Timed pushdown automata revisited

Lorenzo Clemente
University of Warsaw

Sławomir Lasota
University of Warsaw

***Abstract*—This paper contains two results on timed extensions of pushdown automata (PDA). As our first result we prove that the model of dense-timed PDA of Abdulla *et al.* collapses: it is expressively equivalent to dense-timed PDA with *timeless* stack. Motivated by this result, we advocate the framework of first-order definable PDA, a specialization of PDA in sets with atoms, as the right setting to define and investigate timed extensions of PDA. The general model obtained in this way is Turing complete. As our second result we prove NEXPTIME upper complexity bound for the non-emptiness problem for an expressive subclass. As a byproduct, we obtain a tight EXPTIME complexity bound for a more restrictive subclass of PDA with timeless stack, thus subsuming the complexity bound known for dense-timed PDA.**

I. Introduction

Background. Timed automata [1] are a popular model of time-dependent behavior. A timed automaton is a finite automaton extended with a finite number of variables, called clocks, that can be reset and tested for inequalities with integers; so equipped, a timed automaton can read timed words, whose letters are labeled with real (or rational) timestamps. The value of a clock implicitly increases with the elapse of time, which is modeled by monotonically increasing timestamps of input letters.

In this paper, we investigate timed automata extended with a stack. An early model extending timed automata with an untimed stack, which we call *pushdown timed automata* (PDTA), has been considered by Bouajjani *et al.* [2]. Intuitively, PDTA recognize timed languages that can be obtained by extending an untimed context-free language with regular timing constraints. A more expressive model, called *recursive timed automata* (RTA), has been independently proposed (in an essentially equivalent form) by Trivedi and Wojtczak [3], and by Benerecetti *et al.* [4]. RTA use a timed stack to store the current clock valuation, which can be restored at the time of pop. This facility makes RTA able to recognize timed language with non-regular timing constraints (unlike PDTA).

More recently, *dense-timed pushdown automata* (dtPDA) have been proposed by Abdulla *et al.* [5] as yet another extension of PDTA. In dtPDA, a clock may be pushed on the stack, and its value increases with the elapse of time, exactly like the value of an ordinary clock. When popped from the stack, the value may be tested for inequalities with integers. The non-emptiness problem for dtPDA is solved in [5] by an ingenious reduction to non-emptiness of classical *untimed* PDA. As a byproduct, this shows that the untiming projection of dtPDA-language is context-free. Perhaps surprisingly, we prove the semantic collapse of dtPDA to PDTA, i.e., dtPDA with timeless stack but timed control locations: every dtPDA may be effectively transformed into a PDTA *that recognizes the same timed language*. Notice that this is much stronger than a mere reduction of the non-emptiness problem from the former to the latter model. Intuitively, the collapse is caused by the accidental interference of the LIFO stack discipline with the monotonicity of time, combined with the restrictions on stack operations assumed in dtPDA. Thus, dtPDA are equivalent to PDTA, and therefore included in RTA. The collapse motivates the quest for a more expressive framework for timed extensions of PDA.

Timed register pushdown automata. We advocate *sets with atoms* as the right setting for defining and investigating timed extensions of various classes of automata. This setting is parametrized by a logical structure $\mathbb{A}$, called *atoms*. Intuitively speaking, sets with atoms are very much like classical sets, but the notion of finiteness is relaxed to orbit-finiteness, i.e., finiteness up to an automorphism of atoms $\mathbb{A}$. The relaxation of finiteness allows to capture naturally various infinite-state models. For instance, ignoring some inessential details, register automata [6] (recognizing data languages) are expressively equivalent to the reinterpretation of the classical definition of 'finite NFA' as 'orbit-finite NFA' in sets with *equality atoms* $(\mathbb{N}, =)$ (see [7] for details), and analogously for register pushdown automata [8].

Along similar lines, timed automata (without stack) are essentially a subclass of NFA in sets with *timed atoms* $(\mathbb{Q}, \leq, +1)$, i.e., rationals with the natural order and the $+1$ function (see [9] for details). The automorphisms of timed atoms are thus monotonic bijections from $\mathbb{Q}$ to $\mathbb{Q}$ that preserve integer differences. In fact, to capture timed automata it is enough to work in a well-behaved subclass of sets with timed atoms, namely in *(first-order) definable sets*. Examples of definable sets are

$$A = \{(x,y,z) \in \mathbb{Q}^3 \ : \ x < y < z+1 < x+4\}$$
$$A' = \{(x,y) \in \mathbb{Q}^2 \ : \ x = y \ \vee \ y > x+2\}.$$

The first one is orbit-finite, while the other is not.

By reinterpreting the classical definition of PDA in definable sets we obtain a powerful model, which we call *timed register PDA* (trPDA), where, roughly speaking, a clock (or even a tuple of clocks) may be pushed to, and popped from the stack, conditioned by arbitrary clock constraints referring possibly

The first author acknowledges a partial support of the Polish National Science Centre grant 2013/09/B/ST6/01575.

The last author acknowledges a partial support of the Polish National Science Centre grant 2012/07/B/ST6/01497.

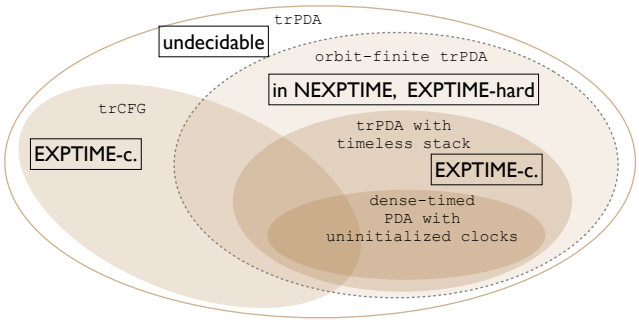

Fig. 1: Classes of timed pushdown languages.

to other clocks. Notice that monotonicity is not part of the definition of timed atoms, and thus in general trPDA read non-monotonic timed words, unlike classical timed automata or dense-timed PDA. This is not a restriction, since monotonicity can be checked by the automaton itself, and thus we can model monotonic as well as non-monotonic timed languages. An example language recognized by a trPDA (or even by trCFG) is the language of palindromes over the alphabet A defined above. Another example is the language of bracket expressions over the alphabet $\{[,]\}\times\mathbb{Q}$, where the timestamps of every pair of matching brackets belong to A'. These languages intuitively require a timed stack in order to be recognized, and thus fall outside the class of dtPDA due to our collapse result.

Contributions. In view of possible applications to verification of time-dependent recursive programs, we focus on the computational complexity of the non-emptiness problem for trPDA. We isolate several interesting classes of trPDA, which are summarized in Fig. 1. All intersections are non-trivial. Our model subsume dtPDA, for the simple reason that the finite-state control is essentially a timed-register NFA, which subsumes timed automata, i.e., the finite-state control of dtPDA. For the general model we prove undecidability of non-emptiness. This motivates us to distinguish an expressive subclass, which we call *orbit-finite* trPDA, which is obtained from the general model by imposing a certain orbit-finiteness restriction on push and pop operations. We show that non-emptiness of orbit-finite trPDA is in `NExpTime`. This is shown by reduction to non-emptiness of the least solution of a system of equations over sets of integers (cf. [10] and references therein). This reduction is the technical core of the paper. Moreover, it shows the essentially *quantitative* flavor of the dense time domain $(\mathbb{Q},\leq,+1)$ as opposed to other kind of atoms, like equality $(\mathbb{N},=)$ or total-order atoms $(\mathbb{Q},\leq)$. Note that $(\mathbb{R},\leq,+1)$ has the same first-order theory of the rationals, and thus considering the latter instead of the reals is with no loss of generality. Interestingly, our proofs work just as well over the discrete time domain $(\mathbb{Z},\leq,+1)$.

In order to establish the claimed complexity upper bound, we establish, along the way, tight complexity results for solving systems of equations in special form. From this analysis, we derive `ExpTime`-completeness of the subclass of trPDA with timeless stack. Due to our collapse result, under a simple technical assumption that preserves non-emptiness, dtPDA can be effectively transformed into trPDA with timeless stack, and thus we subsume the `ExpTime` upper bound shown in [5].

Finally, we consider the reinterpretation of context-free grammars in sets with timed atoms. We prove that timed context-free languages are a strict subclass of trPDA languages, and that their non-emptiness is `ExpTime`-complete.

Except for the technical results, the paper offers a wider perspective on modeling timed systems. We claim that sets with atoms have a significant and still unexplored potential for capturing timed extensions of classical models of computation.

Organization. In Sec. II we show the collapse result for dtPDA. In Sec. III we introduce the setting of definable sets. Then, in Sec. IV we define trPDA and its subclasses, formulate our complexity results, and relate in detail these results to the previously known `ExpTime`-completeness of dtPDA. The following Sec. V is the core technical part of the paper and it is devoted to the proofs of the upper bounds. The last section contains final remarks and sketch of future work. The missing parts of the proofs are delegated to the appendix.

