \cup R$. Now, in the accepting state, upon reading a triplet (c, a, c'), the automaton stays in the accepting state if c' is maximally consistent, $c \subseteq c'$ and $\mathsf{cstr}(\overline{a}) \subseteq c'$. Otherwise it goes to the sink state.

Now, our goal is to build an automaton for the set W'. Note that $\mathsf{W}_{\mathcal{A},R} = \{\overline{a} \mid \forall \overline{c} \forall \overline{c'}, (\overline{a},\overline{c},\overline{c'}) \in \mathsf{W}'\}$ $(\star)$.

Now let us call $\mathcal{A}_{sat}$ the non-deterministic parity automaton recognizing $\{(\overline{a},\overline{c},\overline{c'}) \in \mathsf{A} \mid \overline{c'} \in \mathsf{SAT}_{k,V\cup R}\}$. Now, any constraint automaton $\mathcal{A}$ can be seen as $\mathcal{A}_{stx}$, a finite state automata over the finite alphabet $\mathcal{C}_V^{(k)}$, that recognizes $L_{stx}(\mathcal{A})$, which is therefore regular. Note that $\mathcal{A}_{stx}$ is a universal coBüchi automaton, we also extend this automaton to take as input triplet of A, that is we define $\mathcal{A}'_{stx}$ the automata recognizing $\{(\overline{a},\overline{c},\overline{c'}) \in \mathsf{A} \mid \overline{c} \in L_{stx}(\mathcal{A})\}$. Therefore, we define

$$\mathsf{W}' = L(\mathcal{A}_{\mathsf{join}})^C \cup L(\mathcal{A}_{sat})^C \cup L(\mathcal{A}'_{stx})$$

We denote $U = \mathsf{W}'^C = L(\mathcal{A}_{\mathsf{join}}) \cap L(\mathcal{A}_{sat}) \cap L(\mathcal{A}'_{stx})^C$ and show how to build an automaton recognizing it. As described above, the automaton $\mathcal{A}_{\mathsf{join}}$ is a Büchi automaton (safety is a particular case of Büchi). The automaton $\mathcal{A}_{sat}$ can be converted into a non-deterministic Büchi automaton with $O(n_s.c_s)$ states. Finally, interpreting $\mathcal{A}'_{stx}$ as a non-deterministic Büchi automaton, we get a non-deterministic Büchi automaton recognizing $L(\mathcal{A}'_{stx})^C$ in constant time. The classical intersection operation on non-deterministic Büchi automata yields a non-deterministic Büchi automaton $\mathcal{A}_U$ with $O(n.n_s.c_s)$ states recognizing U. Let $\mathcal{A}_{\mathsf{W}'}$ be $\mathcal{A}_U$ with inverted parity (one become zero and two become one) interpreted as a universal coBüchi automaton, it will recognize the complement of $\mathcal{A}_U$. We finally have $L(\mathcal{A}_{\mathsf{W}'}) = W'$, and $\mathcal{A}_{\mathsf{W}'}$ has $O(n.n_s.c_s)$ states.

Using the equality ($\star$), by projecting the transitions of U on the alphabet $Act_{\mathbb{I},\mathbb{O},R}$, and taking a product with a deterministic automaton with $O(2^{|R|})$ states recognizing $\mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$, we get a universal coBüchi automaton recognizing W, with $O(n.n_s.c_s.2^{|R|})$ states.

◀

Proof of Lemma 15

▶ **Lemma 15.** *Let $\mathcal{D}$ be a decidable completable data domain (resp. decidable in* ExpTime*), then $\mathcal{D}$ has ω-regular satisfiability (resp. a deterministic parity automaton recognizing $\mathsf{SAT}_{k,X}$ can be constructed that has $exp(k.|X|)$ states and 2 priorities.*

Proof. Given a constraint C, we let $future(C)$ be the constraint which consists of all literals $\ell(x_1^{(i_1-1)},\ldots,x_n^{(i_n-1)})$ for all literals $\ell(x_1^{(i_1)},\ldots,x_n^{(i_n)}) \in C$ such that $i_1,\ldots,i_n \neq 0$.

We show the ω-regularity by exhibiting a safety deterministic automaton $S = (Q, i, \delta, F)$ recognizing $\mathsf{SAT}_{k,V\cup R}$.

- $Q = \{q_{init}\} \cup \{c \in \mathcal{MC}_V^{(k-1)} \mid c \text{ is satisfiable}\}$
- $i = q_{init}$
- $\delta = \{(q, c, p) \mid q = c|_{V^{(k-1)}} \text{ and } p = future(c)\}$
- $F = Q$

Now let $\overline{w} \in \mathcal{MC}_V^{(k)}$, as long as the constraints w_i and w_{i+1} are compatible with the previous one and are satisfiable, we can take a valuation of the first one and thanks to the completion property there exists a completion on the new variable. Our automata do check those two property and, as such recognize $\mathsf{SAT}_{k,V\cup R}$. The size of $\mathcal{MC}_{V\cup R}^{(k)}$ is exponential in V and R and computing the states of the automata consist of enumerating the constraint of $\mathcal{MC}_{V\cup R}^{(k)}$ and checking if the constraint are satisfiable which is P then computing the states is in EXP, then computing the transition is easy, for any two state p, q, you check if $future(p) = q|_{V^{(k-1)}}$ if it is you can add a transition labeled $t \in \mathcal{MC}_{V\cup R}^{(k)}$ such that for $x^{(i)} \in (V \cup R)^{(k)}$ $t(x^{(i)}) = \begin{cases} p(x^{(i)}), & \text{if i} \in [\![0, k-1]\!] \\ q(x^{(i)}), & \text{if x} = k \end{cases}$.

◀

Proof of Theorem 17

▶ **Theorem 17.** *Let $\mathcal{D}$ be a decidable completable data domain (resp. decidable in* ExpTime*). Then register-bounded synthesis from CLTL($\mathcal{D}$) is decidable (resp.* 2ExpTime-c*). It is* 2ExpTime-hard *for any fixed number of registers $r \geq 2$.*

Proof. **Upper bound**

Consider some formula $\Phi \in CLTL(\mathcal{D})$. By Proposition 5, we can build a universal co-Büchi constraint automaton $\mathcal{A}$ whose size is expoential in the size of Φ. Let $\mathcal{A}$ denote this automaton. Thanks to Lemma 13, register bounded synthesis w.r.t. $L(\mathcal{A})$ boils down to classical synthesis for the specification $\mathsf{W}_{\mathcal{A},R}$. By Lemma 15, $\mathsf{SAT}_{k,V\cup R}$ can be constructed in exponential time, and thus, by Lemma 14, $\mathsf{W}_{\mathcal{A},R}$ is recognized by a universal co-Büchi automaton of exponential size. The result follows as synthesis over a finite alphabet from a universal co-Büchi automaton can be solved in exponential time [19].

Lower bound

For the lower bound, we reduce the problem of LTL synthesis (over finite alphabets), which is already 2EXPTIME-C [20], to bounded register synthesis with two registers, over domain $\mathcal{D}$ (remind that we always assume that have the equality predicate). We discuss how to extend it to r registers at the end of the proof.

Intuitively, we will use two distinct data values in order to encode the two boolean values, and add constraints in the formula in order to ensure that the register transducer uses two registers to store these data values all along the execution.

Remind that for LTL synthesis, the atomic propositions are partitioned into input atomic propositions in some finite set AP_{in} controlled by the environment, and output atomic propositions in some finite set AP_{out} controlled by the system. Let $AP = AP_{in} \uplus AP_{out}$. The idea is the following, for each propositional variable $a \in AP$, we associate a variable $v_a \in \mathbb{I}$ if $a \in AP_{in}$, and $v_a \in \mathbb{O}$ if $a \in AP_{out}$, and four variables $v_\top^{in} \in \mathbb{I}$, $v_\perp^{in} \in \mathbb{I}$, $v_\top^{out} \in \mathbb{O}$ and $v_\perp^{out} \in \mathbb{O}$. So, $\mathbb{I} = \{v_a \mid a \in AP_{in}\} \cup \{v_\top^{in}, v_\perp^{in}\}$, and $\mathbb{O} = \{v_a \mid a \in AP_{out} \cup \{v_\top^{out}, v_\perp^{out}\}\}$.

Now we define the following formulae *Assume* and *Guarantee* which belong to $CLTL(\mathcal{D})$ over the set of variables $\mathbb{I} \cup \mathbb{O}$:

$$Assume := (v_\top^{in} \neq v_\perp^{in}) \wedge G\left(v_\top^{in} = (v_\top^{in})^{(1)} \wedge v_\perp^{in} = (v_\perp^{in})^{(1)}\right) \wedge \bigwedge_{a \in AP} G(v_a = v_\top^{in} \vee v_a = v_\perp^{in})$$

$$Guarantee := G\left(v_\top^{in} = v_\top^{out} \wedge v_\perp^{in} = v_\perp^{out}\right)$$

We now define a reduction $\Psi : LTL \to CLTL(\mathcal{D})$ such that $\Psi(\Phi) = Assume \to (Guarantee \wedge \Phi[a \leftarrow (v_a = v_\top^{in}), \neg a \leftarrow (v_a = v_\perp^{in}), \forall a \in AP])$.

The part within [.] in the expression above means that any occurrence the literal a is replaced by the constraint $v_a = v_\top^{in}$. Similarly, the negation $\neg a$ is replaced by the constraint $v_a = v_\perp^{in}$. Hence, the two values $v_\top^{in}$ and $v_\perp^{in}$ represent the two boolean values.

We show that Φ is a positive instance of LTL synthesis iff $\Psi(\Phi)$ can be realized by a register transducer with two registers.