II. Dense-timed pushdown automata

As the first result of the paper, we show that dtPDA as proposed by [5] recognize the same timed languages as its variant with timeless stack. This result is much stronger than the reduction proposed in [5], which shows that dtPDA and its variant with timeless stack are equivalent w.r.t. the *untimed* language (as opposed to the full timed language). In fact, we even prove this for a non-trivial generalization of the model of [5] with diagonal pop constraints (cf. below). In view of our collapse result, we abuse terminology and we also call the extended model dtPDA. A *clock constraint* over a set of clocks X is a formula φ generated by the following grammar:

$$\varphi \;::=\; \mathbf{t} \mid x \sim k \mid x - y \sim k \mid \varphi \wedge \varphi,$$

where $\mathbf{t}$ is the trivial constraint which is always true, $x, y \in X$, $k \in \mathbb{Z}$, and $\sim \in \{<, \leq, \geq, >\}$. We do not have disjunction $\vee$ since it can be simulated by nondeterminism in the transition relation of the automaton. We write $y - x \sim k \in \varphi$ to denote that $y - x \sim k$ is a conjunct in φ. A *dense-timed pushdown automaton* (dtPDA) is a tuple $\mathcal{T} = (L, l_0, \Sigma, \Gamma, X, z, \Delta)$ where L is a finite set of control locations, $l_0 \in L$ is the initial location, Σ is a finite input alphabet, Γ is a finite stack alphabet, X is a finite set of clocks, and z is a special clock not in X representing the age of the topmost stack symbol. The last item Δ is a set of transition rules of the form: $l \xrightarrow{a,\varphi,Y,op} l'$ with $l, l' \in L$ control locations, $a \in \Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ an input letter, φ a constraint over clocks in X, a subset $Y \subseteq X$ of clocks that will be reset, and *op* is either nop, $\mathsf{pop}(\alpha \models \psi_0)$, or $\mathsf{push}(\alpha \models \psi_1)$, where $\alpha \in \Gamma$ a stack symbol, ψ_0 a constraint over clocks in $X \cup \{z\}$ (called *pop constraint*) and ψ_1 a constraint over $\{z\}$ (called *push constraint*). An automaton has *timeless stack* if all its pop operations $\mathsf{pop}(\alpha \models \mathbf{t})$ have the trivial constraint $\mathbf{t}$, in which case just write $\mathsf{pop}(\alpha)$.

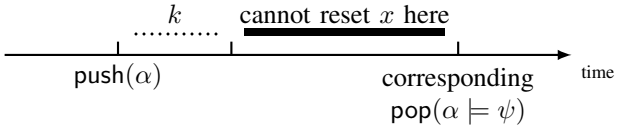

Fig. 2: Reset restriction on x when $z - x \lesssim k \in \psi$.

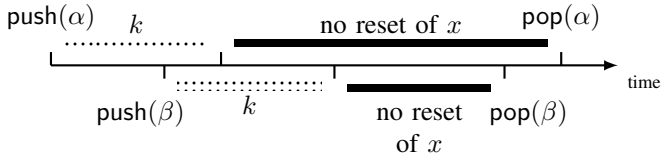

Fig. 3: Current reset restrictions always subsume new ones.

The formal semantics of dtPDA follows [5], and can be found in the appendix. Intuitively, every symbol on the stack carries a nonnegative rational number representing its age. Ages increase monotonically as time elapses, all at the same rate, and at the same rate as the other clocks of the automaton. Every time a new symbol is pushed on the stack, its age is nondeterministically initialized to a value of z satisfying the push constraint ψ_1, and it can be popped only if its current age satisfies the constraint ψ_0. Note that the push constraint ψ_1 essentially forces the initial age into a (possibly unbounded) interval. The original definition of [5] imposed the same restriction on pop constraints. Our definition of pop constraint is more liberal, since we allow more general *diagonal pop constraints* of the form $z - x \sim k$. Despite this seemingly more general definition, we show nonetheless that the stack can be made *timeless* while preserving the timed language recognized by the automaton.

Theorem II.1. *A dtPDA $\mathcal{T}$ can be effectively transformed into a dtPDA $\mathcal{U}$ with timeless stack recognizing the same timed language. Moreover, $\mathcal{U}$ has linearly many clocks w.r.t. $\mathcal{T}$, and exponentially many control locations.*

Proof (sketch). We explain here the basic idea of the transformation. The formal construction can be found in the appendix. W.l.o.g. we assume that: 1) Pop constraints are conjunctions of formulae of the form $z-x \sim k$, 2) transition rules involving a push or pop operation never reset clocks, and 3) the initial age of stack symbols pushed on the stack is always 0. These assumptions will simplify the construction; we show in the appendix how an automaton can be modified in order to satisfy them. The intuition is that a pop constraint of the form $z - x \lesssim k$ with $\lesssim \in \{<, \leq\}$ implies that clock x cannot be reset after k (possibly negative) time units of the push and before the corresponding pop. We call this a *reset restriction*; cf. Fig. 2. We call a pop constraint $z - x \lesssim k$ *active* if it has been guessed to hold at the time of a future pop. To keep track of reset restrictions, we carry in the control state a set R of tuples of the form $(x, \lesssim, k)$ for every active pop constraint. An extra clock $\hat{x}_{\lesssim k}$ which is reset at time of push is used to check $\hat{x}_{\lesssim k} \lesssim k$ whenever x is reset in order to guarantee that x is not reset too late. If $k \leq 0$, then we need to additionally check that x was not reset within the last $-k$ time units, which amounts to check $x \gtrsim -k$ at the time of push. The crucial observation is that, if a new reset restriction $(x, \lesssim, k)$ arises for an already active constraint, then we can safely ignore it since it is always subsumed by the current one. In other words, whenever the old restriction is satisfied, so is the new one, which is thus redundant; cf. Fig. 3.

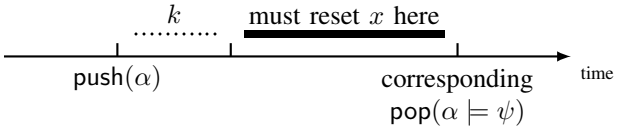

Fig. 4: Reset obligation on x when $z - x \gtrsim k \in \psi$.

The situation for a pop constraint of the form $z - x \gtrsim k$ with $\gtrsim \in \{>, \geq\}$ is dual, since it requires that clock x is reset after at least k (possibly negative) time units of the push and before its corresponding pop. We call this a *reset obligation*; cf. Fig. 4. We keep track of a set O of tuples $(x, \gtrsim, k)$ for every active pop constraint $z - x \gtrsim k$, meaning that clock x must be reset before the *next pop*. When x is reset after k time units of the push, we remove $(x, \gtrsim, k)$ from O. To verify the latter condition, we use an additional clock $\hat{x}_{\gtrsim k}$ which is reset at the time of push, and we check that $\hat{x}_{\gtrsim k} \gtrsim k$ holds. A new reset obligation with $k \leq 0$ is discarded if $-x \gtrsim k$ already holds at the time of push. A pop is allowed only if O is empty, i.e., all reset obligations have been satisfied. The crucial observation is that a new reset obligation $(x, \gtrsim, k)$ always subsumes one already in O, in the sense that, whenever the former is satisfied, so it is the latter; cf. Fig. 5. Thus, previous obligations are always discarded in favor of new ones (this is dual w.r.t. reset restrictions). When there is a new push, we have to additionally guess whether obligations in O not subsumed by new ones will be satisfied either before the matching pop, or after it. In the first case they are kept in O, while in the second case they are pushed on the stack in order to be put back into O at the matching pop. □

The construction uses ε-transitions, which simplifies substantially the encoding. A more complex construction not using ε-transitions can be given, and thus the collapse holds even for ε-free dtPDA. We don't know whether diagonal *push* constraints make the model more expressive, and, in particular, whether the stack can be untimed in this case. (This potentially more general model would still be subsumed by our orbit-finite trPDA from Sec. IV-B.)

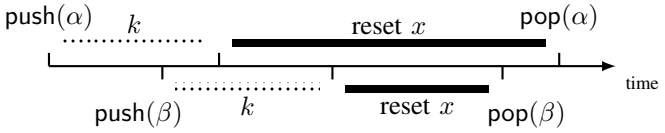

Fig. 5: New reset obligations always subsume current ones.

III. Definable sets and relations

In order to go beyond the recognizing power of dtPDA, we define automata that use timed registers instead of clocks. While a clock stores the difference between the current time and the time of its last reset, a timed register stores an absolute timestamp. Unlike ordinary clocks, timed registers are suitable for the modeling of non-monotonic time, and, even in the monotonic setting, they are more expressive since they can be manipulated with greater freedom than clocks. While in the semantics of clocks diagonal and non-diagonal constraints are inter-reducible [11], in the setting of timed registers only diagonal constraints are meaningful. Consequently, we drop non-diagonal constraints of the form $x \sim k$, and we redefine the notion of *constraint*, by which we mean a positive boolean combination of formulas $x - y \sim k$, where x, y are variables, $k \in \mathbb{Z}$, and $\sim \in \{<, \leq, \geq, >\}$. We use $=$ and $\neq$ as syntactic sugar. Constraints are expressively equivalent to the quantifier-free language of the structure $(\mathbb{Q}, \leq, +1)$, for instance

$$(x + 1 \leq y + 1 \ \wedge \ y \leq x) \ \vee \ \neg(x \leq (y + 1) + 1)$$

can be rewritten as a constraint $x = y \ \vee \ x - y > 2$. For complexity estimations we assume that the integer constants are encoded in binary.