$\Rightarrow$ Let Φ be a positive instance of LTL synthesis, let $\mathbb{T} = (Q, q_0, \delta)$ be a finite transducer that realizes it. We describe how to build a transducer register $\mathbb{T}'$ with two registers that realizes $\Psi(\Phi)$.:

$\mathbb{T}' = (Q', q_{init}, R, V, \delta')$ with

- $Q' = Q \cup q_{init}$
- $R = \{r_1, r_2\}$
- $V = \mathbb{I} \uplus \mathbb{O}$
- $\delta' = \{(q, C_\alpha, \rho_0^{ass}, \rho_\beta^{out}, p) \mid (p, \alpha, \beta, q) \in \delta\} \uplus \{(q_{init}, C_\alpha, \rho^{ass}, \rho_\beta^{out}, p) \mid (q_0, \alpha, \beta, p) \in \delta\}$

Where C_α is

$$C_\alpha = \bigwedge_{a \in AP_{in}} v_a = r_{2-\alpha(a)}$$

And the output ρ_β^{out} is defined as follows:

$$\forall a \in AP_{out}, \rho^{out}(v_a) = \begin{cases} r_1 & \text{if} \quad \beta(a) = 1 \\ r_2 & \text{otherwise} \end{cases}$$

ρ_0 is the trivial assignment and ρ^{ass} is defined as $\rho^{ass}(r_1) = v_\top^{in}$ and $\rho^{ass}(r_2) = v_\perp^{in}$. Note that $\mathbb{T}'$ is not complete, but any completion would satisfy the CLTL formula.

$\Leftarrow$ For the converse, the key to this construction is that by forcing in $\Psi(\Phi)$ the variable $v_\top^{out}$ and $v_\top^{out}$ to be constant and output at each step we can keep track of which register contains which one of the two values, thus allowing to recreate the associated Boolean valuation.

Recall that the data domain $\mathcal{D}$ comes with a distinguished data value d_0, used to initialize the registers of the register transducer. Let $d_\top, d_\perp \in \mathbb{D} \setminus \{d_0\}$, with $d_\top \neq d_\perp$.

We define a mapping $f : \{0,1\}^{AP} \to Val_{\mathbb{I} \uplus \mathbb{O}, \mathbb{D}}$. Let $b \in \{0,1\}^{AP}$ we define $w = f(b)$ as follows: $\forall a \in AP$,

$$w(v_a) = \begin{cases} d_\perp \text{ if } b(a) = 0 \\ d_\top \text{ if } b(a) = 1 \end{cases}$$

$\forall x \in \{in, out\}$, $\forall y \in \{\top, \perp\}$,

$$w(v_y^x) = \begin{cases} d_\perp \text{ if } y = \perp \\ d_\top \text{ if } y = \top \end{cases}$$

We extend this mapping to infinite words as follows. Given $\overline{b} = b_1 b_2 \ldots \in (\{0,1\}^{AP})^\omega$, we define $f(\overline{b}) = \overline{w} = w_1 w_2 \ldots$ where for each $i \geq 1$ we have $w_i = f(b_i)$.

With slight abuse of notation, we may also apply mapping f to $\alpha \in \{0,1\}^{AP_{in}}$.

Let $\mathbb{T} = (Q, q_0, R = \{r_0, r_1\}, V, \delta)$ be a register transducer realizing $\Psi(\Phi)$, we build a new finite state transducer $\mathbb{T}' = (Q', q_0', \delta')$ where $Q' = Q \times \{0,1\} \cup \{q_0'\}$ with q_0' a fresh state. The transitions are defined as follows:

1. Starting from q_0': let $\alpha \in \{0,1\}^{AP_{in}}$, consider $w = f(\alpha)$. By definition of register transducers, registers are initialized to d_0. So we can find the unique transition $(q_0, C, \rho^{ass}, \rho^{out}, q)$ such that $w \cup \{d_0\}^R \models C$. As $\mathbb{T}$ realizes $\Psi(\Phi)$, we know that $\mathbb{T}$ outputs $d_\perp$ and $d_\top$. As both registers were initialized to the different data value d_0, $\mathbb{T}$ must store $d_\top$ and $d_\perp$ in its registers: ρ^{ass} is a total mapping. We build a transition $t' = (q_0', \alpha, \beta, (q, x))$, with β defined as:
$$\begin{array}{llll} \forall a \in AP_{out}, & \beta(a) = 1 & \text{if} & \rho^{out}(v_a) = \rho^{out}(v_\top^{out}) \\ \forall a \in AP_{out}, & \beta(a) = 0 & \text{if} & \rho^{out}(v_a) = \rho^{out}(v_\perp^{out}) \end{array}$$
and $x = \begin{cases} 1 \text{ if } \rho^{ass}(r_1) = v_\top^{in} \in C \\ 0 \text{ otherwise} \end{cases}$
2. Now for each $(q, x) \in Q \times \{0,1\}$, $\alpha \in \{0,1\}^{AP_{in}}$. Again we let $w = f(\alpha)$ and we consider the register valuation which maps r_x to $d_\top$ and r_{1-x} to $d_\perp$. So we can find the unique transition $(q, C, \rho^{ass}, \rho^{out}, q')$ such that the resulting valuation over $\mathbb{I} \cup R$ satisfies C. we build a new transition $t' = ((q, x), \alpha, \beta, (q', x'))$, with β and x' defined as $x' = \begin{cases} 1 \text{ if } \rho^{out}(v_\top^{out}) = r_1 \\ 0 \text{ otherwise} \end{cases}$. This is correct as $\mathbb{T}$ realizes $\Psi(\Phi)$, hence it should satisfy the formula *Guarantee* that ensures that $v_\top^{out}$ is equal to $d_\top$.

δ' is then the union of the transitions previously defined. First observe that by construction $\mathbb{T}'$ is input-deterministic, and thus a transducer.

We now want to show the correction of this construction. That is, $L(\mathbb{T}') \subseteq L(\Phi)$.