A constraint φ over variables $x_1, \ldots, x_n$ defines a subset $[\varphi] \subseteq \mathbb{Q}^n$, assuming an implicit order on variables; n is called the *dimension* of φ, or $\dim \varphi$. In the sequel, we use disjoint unions of sets defined by constraints, and we call these sets *definable sets*. Formally, a definable set is an indexed set

$$X \quad = \quad \{X_l\}_{l \in L}, \tag{1}$$

where L is a finite index set and for every $l \in L$, the set $X_l = [\varphi_l]$ is defined by a constraint. $(L, \{\varphi_l\}_{l \in L})$ is a *constraint representation* of the set (1). When convenient we identify X with the disjoint union

$$\biguplus_{l \in L} X_l, \tag{2}$$

and write $\langle l, v \rangle \in X$ instead of $v \in X_l$. The automata in this paper will have definable state spaces. An index $l \in L$ may be understood as a control location, and a tuple $v \in \mathbb{Q}^n$ may be understood as a valuation of n registers (hence variables may be understood as register names). Under this intuition, φ_l is an invariant that constraints register valuations in a control location l. Similarly, an alphabet letter will contain an element of a finite set L, and a tuple $v \in \mathbb{Q}^n$ conforming to a constraint.

We do not assume that all component sets $[\varphi_l]$ have the same dimension (in particular, the number of registers may vary from one control location to another). Observe that when all dimensions are 0, the set (2) is a finite set of the same cardinality as the indexing set L. Those sets, as well their elements, we call *timeless*; the elements which are not timeless we call *timed*. When describing concrete definable sets we will omit the formal indexing; for instance, we will write

$$\{l, l', l''\} \ \uplus \ \mathbb{Q}^2 \qquad \text{or} \qquad \{l, l', l''\} \ \cup \ \{k\} \times \mathbb{Q}^2$$

for a set consisting of $\mathbb{Q}^2$ and three other elements.

Along the same lines we define definable (binary) relations. For two definable sets $X = \{X_l\}_{l \in L}$ and $Y = \{Y_k\}_{k \in K}$, a definable relation $R \subseteq X \times Y$ is an indexed set

$$R \quad = \quad \{R_{(l,k)}\}_{(l,k) \in L \times K}, \tag{3}$$

where the indexing set is the Cartesian product $L \times K$ and every set $R_{(l,k)}$ is defined by a constraint satisfying $R_{(l,k)} \subseteq X_l \times Y_k$; in particular, $\dim R_{(l,k)} = \dim X_l + \dim Y_k$.

Transition relations of automata will be definable relations in the sequel. The relation $R_{(l,k)}$ is a constraint on a transition from a control location l to another control location k: it prescribes how a valuation of registers in l *before* the transition may relate to a valuation of registers in k *after* the transition.

Likewise one defines relations of greater arities. Thus a constraint representation of an n-ary definable relation consists of n finite index sets $L_1, \ldots, L_n$, and formulas

$$\varphi_{(l_1, \ldots, l_n)}, \qquad \text{for } (l_1, \ldots, l_n) \in L_1 \times \ldots \times L_n.$$

Note that the number of indexes $(l_1, \ldots, l_n)$ may be exponential in n. When such a relation is input to an algorithm, the presentation is allowed to omit those formulas which define the empty set, i.e., $[\varphi_{(l_1, \ldots, l_n)}] = \emptyset$.

Remark: Constraints are as expressive as first-order logic of $(\mathbb{Q}, \leq, +1)$: similarly like a constraint, a first-order formula φ with free variables $x_1, \ldots, x_n$ *defines* the subset $[\varphi] \subseteq \mathbb{Q}^n$, and may be effectively transformed into an equivalent constraint ψ, namely one satisfying $[\varphi] = [\psi]$.

A. Orbit-finite sets

The setting of definable sets is a natural specialisation of the more general setting of *sets with timed atoms* $(\mathbb{Q}, \leq, +1)$. A *time automorphism*, i.e., an automorphism of timed atoms $(\mathbb{Q}, \leq, +1)$, is a monotonic bijection $\pi : \mathbb{Q} \to \mathbb{Q}$ preserving integer distances, i.e., $\pi(x+k) = \pi(x)+k$ for every $k \in \mathbb{Z}$. We consider only sets invariant under time automorphism, which are called *equivariant sets*[1]. In general, equivariant sets are infinite unions of orbits, where the *orbit* of an element e is

$$\text{orbit}(e) \ = \ \{\pi\, e \ : \ \pi \text{ time automorphism}\}.$$

We restrict our attention to *orbit-finite* sets, which are those equivariant sets that decompose into a *finite* union of orbits.

A time automorphism π acts on any element e by renaming all time values $t \in \mathbb{Q}$ appearing in e, but leaves the other

[1]For well-behaved atoms, like equality atoms, finitely supported sets can be considered. In case of timed atoms, we restrict ourself to equivariant sets, i.e., those which are supported by the empty set.

structure of e intact. For instance, it distributes on tuples and disjoint unions:

$$\begin{aligned}\pi(t_1,\ldots,t_n) &= (\pi(t_1),\ldots,\pi(t_n))\\ \pi(\langle l,v\rangle) &= \langle l,\pi(v)\rangle.\end{aligned}$$

Thus components X_l's of a definable set are preserved by time automorphisms, and independently partition into orbits.

As an example, the orbit of $(2, 3.3, -1.7)$ is the set $\{(x,y,x') \in \mathbb{Q}^3\} \, 1 < y - x < 2 \ \wedge \ y - x' = 5$. The set $\mathbb{Q}$ has only one orbit (i.e., $\text{orbit}(x) = \mathbb{Q}$ for every $x \in \mathbb{Q}$), but the set $\mathbb{Q}^2$ is already orbit-infinite, its orbits being of the form

$$\{(x,y) \ : \ x - y = z\} \quad \text{or} \quad \{(x,y) \ : \ z < x - y < z + 1\}$$

for every integer $z \in \mathbb{Z}$. Orbits in $\mathbb{Q}^n$ are definable by constraints; we call these constraints *minimal* as they define the inclusion-minimal nonempty equivariant subsets of $\mathbb{Q}^n$. Consequently, every orbit-finite subset of $\mathbb{Q}^n$ is definable. Further, every definable set is equivariant. On the other hand not every equivariant subset of $\mathbb{Q}^n$ is definable (e.g., the equivariant set $\{(x,y) : x - y \text{ is a prime number}\}$), and not every definable subset is orbit-finite, due to the orbit-infiniteness of $\mathbb{Q}^n$ [9].

In the sequel whenever we consider an orbit-finite set, we implicitly assume that it is a disjoint union of subsets of $\mathbb{Q}^n$, $n \geq 0$, and therefore definable.

Define the *span* of a tuple $v \in \mathbb{Q}^n$ with $n > 0$ as $\max v - \min v$, the difference between the maximal and the minimal value in v, and for $n = 0$, let the span be 0 by convention.

Lemma III.1. *An equivariant subset $X \subseteq \mathbb{Q}^n$ is orbit-finite if, and only if, it has uniformly bounded span, i.e., it admits a common bound on the spans of all its elements.*

For an orbit $O \subseteq X$, we will make the notational difference between *the orbit of* e, when $O = \text{orbit}(e)$ with $e \in X$, and *an orbit in* X, when $O \in \text{orbit}(X) := \{\text{orbit}(e) \mid e \in X\}$.

B. Normal form

We prove that every definable set can be transformed into a convenient normal form, which is like the classical partitioning into regions but without restricting to non-negative rationals.

We say that a tuple $v \in \mathbb{Q}^d$ admits a *gap* $g \in \mathbb{Q}$, $g > 0$, if the set of rationals appearing in v can be split into two non-empty sets $L, R \subseteq \mathbb{Q}$ such that

$$\max(L) + g = \min(R).$$

Let a *g-extension* of v be any tuple in $\mathbb{Q}^d$ obtained from v by adding a positive rational h to all elements of R appearing in v, regardless of the choice of sets L and R. (Subtracting h from all elements of L would be equivalent for our purposes as the sets we consider are closed under translations.)

Note that if v admits an integer gap k then all other tuples in $\text{orbit}(v)$ also do. If this is the case for some $k \in \mathbb{Z}$, let the k-extension of an orbit $O \subseteq \mathbb{Q}^d$ be the closure of O under k-extensions, i.e., the smallest set containing O that contains all k-extensions of all its elements. We will build on the specific weakness of constraints: a fixed constraint can not distinguish an orbit O from its k-extension when k is sufficiently large.

An extension (i.e., a k-extension for some integer k) of an orbit O is a definable set. Indeed, a defining constraint is obtained from the minimal constraint φ defining O by syntactically replacing certain equalities $=$ with inequalities $\leq$ (call this constraint extension of φ). For instance, consider

$$\varphi(x,y,z,w) \ \equiv \ 0 < y - x < 1 \ \wedge \ z - y = 7 \ \wedge \ w - z = 7;$$

its 7-extension is $0 < y - x < 1 \ \wedge \ z - y \leq 7 \ \wedge \ w - z \leq 7$.

Lemma III.2 (Normal Form Lemma)**.** *Every definable set X decomposes into a finite union of orbits $O \subseteq X$ and of extensions of orbits $O \subseteq X$. A decomposition can be effectively computed in* `ExpTime`.