We will first prove that $\forall \overline{b} \in (\{0,1\}^{AP})^\omega$, $\overline{b} \models \Phi$ iff $f(\overline{b}) \models \Psi(\Phi)$.

First, we have $f(\bar{b}) \models Assume$ and $f(\bar{b}) \models Guarantee$ by construction. As a consequence $f(\bar{b}) \models \Psi(\Phi)$ iff $f(\bar{b}) \models \Phi[a \leftarrow (v_a = v_\top^{in}), \neg a \leftarrow (v_a = v_\bot^{in}), \forall a \in AP]$. Now we want to show that the latter is equivalent to $\bar{b} \models \Phi$. For all $i \geq 1$, by definition of $f(\bar{b})$, we have:

$$\begin{array}{rcl} \bar{b}, i \models a & \iff & f(\bar{b}), i \models v_a = v_\top^{in} \\ \bar{b}, i \models \neg a & \iff & f(\bar{b}), i \models v_a = v_\bot^{in} \end{array}$$

This entails the expected equivalence.

Secondly, we can show, by induction on the length of the run, that we can derive from the run of $\mathbb{T}'$ on $\bar{b}$ a run of $\mathbb{T}$ on $f(\bar{b})$. This follows from the fact that the transitions of $\mathbb{T}'$ have been built on the images of elements of $\{0,1\}^{AP_{in}}$ by f.

Now we are ready to conclude: let $\bar{b} \in L(\mathbb{T}')$. By the previous property, we deduce $f(\bar{b}) \in L(\mathbb{T})$. As $\mathbb{T}$ realizes $\Psi(\Phi)$, we have $f(\bar{b}) \models \Psi(\Phi)$. Thanks to the property proven before, this entails $\bar{b} \models \Phi$.

Extension to r registers. If we want to reduce synthesis of r registers transducers, we can use the same LTL formula. As the only input data values are $d_\top$ and d_{bot}, the registers can only store three data values: $d_\top, d_{bot}, d_0$. Instead of equipping states with a boolean to know which register corresponds to value $d_\top$, we can enrich states with a partition of the set of r registers into three sets corresponding to the three data values. As values $d_\top$ and $d_\bot$ must be output at each step, we know that sets corresponding to these two values must be non-empty. The rest of the construction can easily be adapted. ◀

B Omitted proofs of Section 5

▶ **Lemma 33.** *Let $\mathcal{A}$ be a universal co-Büchi CA with n states. If $\mathcal{D}$ is effectively ω-regularly approximable, then $\mathsf{W}^Q_{\mathcal{A},R}$ is effectively ω-regular: given $k \in \mathbb{N}$ and a set of registers R, if $\mathsf{QSAT}_{k,V \cup R}$ is recognizable by a non-deterministic parity automaton with n_s states and c_s colors, then $\mathsf{W}^Q_{\mathcal{A},R}$ is recognizable by a universal co-Büchi automaton with $O(n_s.c_s.n.2^{|R|})$ states.*

Proof. $\mathsf{W}^Q_{\mathcal{A},R}$ is obtained from $\mathsf{W}_{\mathcal{A},R}$ by replacing $\mathsf{SAT}_{k,V \cup R}$ by $\mathsf{QSAT}_{k,V \cup R}$. As the proof of Lemma 14 is based on automaton constructions, the same reasoning can be used to analyze the set $\mathsf{W}^Q_{\mathcal{A},R}$. ◀

Proof of Theorem 22

▶ **Theorem 22.** *Let $\mathcal{D}$ be an effectively ω-regularly approximable data domain. Then register-bounded synthesis from specifications expressed as universal co-Büchi CA is decidable.*

Proof. Consider some universal co-Büchi CA $\mathcal{A}$. Lemmas 13 and 23 reduce realizability of $L(\mathcal{A})$ to (finite) realizability of $\mathsf{W}^Q_{\mathcal{A},R}$. Lemma 33 entails that as $\mathcal{D}$ is effectively ω-regular, then the set $\mathsf{W}^Q_{\mathcal{A},R}$ can be effectively recognized by a universal co-Büchi automaton. We conclude using Büchi-Landweber's Theorem. ◀

Proof of Lemma 27

▶ **Lemma 27.** *If a data domain $\mathcal{D}$ is effectively AW-RA, then it is effectively ω-regularly approximable: if QFEAS_R can be recognized by a non-deterministic parity automaton with $exp(|R|)$ states and $poly(|R|)$ priorities, then $\mathsf{QSAT}_{k,X}$ can be recognized by a non-deterministic parity automaton with $exp((k+1).|X|)$ states and $poly((k+1).|X|)$ priorities.*

Proof. The intuitive idea of the construction is to translate what happens in the setting of constraint sequences into that of action words over inputs of dimension 1. To that end, intuitively, we need to trade the higher dimension of input variables ($|X|$) and the possibility to look at horizon k with longer executions. Each step in the setting of constraint sequences will be simulated by $(k+1).|X|$ steps in the action words setting.