For orbit-finite X, the lemma yields an effective enumeration of orbits in X, since extensions of orbits, being orbit-infinite, do not appear in the decomposition of X:

Corollary III.3. *A decomposition of an orbit-finite definable set X into orbits is computable in* `ExpTime`.

Example III.1. Consider the following set $X = \{(x,y,z) \in \mathbb{Q}^3 \mid 0 < y - x < 1 \wedge z - y > 3\}$. One possible decomposition of X consists of orbits in X that do not admit a gap larger than 4, and of 4-extensions of all orbits that admit a gap 4, for instance the 4-extension of the orbit $\{(x,y,z) \in \mathbb{Q}^3 \ : \ 0 < y - x < 1 \ \wedge \ z - y = 4\}$.

Thanks to the Normal Form Lemma, we define the *normal form* of constraints, i.e., disjunction of minimal constraints and extensions thereof. In the sequel we assume whenever convenient that the constraint representations of definable sets are already in normal form. The exponential blowup introduced by this transformation will combine well with the polynomial complexity w.r.t. the normal form representations, thus yielding the exponential time overall complexity.

A relevant property of normal form sets is that they admit easy computation of projections:

Lemma III.4 (Projection Lemma)**.** *Given a definable set $X \subseteq \mathbb{Q}^d$ in normal form, its projection onto a subset of coordinates $\{1 \ldots d\}$, in normal form, is computable in polynomial time.*

Indeed, projection distributes over disjunction, and projection of a minimal constraint, or of extension thereof, is computed essentially by elimination of variables.

IV. Timed register pushdown automata

We define a new model of timed PDA by reinterpreting the standard presentation of PDA in the setting of definable sets. Our approach generalized the approach of [9] where NFA were considered. Classical PDA can be defined in a number of equivalent ways. In the setting of this paper, the choice of definition will be critical for tractability. In the most general variant, a PDA $\mathcal{A}$ consists of a finite input alphabet A, a finite set of states Q, initial and final states $I, F \subseteq Q$, a finite stack alphabet S, and a finite set of transition rules

$$\rho \ \subseteq \ (Q \times S^*) \times A_\varepsilon \times (Q \times S^*),$$

where $A_\varepsilon = A \cup \{\varepsilon\}$. The semantics of a PDA is defined as usual. A transition rule $(q, v, a, q', v') \in \rho$ describes a transition which reads input a, changes state from q to q', pops a sequence of symbols v from the stack and replaces it by v'. Formally, the transitions of a PDA are between configurations $c, c' \in Q \times S^*$, and $(q, v, a, q', v') \in \rho$ induces a transition $c \xrightarrow{a} c'$ if $c = (q, vw)$ and $c' = (q', v'w)$ for some $w \in S^*$. Similarly, one defines unlabeled transitions $c \longrightarrow c'$, the reachability relation $c \longrightarrow^* c'$, runs, accepting runs (runs starting in a state from I with empty stack, and ending in a state from F with arbitrary stack), and the language $L(\mathcal{A}) \subseteq A^*$ accepted by a PDA $\mathcal{A}$.

We reinterpret the definition of PDA by dropping the finiteness of the components. Instead, we require Q, A, S, I and F to be orbit-finite (and, thus, definable), and the relation ρ to be definable. The *dimension* of a PDA is the maximal dimension of its states Q. These orbit-finiteness requirements are necessary to obtain a model with decidable emptiness, since it has been shown in [9] that having orbit-infinite states leads to undecidability already in NFA. Since Q is orbit-finite, by Lemma III.1 there exists a uniform bound on the span of every vector in Q.

Note that ρ, being definable, is necessarily a subset of

$$\rho \subseteq (Q \times S^{\leq n}) \times A_\varepsilon \times (Q \times S^{\leq m}), \tag{4}$$

for some $n, m \in \mathbb{N}$, where $S^{\leq n} = S_0 \cup S \cup S^2 \cup \ldots \cup S^n$, where $S_0 = \{\varepsilon\}$. The generalized model we call *timed register PDA* (trPDA). Most importantly, the semantics of trPDA is defined *exactly as* the semantics of classical PDA. We assume acceptance by final state. This is expressively equivalent to acceptance by empty stack, or by final state and empty stack.

By the *size* of a trPDA we mean the size of its constraint representation, i.e., the sum of sizes of all defining constraints, where we assume that integer constants are encoded in binary.

As already in the case of NFA, also for PDA imposing an orbit-finiteness restriction on ρ would be too restrictive, in the sense that the model would recognize a strictly smaller class of timed languages than with unrestricted ρ. Example IV.1 illustrates this, and shows the interaction between timed symbols in the stack, state, and input.

Example IV.1. Consider the input alphabet $A = [\varphi]$, where $\varphi(x, y) \equiv x < y < x + 4$, and the language L of even-length monotonic palindromes over A, i.e., $L = \{(u_1, v_1) \ldots (u_{2n}, v_{2n}) \in A^* \mid u_1 \leq \ldots \leq u_n$ and $(u_i, v_i) = (u_j, v_j)$ for every $1 \leq i \leq 2n$ and $j = 2n + 1 - i\}$. A trPDA recognizing this language has state space of dimension 1 (i.e., 1 register) $Q = \{i\} \uplus \{1\} \times \mathbb{Q} \uplus \{2, f\}$, with i and f the initial and final states, respectively. The stack alphabet $S = A \uplus \{\bot\}$ extends the input alphabet by the symbol $\bot$. There are three groups of ε-transition rules, namely

$$(i, \varepsilon, \varepsilon, (1, t), \bot), \qquad ((1, t), \varepsilon, \varepsilon, 2, \varepsilon), \qquad (2, \bot, \varepsilon, f, \varepsilon),$$

for any $t \in \mathbb{Q}$, used to initiate the first half, to change to the second half, and to finalize the second half of a computation of an automaton. In state $(1, t)$ the automaton pushes an input letter (u, v) to the stack, while checking for monotonicity $t \leq u$, as described by the transition rules, for $t \leq u$,

$$((1, t), \varepsilon, (u, v), (1, u), (u, v)) \in Q \times S^0 \times A \times Q \times S.$$

Finally, in state 2 the automaton pops a symbol (u, v) from the stack, while checking for equality with the input letter, as described by the transition rules:

$$(2, (u, v), (u, v), 2, \varepsilon) \in Q \times S \times A \times Q \times S^0.$$

Observe that we can not require the set ρ of transition rules to be orbit finite. Indeed, this would impose a bound on the span of tuples in ρ, in particular on the difference $u - t$ in the push transition rules, and therefore also on the differences $u_{i+1} - u_i$ between consecutive input letters.

The *non-emptiness problem* asks whether the language recognised by a given trPDA is non-empty. We observe that the problem is undecidable for general trPDA.

Theorem IV.1. *Non-emptiness of trPDA is undecidable.*

The undecidability of the general model motivates us to consider several restrictions of trPDA for which we can show decidability of the non-emptiness problem. We consider timed register context-free grammars in Sec. IV-A, orbit-finite trPDA in Sec. IV-B, and trPDA with timeless stack in Sec. IV-C.

A. Timed register context-free grammars

Context-free grammars are PDA with one state where each transition pops exactly one symbol off the stack. A *timed register context-free grammar* (trCFG) $\mathcal{G}$ consists of the following items: an orbit-finite set S of symbols, a starting symbol $I \in S$ which is initially pushed on the stack, an orbit-finite input alphabet A, and a definable set of productions

$$\rho \subseteq S \times A_\varepsilon \times S^*.$$

Acceptance is by empty stack, i.e., when all symbols are popped off the stack. We call languages recognized by trCFG *timed register context-free languages*.

Example IV.2. Let $A = \mathbb{Q}$, and consider the language L of timed palindromes of even length, i.e., $L = \{x_1 \cdots x_{2n} \in \mathbb{Q}^* \mid \forall (1 \leq i \leq n) \cdot x_i = x_{2n-i+1}\}$. This language can be recognized by a trCFG with symbols $S = \{1\} \uplus \{2\} \times \mathbb{Q}$ and productions of the form $\rho = \{(1, x, 1 \cdot (2, y)), (1, x, (2, y)), ((2, x), y, \varepsilon) \mid x, y \in \mathbb{Q} \cdot x = y\}$. We will see later that this language cannot be accepted by trPDA with timeless stack.

Define the *untiming* of a word $a_1 \ldots a_n \in A^*$ over an orbit-finite alphabet A as its projection to orbits $\mathrm{orbit}(a_1) \ldots \mathrm{orbit}(a_n) \in \Sigma^*$, where $\Sigma = \mathrm{orbits}(A)$. Untiming naturally extends to languages. In the lemma below we show that the untiming of a language of trCFG is context-free. This contrasts with languages of trPDA; cf. Example IV.3. Therefore, trCFG are weaker than general trPDA.