Let $\mathcal{D}$ be an effectively AW regularly approximable data domain. Let $X = \{x_1, \dots, x_{|X|}\}$ be a set of variables and $k \in \mathbb{N}$. We define the following set of registers:

$$R = \{x_{i,j} \mid 1 \leq i \leq |X|, 0 \leq j \leq k\}$$

In particular, we have $|R| = (k+1).|X|$

Formally, we define a mapping $\Psi : (\mathcal{MC}_X^{(k)})^\omega \to (Act_{\{\star\},\emptyset,R})^\omega$ from constraint sequences over $X^{(k)}$ to action words over R (with the same conditions as above, *i.e.* singleton input variables, and empty set of output variables). We describe how it works on an example. Assume that $X = \{x, y, z\}$ and $k = 1$. We thus have access to six data values, which we will store in six registers. Each step in the constraint sequence setting is simulated by six steps in the action word setting. The way we convert $\overline{w} = c_1 c_2 \dots$ is depicted on Figure 3.

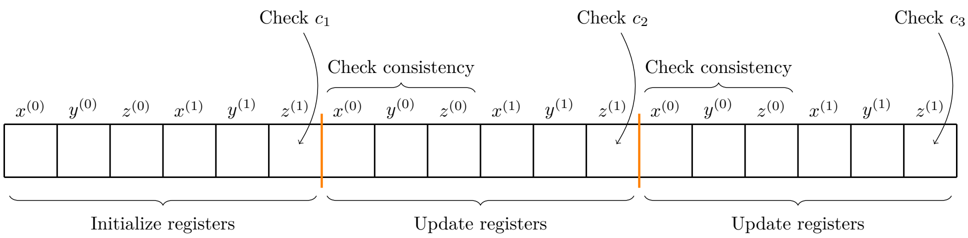


Figure 3 Illustration of the construction of Lemma 27.

During the first six steps, we initialize the registers with the data values that we read. At the end of these steps, we are able to check the first constraint c_1. Then, the data are processed six by six as follows: the first three data should correspond to data seen previously (as $x^{(0)}$ corresponds to $x^{(1)}$ one step before), so we have to check consistency. Still, we update the registers, and at the end of these six steps, we are able to check the constraint c_2.

We give now the formal definition of mapping Ψ. This mapping is defined from two mappings Ψ_{init} and Ψ_{next} that we define now.

We call $\Psi_{init} : \mathcal{MC}_X^{(k)} \to (Act_{\{\star\},\emptyset,R})^*$ the function from $\mathcal{MC}_X^{(k)}$ to finite register action words that for a given constraint c generates the following action word:

$$\Psi_{init}(c) = \prod_{j=0}^{k} \prod_{i=1}^{|X|} (\alpha_{i,j}(c), x_{i,j} \downarrow)$$

where

$$\alpha_{i,j}(c) = \begin{cases} \bigwedge\limits_{p(x_{i_1}^{(j_1)} \dots x_{i_l}^{(j_l)}) \in c} p(x_{i_1,j_1} \dots x_{i_l,j_l})[x_{|X|,k} \leftarrow \star] & \text{if } i = |X| \text{ and } j = k \\ \top & \text{otherwise.} \end{cases}$$

Similarly, we define $\Psi_{next} : \mathcal{MC}_X^{(k)} \to (Act_{\{\star\},\emptyset,R})^*$ the function from $\mathcal{MC}_X^{(k)}$ to finite register action words that for a given constraint c generates the following action word:

$$\Psi_{next}(c) = \left(\prod_{j=0}^{k-1} \prod_{i=1}^{|X|} (x_{i,j+1} = \star, \downarrow x_{i,j}) \right) \prod_{i=1}^{|X|} (\alpha_{i,k}(c), \downarrow x_{i,k})$$

where $\alpha_{i,j}(c)$ is defined before for Ψ_{init}.

We are now ready to define Ψ. Given $\overline{c} = c_1c_2c_3\ldots \in (\mathcal{MC}_X^{(k)})^\omega$, we define $\Psi(\overline{c}) \in (Act_{\{\star\},\emptyset,R})^\omega$ as follows:

$$\Psi(\overline{c}) = \Psi_{init}(c_1)\Psi_{next}(c_2)\Psi_{next}(c_3)\ldots$$

We claim that this mapping fulfills the following two properties:

(i) $\forall \overline{w} \in (\mathcal{MC}_X^{(k)})^\omega, \overline{w} \in \mathsf{SAT}_{k,X} \Leftrightarrow \Psi(\overline{w}) \in \mathsf{FEAS}_R$

(ii) $\forall \overline{w} \in (\mathcal{MC}_X^{(k)})^\omega, \overline{w} \in \mathsf{LASSO} \Rightarrow \Psi(\overline{w}) \in \mathsf{LASSO}_{AW}$

We explain why this entails the expected result. We let $\mathsf{QSAT}_{k,X} = \Psi^{-1}(\mathsf{QFEAS}_R)$ and we prove that it satisfies the two expected properties:

- $\mathsf{FEAS}_R \subseteq \mathsf{QFEAS}_R$ entails, by monotonicity of the inverse image, $\Psi^{-1}(\mathsf{FEAS}_R) \subseteq \Psi^{-1}(\mathsf{QFEAS}_R)$. This entails $\mathsf{SAT}_{k,X} \subseteq \mathsf{QSAT}_{k,X}$ as $\mathsf{SAT}_{k,X} = \Psi^{-1}(\mathsf{FEAS}_R)$ by Property (i) and $\mathsf{QSAT}_{k,X} = \Psi^{-1}(\mathsf{QFEAS}_R)$ by definition.
- To show $\mathsf{SAT}_{k,X} \cap \mathsf{LASSO} = \mathsf{QSAT}_{k,X} \cap \mathsf{LASSO}$, we only have to prove $\mathsf{QSAT}_{k,X} \cap \mathsf{LASSO} \subseteq \mathsf{SAT}_{k,X}$, the other inclusions being trivial. Consider some $\overline{w} \in \mathsf{QSAT}_{k,X} \cap \mathsf{LASSO}$. We thus have $\Psi(\overline{w}) \in \mathsf{QFEAS}_R$ and Property (ii) entails $\Psi(\overline{w}) \in \mathsf{LASSO}_{AW}$. This entails $\Psi(\overline{w}) \in \mathsf{FEAS}_R$, and thus $\overline{w} \in \mathsf{SAT}_{k,X}$.

We first prove that Ψ satisfies Property (i). That is $\mathsf{SAT}_{k,X} = \Psi^{-1}(\mathsf{FEAS}_R)$.

Let $\overline{c} \in \mathsf{SAT}_{k,X}$. Then there exists $\overline{w} \in (Val_{X,\mathbb{D}})^\omega$ such that $\overline{w} \models \overline{c}$. We first build the input sequence for the action word (the sequence of data d_i) from $\overline{w} = w_1w_2\ldots$. Let $\overline{d} = \prod_{l=1}^{\infty}\prod_{j=0}^{k}\prod_{i=1}^{|X|} w_l(x_i^{(j)})$, we can now build by induction $\nu_0 : r \in R \to d_0$ and ν_{i+1} as the content of the registers upon realizing the action $\Psi(c)_i$, *i.e.* $\nu_i \xrightarrow{d_i, \Psi(c)_i} \nu_{i+1}$. There are three cases:

- when the test is $\top$, it is direct,
- when it is $x_{i,j+1} = *$, in which case it is enough to observe that $w_l(x_i^{(j+1)}) = w_{l+1}(x_i^{(j)})$ as long as $j < k$,
- lastly in the case where the test is $\bigwedge_{p(x_{i_1}^{(j_1)}\ldots x_{i_l}^{(j_l)})\in c_i} p(x_{i_1,j_1}\ldots x_{i_l,j_l})[x_{|X|,k} \leftarrow *]$, the content of the register has been fully updated except for $x_{|X|,k}$ but the substitution changes its value to the current data.

Then, each constraint holds in c_i if it holds in the test over the registers.

This gives us the left to right implication. The converse implication comes from the fact that any data sequence satisfying $\Psi(\overline{c})$ can be folded back to a sequence of valuations from $(Val_{X,\mathbb{D}})^\omega$, the $x_{i,j+1} = *$ tests ensure the repetition of each data k time. Once folded, the constraint in c_i is evaluated on the same data as in the action word hence satisfiability of $\overline{c}$ comes from the satisfiability of $\Psi(\overline{c})$.

We now want to prove that Ψ preserves ultimate periodicity, *i.e.* Property (ii).

Let $\overline{c} \in (\mathcal{MC}_X^{(k)})^\omega$ ultimately periodic. Then it can be rewritten as $c_1\ldots c_u(c_{u+1}\ldots c_{u+t})^\omega$. We will suppose $u > 1$ without loss of generality. Then we have:

$$\Psi(\overline{c}) = \Psi_{init}(c_1)\Psi_{next}(c_2)\ldots\Psi_{next}(c_u)(\Psi_{next}(c_{u+1})\ldots\Psi_{next}(c_{u+t}))^\omega$$

Which is ultimately periodic of period $t(k+1)|X|$.

Regarding complexity, one can observe that Ψ is realized by an FT $\mathbb{T}_\Psi$ with two states. As $\mathsf{QSAT}_{k,X} = \Psi^{-1}(\mathsf{QFEAS}_R)$, we can build an automaton accepting $\mathsf{QSAT}_{k,X}$ by doing a wreath product between $\mathbb{T}_\Psi$ and an automaton recognizing QFEAS_R, yielding the result, as we have $|R| = (k+1).|X|$.

◀

C Omitted proofs of Section 6

▶ **Lemma 34.** *Let $\overline{a} \in \mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$, $\forall \overline{a'} \in \mathsf{compl}_{\mathbb{I}_h}(\overline{a})$ and $\forall \overline{w}|_V \models \overline{a'}$, $\overline{w}|_{\mathbb{I}_v \cup \mathbb{O}} \models \overline{a}$*

Proof. Let $\overline{a} = (C_1, \rho_1^{ass}, \rho_1^{out}) \cdots \in \mathsf{AW}^{\omega}_{\mathbb{I},\mathbb{O},R}$, let $\overline{a'} \in \mathsf{compl}_{\mathbb{I}_h}(\overline{a})$ and let $\overline{w} \in Val_{V,\mathbb{D}}$ such that $\overline{w} \models \overline{a'}$. By def of $\mathsf{compl}_{\mathbb{I}_h}(\overline{a})$, $\overline{a'} = (C'_1, \rho_1^{ass}, \rho_1^{out}) \ldots$. Let $\overline{w^R} \in Val^{\omega}_{R,\mathbb{D}}$, the valuation word of the register along the execution of $\overline{a'}$ on $\overline{w}$. We will build by induction $\overline{x^R} \in Val^{\omega}_{R,\mathbb{D}}$ the valuation word of the register along the execution of $\overline{a}$ on $\overline{w}|_{\mathbb{I}_v \cup \mathbb{O}}$.

Let $i \geq 1$ suppose $w_i^R = x_i^R$. First we show that the valuation of this register valuation do satisfy tests of $\overline{a}$. By hypothesis $\overline{w} \models \overline{a'}$ so by semantics of action words, $w_i|_{\mathbb{I}} \times w_i^R \models C'_i$ by projection we have $w_i|_{\mathbb{I}_v} \cup w_i^R \models C'_i|_{\mathbb{I}_v \cup R}$ but $C_i = C'_i|_{\mathbb{I}_v \cup R}$ so $w_i|_{\mathbb{I}_v} \cup w_i^R \models C_i$ and by induction hypothesis $C_i = C'_i|_{\mathbb{I}_v \cup R}$ so $w_i|_{\mathbb{I}_v} \uplus x_i^R \models C_i$.

Then we show that $w_{i+1}^R = x_{i+1}^R$. Again by hypothesis $\overline{w} \models \overline{a'}$ so by semantics of action words, we have:

- let $r \in R$ if $\rho_i^{ass}(r)$ is defined we have $w_{i+1}^R(r) = w_i|_{\mathbb{I}} \circ \rho_i^{ass}(r)$ then by definition of $\mathsf{compl}_{\mathbb{I}_h}(\overline{a})$, we know that $\rho^{ass}(r) \in \mathbb{I}_v$ because it is initially an assignment of $\overline{a}$ so $w_{i+1}^R(r) = w_i|_{\mathbb{I}_v} \circ \rho_i^{ass}(r)$, and by semantics of $\overline{a}$, $x_{i+1}^R(r) = w_i|_{\mathbb{I}_v} \circ \rho_i^{ass}(r)$ so $x_{i+1}^R(r) = w_{i+1}^R(r)$
- let $r \in R$ if $\rho_i^{ass}(r)$ is not defined we have $w_i^R(r) = w_{i+1}^R(r)$ then $x_i^R(r) = x_{i+1}^R(r)$ do respect the semantics of $\overline{a}$ and as $x_i^R(r) = w_i^R(r)$, we get $x_{i+1}^R(r) = w_{i+1}^R(r)$

Lastly we need to ensure that the execution on $\overline{a}$ do output $\overline{w}|_{\mathbb{O}}$. We have by hypothesis $w_i|_{\mathbb{O}} = w_{i+1}^R \circ \rho^{out}$, and in last point we got $x_{i+1}^R(r) = w_{i+1}^R(r)$ so $w_i|_{\mathbb{O}} = x_{i+1}^R \circ \rho^{out}$. That allows us to conclude by semantics of action word that $\overline{w}|_{\mathbb{I}_v \cup \mathbb{O}} \models \overline{a}$

◀

▶ **Lemma 30** (Partial observation transfer Lemma)**.** *Let $\mathcal{A}$ be a universal co-Büchi CA. Then $L(\mathcal{A})$ is PO-realizable by a RT with $|R|$ registers iff $\mathsf{W}^{PO}_{\mathcal{A},R}$ is realizable by a FT.*

Proof. Let $\mathbb{D}$ be our data domain, $\mathcal{A}$ be a constraint automata, and $\mathbb{T}$ be a register transducer.

$\Rightarrow$ Suppose $\mathcal{A}$ is PO-realized by $\mathbb{T}$. We show that $\mathbb{T}_{stx}$ is the finite state transducer that realizes $\mathsf{W}^{PO}_{\mathcal{A},R}$, that is, $L(\mathbb{T}_{stx}) \subseteq \mathsf{W}^{PO}_{\mathcal{A},R}$.

Let $\overline{a} \in L(\mathbb{T}_{stx})$. Let $\overline{a'} \in \mathsf{compl}_{\mathbb{I}_h}(\overline{a})$, $\overline{c} \in (\mathcal{C}_V^{(k)})^{\omega}$ such that $\exists \overline{c'} \in \mathsf{join}(\overline{a'}, \overline{c})$ and $\overline{c'} \in \mathsf{SAT}_{k,V\cup R}$.

By definition, there is $\overline{w} \in (Val_{V\cup R,\mathbb{D}})^{\omega}$, such that $\overline{w} \models \overline{c'}$. We now need to show that $\overline{c} \in L(\mathcal{A}_{stx})$. Let ρ be a run of $L(\mathcal{A}_{stx})$ on $\overline{w}$. By Lemma 12 we have, $\overline{w}|_V \models \overline{a'}$ and $\overline{w}|_V \models \overline{c}$. We have $\overline{a'} \in \mathsf{compl}_{\mathbb{I}_h}(\overline{a})$ and $\overline{w}|_V \models \overline{a'}$ so by Lemma 34, $\overline{w}|_{\mathbb{I}_v \cup R} \models \overline{a}$. By hypothesis we have $\mathcal{A}$ is PO-realized by $\mathbb{T}$ so for all $\overline{w_h} \in Val_{\mathbb{I}_h,\mathbb{D}}$, $\overline{w}|_{\mathbb{I}_v \cup \mathbb{O}} \times \overline{w_h} \in L(\mathcal{A}_{stx})$. And in particular for $\overline{w_h} = \overline{w}|_{\mathbb{I}_h}$, so $\overline{w}|_V \in L(\mathcal{A}_{stx})$. Then ρ is an accepting run of $\mathcal{A}_{stx}$, that is $\overline{c} \in L(\mathcal{A}_{stx})$. So $\overline{a} \in \mathsf{W}^{PO}_{\mathcal{A},R}$, hence $L(\mathbb{T}_{stx}) \subseteq \mathsf{W}^{PO}_{\mathcal{A},R}$.