Lemma IV.2. *The untiming of a timed register context-free language is effectively context-free.*

Theorem IV.3. *Non-emptiness problem of trCFG is* ExpTime*-complete.*

B. Orbit-finite timed register PDA

We have seen that restricting trPDA to grammars yields a decidable model. In this section, we investigate another natural restriction of trPDA with decidable non-emptiness. A transition rule $(q, v, a, q', v') \in \rho$ splits naturally into its left-hand side (lhs) $(q, v) \in Q \times S^*$ and its right-hand side (rhs) $(q', v') \in Q \times S^*$. Let *orbit-finite trPDA* be the subclass of trPDA where the projections of ρ to both lhs's and rhs's, i.e., the following two sets

$$\{(q, v) \mid \exists a, q', v'.\ (q, v, a, q', v') \in \rho\}$$
$$\{(q', v') \mid \exists q, v, a.\ (q, v, a, q', v') \in \rho\},$$

are orbit-finite. By Lemma III.1 this means that both lhs's and rhs's have uniformly bounded span. We still *do not* require the whole relation ρ to be orbit-finite.

As long as the recognized language is considered, orbit-finite trPDA may be transformed into a convenient short form, with the transition rules split into

$$\begin{aligned} \rho &= \text{PUSH} \cup \text{POP}, \\ \text{PUSH} &\subseteq Q \times A_\varepsilon \times Q \times S, \\ \text{POP} &\subseteq Q \times S \times A_\varepsilon \times Q \end{aligned} \tag{5}$$

(thus one of lhs, rhs is a single state from Q, and the other is a pair from $Q \times S$) where the two sets

$$\{(q', s') \mid \exists q, a.\ \text{PUSH}(q, a, q', s')\}$$
$$\{(q, s) \mid \exists q, a.\ \text{POP}(q, s, a, q')\}$$

are orbit-finite. This short form easily enables the simulation of transition rules of the form $\text{NOP}(q, a, q') \in Q \times A_\varepsilon \times Q$ that do not operate on stack, by a push followed by a pop. The trPDA in Example IV.1 is in short form.

Lemma IV.4. *An orbit-finite trPDA can be transformed into a language-equivalent trPDA in short form* (5) *of polynomially larger size.*

Thus, from now on we always conveniently assume that an orbit-finite trPDA is given in short form. According to the following example, untiming of the language of an orbit-finite trPDA needs not be context-free.

Example IV.3. Consider the language L of palindromes over the timeless alphabet $A = \{a, b\}$ containing the same number of a's and b's. L can be recognized by a trPDA of dimension 1 with state space $Q = \{i\} \uplus \{1, 2\} \times \mathbb{Q} \uplus \{f\}$ and stack alphabet $S = \{a, b\} \uplus \{\bot\} \times \mathbb{Q}$. as follows. At the beginning, a rational $t \in \mathbb{Q}$ is guessed and $(\bot, t)$ is immediately pushed to the stack according to the transition rules:

$$(i, \varepsilon, (1, t), (\bot, t)) \in Q \times \{\varepsilon\} \times Q \times S, \qquad \text{for } t \in \mathbb{Q}.$$

Palindromicity of L is checked by pushing timeless symbols a, b on the stack in the first half of the computation, and by popping and matching them during the second half. Additionally, the value stored in the state is increased at each occurrence of a, and decreased at each occurrence of b, according to the transition rules:

$$\left\{ \begin{array}{l} ((1, t), a, (1, t+1), a), \\ ((1, t), b, (1, t-1), b) \end{array} \mid\ t \in \mathbb{Q} \right\} \subseteq Q \times A \times Q \times S$$
$$\left\{ \begin{array}{l} ((2, t), a, a, (2, t+1)), \\ ((2, t), b, b, (2, t-1)) \end{array} \mid\ t \in \mathbb{Q} \right\} \subseteq Q \times S \times A \times Q$$

At the end of the computation, it remains to check that the number of a's equals the number of b's. After the last timeless symbol is popped off the stack, on the bottom thereof we have $(\bot, t)$ where t is the original value stored there at the beginning of the computation. It suffices to pop this timed symbol with a transition rule:

$$((2, t), (\bot, t), \varepsilon, f) \ \in\ Q \times S \times \{\varepsilon\} \times Q, \qquad \text{for } t \in \mathbb{Q},$$

which checks equality with the value stored in the state.

As our second main result, we prove decidability of non-emptiness for orbit-finite trPDA:

Theorem IV.5. *Non-emptiness of orbit-finite trPDA is in* NExpTime.

Recall that we assume that integer constants appearing in constraint representation of a trPDA are encoded in binary. We prove the theorem in Sec. V by reducing non-emptiness of trPDA to non-emptiness of systems of equations over set of integers.

C. trPDA with timeless stack

To obtain a better complexity upper-bound, and for comparison with previous work, we identify the subclass of trPDA where the stack alphabet is timeless (i.e., finite). We call this subclass *trPDA with timeless stack*, which corresponds precisely to timed-register automata [9] augmented with a timeless stack (in the spirit of [2]). Observe that this is a subclass of orbit-finite trPDA, by the following observation:

Proposition IV.6. *Cartesian product of an orbit-finite set and a timeless one is orbit-finite.*

Thus, lhs and rhs are orbit-finite if Q is orbit-finite and S is timeless. This class is weaker than orbit-finite trPDA. Indeed, the automaton recognizing language L described in Example IV.3 is orbit-finite. On the other hand L is not recognized by a trPDA with timeless stack, due to the following:

Lemma IV.7. *Untiming of the language of trPDA with timeless stack is effectively context-free.*

Proof (sketch). Replace the state space Q by the set of orbits of Q (similarly to the region construction), and consider transitions between orbits, labelled with orbits of the input alphabet A, defined existentially. This operation does not preserve the timed language L recognized by the automaton in general, but it does preserve reachability properties, and in particular the untiming projection of L. Since the stack is timeless, no special care is needed to handle it. □

Languages of trPDA with timeless stack are thus a strict subclass of those of orbit-finite trPDA, even over finite alphabets. Moreover, languages of trCFG are incomparable with languages of trPDA with timeless stack. An example of trCFG language which is not recognized by trPDA with timeless stack is the language of timed palindromes from Example IV.2. This language clearly cannot be recognized with a timeless stack since it requires to remember unboundedly many possibly different timestamps. For the other inclusion, the example below shows a language recognized by a trPDA with timeless stack but not recognized by a trCFG.

Example IV.4. Take $A = \{c\} \times \mathbb{Q} \uplus \{a, b\}$, and consider the language

$$L = \{(c, x)\, w\, (c, y) \mid w \text{ palindrome over } \{a, b\}, y - x = |w|\}.$$

L can be recognized by a trPDA with timeless stack which stores x in a register, and then uses the untimed stack to check that w is a palindrome and incrementing the register at every letter. Finally, it checks that y equals the value of the register. It can be shown that L cannot be recognized by a trCFG by a standard pumping argument. Intuitively, a sufficiently long word $s \in L$ can be split into $s = uvwxy$ s.t. at least one of v and x is non-empty, and, for every $i \geq 0$, $s_i := uv^i wx^i y \in L$. Since s has only two timestamps (at the beginning and at the end), pumping cannot involve them. Thus, v and x are substrings of the palindrome w, and pumping necessarily changes its length, which contradicts $s_i \in L$.

As our last main result, we derive a tight upper complexity bound for trPDA with timeless stack.

Theorem IV.8. *Non-emptiness for trPDA with timeless stack is* `ExpTime`*-complete.*

Remark: It follows from the proof that non-emptiness of automata in normal form is decidable in time polynomial in its size and exponential in its dimension.

D. dtPDA as trPDA with timeless stack

Our definition of trPDA differs from dtPDA [5] in the same way as timed register automata of [9] differ from classical timed automata [1]. The first difference is semantic: dtPDA (like timed automata) recognize timed languages where each input symbol carries only a single time-stamp. In this sense, they correspond to trPDA with a 1-dimensional input alphabet.

Moreover, languages of trPDA are closed under translations $x \mapsto x + t$, for $t \in \mathbb{Q}$, while languages of dtPDA are not. In order to fairly compare the two models, we assume (along the lines of [9]) that a dtPDA starts its computation with *uninitialized clocks*, instead of all clocks initialized with 0. This is not a restriction since a dtPDA $\mathcal{T}$ can be faithfully simulated by a dtPDA with uninitialized clocks $\mathcal{T}'$. For instance, as the first step, $\mathcal{T}'$ may initialize all its clocks with the timestamp of the first input letter (a, t) and then proceed as $\mathcal{T}$, and thus $L(\mathcal{T}') = \bigcup_{t \in \mathbb{Q},\, a \in \Sigma} (a, t)\, (L(\mathcal{T}) + t)$. This transformation clearly preserves non-emptiness.

Lemma IV.9. *A dtPDA with uninitialized clocks and timeless stack $\mathcal{T}$ can be effectively transformed into a language-equivalent normal form trPDA $\mathcal{A}$ with timeless stack. If $\mathcal{T}$ has n clocks then the dimension of $\mathcal{A}$ is $n + 1$ and its size is exponential in n.*

We sketch the construction. By definition, dtPDA accept monotonic words, while languages recognized by 1-dimensional trPDA are non-monotonic in general. Notice that monotonicity of input can be enforced by a trPDA by adding an additional special register x_0 in every control state, to store the timestamp of the last input, and by intersecting the transition rules with the additional constraint $x_0 \leq x_0'$ relating the values of the special register before and after a transition.