$\Leftarrow$ Suppose $\mathsf{W}_{\mathcal{A},R}$ is realized by $T = (Q, I, \Sigma_i, \Sigma_o, \Delta)$ a finite transducer on the finite alphabet of action word. So we know that:

- $\Sigma_i = \mathcal{MC}^{(k)}_{\mathbb{I}_v \cup \mathbb{O} \cup R}$
- $\Sigma_o = Assign_{\mathbb{I}_v,R} \times Val_{\mathbb{O},R}$
- $\Delta \subseteq Q \times \Sigma_i \to \Sigma_o \times Q$

We name $\mathbb{T}$ the R-register transducer over $\mathbb{D}$ generated by T (such that $T = \mathbb{T}_{stx}$) and show that $\mathbb{T}$ PO-realizes $\mathcal{A}$.

Let $\overline{w} \in L(\mathbb{T}) \subseteq (Val_{\mathbb{I}_v \cup \mathbb{O},\mathbb{D}})^{\omega}$ and $\overline{w_h} \in (Val_{\mathbb{I}_h,\mathbb{D}})^{\omega}$ a valuation of the hidden variable along the run. We let $\overline{a}$ be a constraint word generated by the run of T on $\overline{w}$. By definition, $\overline{a} \in L(\mathbb{T}_{stx})$. We can build the register valuation word along that run $\overline{w_r} \in (Val_{R,\mathbb{D}})^{\omega}$ by following the execution along the action word $\overline{a}$. We name $\overline{c}$ the constraint word generated by a run ρ of $\mathcal{A}$ on $\overline{w} \times \overline{w_h}$. Then we can build a maximally consistent constraint word $\overline{c'}$

over $V \cup R$ such that $\overline{w} \uplus \overline{w_h} \uplus \overline{w_r} \models \overline{c'}$ by choosing for every literal and its negation the one that $\overline{w} \uplus \overline{w_h} \uplus \overline{w_r}$ satisfies. Then $\overline{c'} \in \mathsf{SAT}_{k,V\cup R}$, by definition. We just said that $\overline{c}$ is a word generated by $\overline{w} \uplus \overline{w_h}$, so $\overline{w} \uplus \overline{w_h} \models \overline{c}$. But $\overline{a}$ is not of the right type so we will build a $\overline{a'} \in \mathsf{compl}_{\mathbb{I}_h}(\overline{a})$, by conserving the same assignment and output but by expanding the tests from variable $\mathbb{I}_v \cup R$ to $\mathbb{I} \cup R$ by taking the predicates satisfied by $\overline{w} \uplus \overline{w_h} \uplus \overline{w_r}$. This $\overline{a'}$ is indeed in $\mathsf{compl}_{\mathbb{I}_h}(\overline{a})$ and we also have by construction $\overline{w} \models \overline{ab'}$. Then by Lemma 12, $\overline{c'} = \mathsf{join}(\overline{a'}, \overline{c})$.

But we said that $\overline{a} \in L(\mathbb{T}_{stx})$, so $\overline{a} \in \mathsf{W}^{PO}_{\mathcal{A},R}$ by hypothesis. By definition of $\mathsf{W}^{PO}_{\mathcal{A},R}$, we have $\overline{c} \in L(\mathcal{A}_{stx})$. So $\overline{w} \uplus \overline{w_h} \in L(\mathcal{A})$ and finally $\mathbb{T}$ PO-realizes $\mathcal{A}$. ◀

Proof of Theorem 31

▶ **Theorem 31.** *Let $\mathcal{D}$ be a data domain with effective ω-regular satisfiability. Register-bounded partial observation synthesis from CLTL($\mathcal{D}$) is decidable. If, in addition, $\mathcal{D}$ is completable and decidable in* ExpTime*, then it is in 2*ExpTime*.*

Sketch of proof. The proof is similar to what we did in Section 4. We show that when the domain has effective ω-regular satisfiability, then we can describe the construction of a universal co-Büchi automaton accepting the set $\mathsf{W}^{PO}_{\mathcal{A},R}$, and control its size. More precisely, the difference between definitions of $\mathsf{W}_{\mathcal{A},R}$ and $\mathsf{W}^{PO}_{\mathcal{A},R}$ relies in the universal quantification over $\overline{a'} \in \mathsf{compl}_{\mathbb{I}_h}(\overline{a})$. This universal quantification fits well with the universal coBüchi condition we considered.

Then, the decidability follows from Büchi-Landweber's Theorem. Regarding complexity, the same reasoning applies. ◀