The most substantial difference is that dtPDA use *clocks*, while trPDA use *registers*. A dtPDA has clocks which can be reset and can be compared to an integer constant $x \sim k$, or, in the case of diagonal constraints, a difference of clocks is compared to an integer constant $x - y \sim k$. A trPDA can simulate a dtPDA by having one register $\hat{x}$ for each clock x. A reset of x is simulated by assigning the current input timestamp t to $\hat{x}$; a constraint $x \sim k$ is simulated by $x_0 - \hat{x} \sim k$ (where x_0 is the special register discussed above), and a diagonal constraint $x - y \sim k$ is simulated by $\hat{y} - \hat{x} \sim k$. (The ages for timed stack symbols could be treated similarly. This step is unnecessary for dtPDA with timeless stack.)

To obtain a trPDA we need to ensure that the set of states is orbit-finite. This is done as follows. Let m be the maximal absolute value of a constant in any constraint of a dtPDA. We perform the classical region construction of the dtPDA, and take regions as control locations of the trPDA. In every control location, the defining constraint is the intersection of the region with the constraint $\bigwedge_{x \in X} 0 \leq x_0 - x \leq m$, which makes the set of states orbit-finite. Additionally in every region, those registers that correspond to unbounded clocks are projected away. This is correct as the truth value of transitions constraints involving unbounded clocks does not depend on further elapse of time. This completes the sketch of the construction claimed in Lemma IV.9.

By Theorem II.1, we can remove time from the stack of a dtPDA with a single exponential blowup in the number of control locations (w.r.t. the size of pop constraints), and a linear increase in the number of clocks. By Lemma IV.9, we obtain a trPDA with a further exponential blowup in the number of control locations (w.r.t. number of clocks). Notice that the two blowups compose to a single exponential blowup, as summed up in the following corollary:

Corollary IV.10. *A dtPDA with uninitialized clocks can be effectively transformed into a language-equivalent normal form trPDA with timeless stack of exponential size (w.r.t. pop constraints and clocks) and linear dimension.*

In turn, the blowups in the last corollary and in Theorem IV.8 compose again to a single exponential blowup. Therefore Theorem IV.8 yields the `ExpTime` upper-bound for dtPDA and thus strengthens the `ExpTime` upper bound of [5].

V. Upper bounds

We prove the upper bounds of Theorems IV.5 and IV.8.

A. Equations over sets of integers

We consider systems of equations, interpreted over sets of integers, of the following form

$$\begin{aligned} X_1 &= t_1 \\ &\dots \\ X_n &= t_n, \end{aligned}$$

one for each variable X_i, where right-hand side expressions $t_1, \dots, t_n$ use variables $X_1 \dots X_n$ appearing in left hand sides, constants $\{-1\}, \{0\}, \{1\}$, union $\cup$, intersection $t \cap \{0\}$ with the constant $\{0\}$, and element-wise addition of sets of integers, $X + Y = \{x + y : x \in X \text{ and } y \in Y\}$. Note that the use of intersection is assumed to be very limited; for systems of equations with unrestricted intersection (e.g., $X \cap Y$), the non-emptiness problems is undecidable [12].

A solution ν of a system of equations assigns to every variable X a set $\nu(X) \subseteq \mathbb{Z}$ of integers. We are only interested in the least solution. Note that intersection and addition distribute over union, in the sense that $(t_0 \cup t_1) \cap t_2 = (t_0 \cap t_2) \cup (t_1 \cap t_2)$, and $(t_0 \cup t_1) + t_2 = (t_0 + t_2) \cup (t_1 + t_2)$. Thus, as long as the least solution is considered, a system of equations may be equivalently presented by a set of inclusions $X \supseteq t$, where t does not use union, with the proviso that many inclusions may apply to the same left-hand side variable X.

Example V.1. For instance, the set of all integers is the least solution for Z below; we can also succinctly represent large constants $m \in \mathbb{Z}$ as the least solution $\{m\}$ for $Z_{=m}$:

$$\begin{aligned} Z &\supseteq \{0\} \\ Z &\supseteq \{1, -1\} + Z \end{aligned} \qquad \begin{aligned} Z_{=0} &\supseteq \{0\} \\ Z_{=2m} &\supseteq Z_m + Z_m \\ Z_{=2m+1} &\supseteq Z_m + Z_m + \{1\}. \end{aligned}$$

Infinite intervals of the form $\mathbb{Z}_{<m} = (-\infty, m)$ and $\mathbb{Z}_{>m} = (m, \infty)$ are easily expressible as the least solutions of $Z_{<m}$ and $Z_{>m}$ in

$$\begin{aligned} Z_{<m} &\supseteq Z_{=(m-1)} \\ Z_{<m} &\supseteq Z_{<m} + \{-1\} \end{aligned} \quad \text{and} \quad \begin{aligned} Z_{>m} &\supseteq Z_{=(m+1)} \\ Z_{>m} &\supseteq Z_{>m} + \{1\} \end{aligned}$$

We will use these definitions later in this section.

By introducing additional auxiliary variables, one may easily transform the inclusions into the following binary form:

$$X \supseteq \{k\} \qquad X \supseteq Y \cap \{0\} \qquad X \supseteq Y + Z, \tag{6}$$

where k is $-1, 0$ or 1. For future reference we distinguish a subclass of *intersection-free* systems of equations which use no intersection. All equations in the previous Example V.1 are of this form.

The *non-emptiness problem* asks, for a given system Δ of equations and a variable X therein, whether the least solution ν of Δ assigns to X a non-empty set of integers. The *membership problem* asks, given an additional integer $k \in \mathbb{Z}$ (coded in binary), whether $k \in \nu(X)$.

Lemma V.1. *The non-emptiness problem for intersection-free systems of equations is in P.*

Proof. If Δ is intersection-free, its non-emptiness reduces to non-emptiness of a context-free grammar over three letters $\{-1, 0, 1\}$. Variables of Δ are non-terminal symbols, and every inclusion gives raise to one production. Addition is replaced by concatenation. □

Lemma V.2. *The non-emptiness and membership problems of systems of equations are both NP-complete. The membership problem is NP-hard already for intersection-free systems.*

B. From trPDA to systems of equations

We show an `ExpTime` reduction of non-emptiness of orbit-finite trPDA to non-emptiness of systems of equations. Additionally, if the stack is timeless, then the system of equations is intersection-free. Fix an orbit-finite trPDA $\mathcal{A}$, with states Q, stack alphabet S, and transition rules PUSH and POP.

As a preprocessing we apply few simplifying transformations. First, we rebuild $\mathcal{A}$ so that it has exactly one (therefore timeless) initial state, and exactly one final state. Therefore there are unique initial and final control locations, corresponding to the unique timeless initial and final state. Moreover, in the final state we let $\mathcal{A}$ unconditionally pop all symbols from the stack, and assume w.l.o.g. that $\mathcal{A}$ accepts when not only it is in the final state, but additionally the stack is empty. As the next step of preprocessing, we make all states of $\mathcal{A}$ timed, by adding to every timeless state (including the initial and final one) one dummy timed register. In order to assure orbit-finiteness of $\mathcal{A}$, appropriate additional constraints on the dummy registers are added to PUSH and POP. Thus the transformations described by now preserve orbit-finiteness of $\mathcal{A}$ can be done using its constraint representation. As the last step of preprocessing, we transform $\mathcal{A}$ into normal form. According to Lemma III.2 this is doable in `ExpTime`.

Reachability relation. As we focus on reachability, we ignore the input alphabet and assume the transition rules of $\mathcal{A}$ to be unlabeled, i.e., of the form

$$\text{PUSH}(q, q', s') \quad \text{and} \quad \text{POP}(q, s, q'),$$

where $q, q' \in Q$ and $s' \in S$. Consequently, we assume also unlabeled transitions $c \longrightarrow c'$ between configurations. Using the Projection Lemma III.4, the unlabeled transition rules are easily computed by projecting away the input alphabet.

We define the following binary reachability relation between states of $\mathcal{A}$. Two states are related, written $q \rightsquigarrow q'$, if there is a computation of $\mathcal{A}$ from state q to q' which starts and ends with empty stack. Formally, $q \rightsquigarrow q'$ if for some configurations $(q_1, v_1), \dots, (q_n, v_n)$, $\mathcal{A}$ admits the transitions:

$$(q, \varepsilon) \longrightarrow (q_1, v_1) \longrightarrow \dots \longrightarrow (q_n, v_n) \longrightarrow (q', \varepsilon). \tag{7}$$

It might be the case that $v_i = \varepsilon$ for some $1 \leq i \leq n$.

Proposition V.3. *$L(\mathcal{A})$ is non-empty iff $\rightsquigarrow$ relates an initial state with a final one.*

Lemma V.4. *The relation $\rightsquigarrow$ is the least relation satisfying the following rules:*

$$\text{(base)} \quad \frac{}{q \rightsquigarrow q} \qquad \forall\ (q,q) \in Q^2$$

$$\text{(transitivity)} \quad \frac{q \rightsquigarrow q' \quad q' \rightsquigarrow q''}{q \rightsquigarrow q''} \qquad \forall\ (q,q',q'') \in Q^3$$

$$\text{(push-pop)} \quad \frac{\bar{q} \rightsquigarrow \bar{q}'}{q \rightsquigarrow q'} \qquad \forall\ (q,\bar{q},\bar{q}',q') \in \text{PUSH-POP}$$

where PUSH-POP *is the subset of Q^4 defined as:*

$$\text{PUSH-POP} = \left\{ (q,\bar{q},\bar{q}',q') \mid \exists \bar{s} \in S.\ \begin{matrix} \text{PUSH}(q,\bar{q},\bar{s}), \\ \text{POP}(\bar{q}',\bar{s},q') \end{matrix} \right\} \quad (8)$$

Orbitization. Recall that the transition rules PUSH and POP are equivariant, i.e., are unions of orbits, possibly infinitely many. It follows that the relation $\rightsquigarrow \subseteq Q^2$ is also equivariant, i.e., a union of orbits of Q^2. Call an orbit $O \subseteq Q^2$ *inhabited* if $q \rightsquigarrow q'$ for some $(q,q') \in O$. If this is the case, since $\rightsquigarrow$ is equivariant, and thus a union of orbits, then *every* pair $(q,q') \in O$ satisfies $q \rightsquigarrow q'$. It thus makes sense to think of $\rightsquigarrow$ as containing whole orbits rather than individual elements. Let *initial-final* orbits in Q^2 be the ones containing pairs (i,f) for i initial and f final state; these orbits are determined by the unique initial and final control locations.

Proposition V.5. *$L(\mathcal{A})$ is non-empty iff an initial-final orbit in Q^2 is inhabited.*

Likewise, the relation PUSH-POP $\subseteq Q^4$ is equivariant, i.e., a union of possibly infinitely many orbits in Q^4. Our aim now is to 'orbitize' the rules of Lemma V.4 so that they speak of *orbits* of pairs of states, instead of individual pairs of states, without losing any precision.

The (base) rules orbitizes easily; it speaks of *diagonal* orbits, i.e., orbits of diagonal pairs $(q,q) \in Q^2$. For treating the other rules, we need to speak of projections of n-tuples w onto two coordinates. We use the notation w_{ij} to denote the projection of w onto coordinates i,j, for $1 \le i < j \le n$; the same notation will be used for the projection of a set of tuples. For O an orbit in Q^n, O_{ij} is necessarily an orbit in Q^2.

Lemma V.6. *An orbit O of Q^2 is inhabited if, and only if, $\vdash O$ is derivable according to the rules below:*

$$\text{(orbit base)} \quad \frac{}{\vdash O} \qquad \forall\ \textit{diag. orbit } O \textit{ in } Q^2$$

$$\text{(orbit transitivity)} \quad \frac{\vdash O_{12} \quad \vdash O_{23}}{\vdash O_{13}} \qquad \forall\ \textit{orbit } O \textit{ in } Q^3$$

$$\text{(orbit push-pop)} \quad \frac{\vdash O_{23}}{\vdash O_{14}} \qquad \forall\ \textit{orbit } O \textit{ in } \text{PUSH-POP}$$

Proof. Both directions are proved by induction on the size of derivations. The "if" direction uses equivariance of $\rightsquigarrow$. □

Discretization. The set Q^2 is orbit-infinite. We encode it as a Cartesian product of an orbit-finite set and the integers $\mathbb{Z}$. This will allow us to reduce non-emptiness of $\mathcal{L}(\mathcal{A})$ to non-emptiness of a system of equations.

Consider two states $q = \langle l, v\rangle, q' = \langle l', v'\rangle \in Q$, where $v \in \mathbb{Q}^{n_l}$ and $v' \in \mathbb{Q}^{n_{l'}}$. Since Q is orbit finite, by Lemma III.1 we know that both v and v' have uniformly bounded span, say u. However, the joint vector $(v,v') \in \mathbb{Q}^{n_l+n_{l'}}$ needs not have uniformly bounded span (and indeed Q^2 is orbit-infinite), since rationals in v might be arbitrarily far from rationals in v'. The idea is to "factorize" out the orbit infiniteness of Q^2 by shifting the second vector v' closer to v (in order to have span at most $u+1$), and by keeping track separately of the shift as the only unbounded component.

The first technical step is to extend the tuple v in every state $q = \langle l, v\rangle \in Q$ with one rational number t, written $q \cdot t = \langle l, (v,t)\rangle$, called the *reference point* of $q \cdot t$. Reference points allow to precisely shift vectors so they become closer. Let $\min v$ be the component of v with minimal value. We define

$$\dot{Q} = \{ q \cdot t \mid q = \langle l, v\rangle \in Q, t \in \mathbb{Q}, \min v \le t < \min v + 1 \}.$$

The set of extended tuples $\dot{Q}$ is definable and orbit-finite (of uniform span at most $u+1$), and contains exponentially many orbits. While $\dot{Q}^2$ is not orbit-finite itself, we can now define its subset $\ddot{Q}$ of pairs with equal reference points:

$$\ddot{Q} = \left\{ (q \cdot t, q' \cdot t) \in \dot{Q}^2 \mid t \in \mathbb{Q} \right\}.$$

Thus, $\ddot{Q}$ contains only those pairs of vectors which are close in a precise sense. Applying Corollary III.3 to $\ddot{Q}$ we obtain:

Proposition V.7. *$\ddot{Q}$ is orbit-finite with uniform span $u+1$ and its decomposition into orbits is computable in* ExpTime.

The idea now is to represent an arbitrary pair in Q^2 as an element from $\ddot{Q}$ plus an integer representing "shift" of the second vector. Formally, we define the following *shift mapping* $\pi : \ddot{Q} \times \mathbb{Z} \to Q^2$:

$$\pi\ :\ (q \cdot t, q' \cdot t), z\ \mapsto\ q, q' + z,$$

where $q' + z$ is the state obtained from $q' = \langle l', v'\rangle$ by adding z to all time values in v'. Thus the shift mapping forgets about the equal reference points of q and q', and shifts q' by z. Note that every pair of states in Q^2 is of the form $(q, q'+z)$, for some $z \in \mathbb{Z}$ and $(q \cdot t, q' \cdot t) \in \ddot{Q}$, i.e., the shift mapping is surjective. To distinguish between orbits of Q^2 and $\ddot{Q}$, we use lowercase o for the latter. Every orbit O of Q^2 is the image, under the shift mapping π, of $o \times \{z\}$, for some $z \in \mathbb{Z}$ and some orbit o of $\ddot{Q}$. We will call O the *image orbit* of (o,z). By the *inverse image* of an equivariant set $X \subseteq Q^2$ we mean the set of all pairs (o,z) whose image orbit O is included in X. We will call (o,z) *inhabited* if its image orbit is so.

The inverse image of an orbit O may contain many pairs (o,z), as shown in the example below, but finitely many due to the simplifying assumption that all states are timed.

Example V.2. Consider the orbit $O \subseteq Q^2$ defined by $\varphi(x,y,x') \equiv x < y < x+1 \ \wedge\ y+6 < x' < x+7$, with x,y timed registers of one state and x' timed register

of the other. The inverse image of O contains the pair $(o, 6)$, where $o \subseteq \ddot{Q}$ is defined by $x < y < t = t' = x' < x + 1$, but also the pair $(o', 7)$, where $o' \subseteq \ddot{Q}$ is defined by $y - 1 < x' < x = t = t' < y$.

The inverse image of a definable set admits a decomposition into finitely many sets of a particularly simple form:

Lemma V.8 (Decomposition Lemma)**.** *For a definable subset $X \subseteq Q^2$, its inverse image decomposes into a finite union of sets of the form*

$$\{o\} \times I,$$

where o is an orbit in $\ddot{Q}$, and $I \subseteq \mathbb{Z}$ is one of

$$\mathbb{Z}_{<m} = \{z : z < m\}, \quad \{m\}, \quad \mathbb{Z}_{>m} = \{z : z > m\},$$

for $m \in \mathbb{Z}$. A decomposition of X is computable in ExpTime.

The following corollary will be useful later:

Proposition V.9. *The inverse image of an orbit $O \subseteq Q^2$ is finite and computable in* ExpTime.

We are going to define a system of equations Δ, with variables X_o corresponding to orbits o in $\ddot{Q}$. The construction will conform to the following correctness condition:

Lemma V.10. *The least solution ν of the system Δ assigns to a variable X_o the set*

$$\nu(X_o) \;=\; \{z \in \mathbb{Z} \;:\; (o, z) \text{ is inhabited}\}.$$

Orbits $o \subseteq \ddot{Q}$ that appear in the inverse image of an initial-final orbit $O \subseteq Q^2$ we call initial-final too; again, they are determined by the unique initial and final control locations. Based on the last lemma, we reformulate Proposition V.5 as:

Proposition V.11. *$L(\mathcal{A})$ is non-empty iff $\nu(X_o)$ is non-empty, for an initial-final orbit o.*

Thus non-emptiness of $L(\mathcal{A})$ reduces in ExpTime to non-emptiness of some of the variables X_o in Δ.

To complete the proofs of upper bounds of Theorems IV.5 and IV.8, we need to describe the construction of Δ and prove that it verifies the condition in Lemma V.10.

System of equations. When defining Δ we prefer to use inclusions. Roughly speaking, the system Δ corresponds to the inverse image of the rules in Lemma V.6.

Consider the (orbit base) rule first. We observe that all orbits o appearing in the inverse image of a diagonal orbit O are diagonal as well. Thus for every diagonal orbit o in $\ddot{Q}$ we add to Δ the inclusion

$$X_o \;\supseteq\; \{0\}. \tag{9}$$

For treating the (orbit transitivity) rule we need to extend the shift mapping π from pairs to triples. Define the set of triples of states with equal reference points

$$\dddot{Q} \;=\; \{(q \cdot t, q' \cdot t, q'' \cdot t) \in \dot{Q}^3 \;:\; t \in \mathbb{Q}\},$$

and consider the shift mapping $\pi : \dddot{Q} \times \mathbb{Z}^2 \to Q^3$:

$$(q \cdot t, q' \cdot t, q'' \cdot t), z, z' \;\mapsto\; q, q' + z, q'' + z + z'.$$

As before, π transforms a triple (o, z, z'), where o is an orbit in $\dddot{Q}$, into an orbit O in Q^3. For an orbit O in Q^3, consider any element (o, z, z') of its inverse image, i.e., O is the image of (o, z, z'). The image commutes with projections, i.e., O_{12} is necessarily the image of (o_{12}, z), and likewise O_{23} and O_{13} are images of (o_{23}, z') and $(o_{13}, z+z')$, respectively. Therefore the (orbit transitivity) rule says that if (o_{12}, z) and (o_{23}, z') are inhabited, then $(o_{13}, z + z')$ is inhabited too. Thus, for every orbit o in $\dddot{Q}$ we add the following inclusion to Δ:

$$X_{o_{13}} \;\supseteq\; X_{o_{12}} + X_{o_{23}}. \tag{10}$$

Finally, we address the (orbit push-pop) rule. We consider separately two cases, depending on whether the stack symbol pushed/popped is timeless or timed. Each of the two cases will induce separate inclusions in Δ. Let S be partitioned into timeless stack symbols S_0 and timed stack symbols S_1. S_0 is a finite set. We partition PUSH-POP into PUSH-POP^0 and PUSH-POP^1, where

$$\text{PUSH-POP}^0 = \left\{ (q, \bar{q}, \bar{q}', q') \mid \exists \bar{s} \in S_0.\ \begin{matrix} \text{PUSH}(q, \bar{q}, \bar{s}), \\ \text{POP}(\bar{q}', \bar{s}, q') \end{matrix} \right\}$$
$$\text{PUSH-POP}^1 = \left\{ (q, \bar{q}, \bar{q}', q') \mid \exists \bar{s} \in S_1.\ \begin{matrix} \text{PUSH}(q, \bar{q}, \bar{s}), \\ \text{POP}(\bar{q}', \bar{s}, q') \end{matrix} \right\}$$

First, we consider the (orbit push-pop) rule restricted to only timeless stack symbols. We can write PUSH-POP^0 as a finite sum of products

$$\text{PUSH-POP}^0 = \bigcup_{\bar{s} \in S_0} \text{PUSH}_{\bar{s}} \times \text{POP}_{\bar{s}},$$

where $\text{PUSH}_{\bar{s}}(q, \bar{q}) \equiv \text{PUSH}(q, \bar{q}, \bar{s})$ and $\text{POP}_{\bar{s}}(\bar{q}', q') \equiv \text{POP}(\bar{q}', \bar{s}, q')$. For a fixed $\bar{s} \in S_0$, $\text{PUSH}_{\bar{s}}$ and $\text{POP}_{\bar{s}}$ are definable subsets of Q^2, and thus Lemma V.8 applies.

We need to extend once more the shift mapping π, this time to quadruples. Define the set of quadruples of states with equal reference points

$$\ddddot{Q} \;=\; \{(q \cdot t, \bar{q} \cdot t, \bar{q}' \cdot t, q' \cdot t) \in \dot{Q}^4 \;:\; t \in \mathbb{Q}\},$$

and consider the shift mapping π from $\ddddot{Q} \times \mathbb{Z}^3$ to Q^4:

$$(q \cdot t, \bar{q} \cdot t, \bar{q}' \cdot t, q' \cdot t), z, \bar{z}, z' \;\mapsto\; q, \bar{q} + z, \bar{q}' + z + \bar{z}, q' + z + \bar{z} + z'.$$

Similarly as before, π transforms a quadruple $(o, z, \bar{z}, z')$, where o is an orbit in $\ddddot{Q}$, into an orbit O in Q^4. Similarly as before we define the inverse image of $O \subseteq Q^4$.

The (orbit push-pop) rule says that if $(o_{23}, \bar{z})$ is inhabited, (o_{12}, z) belongs to the inverse image of $\text{PUSH}_{\bar{s}}$ and (o_{34}, z') belongs to the inverse image of $\text{POP}_{\bar{s}}$, then $(o_{14}, z + \bar{z} + z')$ is inhabited. Therefore for every orbit $o \subseteq \ddddot{Q}$ appearing in the inverse image of PUSH-POP, for every $\bar{s} \in S_0$, for every pair of intervals I, I' such that (o_{12}, I) appears in the decomposition of $\text{PUSH}_{\bar{s}}$ and (o_{34}, I') appears in the decomposition of $\text{POP}_{\bar{s}}$ (by Lemma V.8), we add to Δ the inclusion

$$X_{o_{14}} \;\supseteq\; X_{o_{23}} + \; Z_{I+I'}, \tag{11}$$

where $Z_{I+I'}$ is a variable that, in the least solution, is assigned the set of integers $I + I'$ (cf. Example V.1). This completes the proof of the upper bound of Theorem IV.8.

In order to complete the proof of Theorem IV.5, we consider now the (orbit push-pop) rule restricted to only timed stack symbols. For convenience, we extend PUSH-POP^1 with the stack symbol and consider

$$\text{PUSH-POP}^2 = \left\{ (q, \bar{q}, \bar{q}', q', \bar{s}) \in Q^4 \times S_1 \mid \begin{array}{c} \text{PUSH}(q, \bar{q}, \bar{s}), \\ \text{POP}(\bar{q}', \bar{s}, q') \end{array} \right\}$$

Since we are considering orbit-finite trPDA, PUSH_{23} and POP_{12} are orbit-finite. Thus, PUSH-POP^2_{235} is orbit-finite as well (in passing we extend the notation for projection from pairs to to triples of coordinates), due to the restriction to timed stack symbols only. Indeed, the uniform bound on the span of PUSH-POP^2_{235} is at most twice as large as the universal bound on span of sets PUSH_{23} and POP_{12}. By Corollary III.3 we may enumerate all orbits $O \subseteq \text{PUSH-POP}^2_{235}$ in ExpTime.

Consider every orbit $O \subseteq \text{PUSH-POP}^2_{235}$ separately. We transform the set PUSH-POP^2 into normal form (using Lemma III.2) and apply Projection Lemma III.4 to deduce that the set

$$X_O = \left\{ (q, q') \in Q^2 \mid \exists (\bar{q}, \bar{q}', \bar{s}) \in O \cdot \begin{array}{c} \text{PUSH}(q, \bar{q}, \bar{s}), \\ \text{POP}(\bar{q}', \bar{s}, q') \end{array} \right\}$$

is definable and computable in ExpTime. For every $(\bar{o}, \bar{z})$ in the inverse image of $O_{12} \subseteq Q^2$ (we use Proposition V.9 here), and for every (o, I) in the decomposition of X_O (by Decomposition Lemma V.8), we add to Δ the inclusion

$$X_o \supseteq (X_{\bar{o}} \cap \{\bar{z}\}) + Z_{I - \bar{z}}, \tag{12}$$

where $I - \bar{z} = \{z - \bar{z} : z \in I\}$. This completes the construction of Δ. Since Δ is of exponential size, we can solve it NEXPTIME according to Lemma V.2. This concludes the proof of Theorem IV.5.

VI. Conclusions and future work

We have investigated the reinterpretation of the classical definition of pushdown automata in the setting of sets with timed atoms, called trPDA. In order to relate to the previous research we identified the subclass of trPDA with *timeless stack*, and shown that dense-timed PDA of [5] can be effectively transformed into this subclass.

The rest of the paper focused on the non-emptiness analysis of trPDA. We showed that the non-emptiness problem for unrestricted trPDA is undecidable, but decidable in NExpTime for orbit-finite trPDA. Furthermore, non-emptiness for an even smaller subclass of trPDA with timeless stack has been shown ExpTime-complete. The last result subsumes the ExpTime-completness of dtPDA [5], by our language-preserving transformation of dtPDA to trPDA with timeless stack.

As future research, it remains to be closed the complexity gap for orbit-finite trPDA, as well as the detailed study of expressive power of different subclasses of trPDA. Moreover, in this paper we did not consider all reasonable subclasses of trPDA. For instance, we do not know the decidability status of non-emptiness of *lhs orbit-finite* trPDA, defined like orbit-finite trPDA but with the orbit-finiteness restriction imposed on the left-hand sides of transition rules only. With respect to non-emptiness, the class is equivalent to the superclass of short form trPDA (cf. Sec. IV), obtained by dropping the orbit-finiteness restriction on the rhs of PUSH and on the lhs of POP. Our reduction, when extended to this model, yields systems of equations over sets of integers that use intersections with arbitrary intervals. Decidability of such extended systems of equations is, up to our knowledge, an open problem, interesting on its own.

Finally, first-order definable sets may be considered for other atoms. We have recently studied the reachability analysis for PDA for the important class of *oligomorphic* atoms (i.e., $\mathbb{A}^n$ is orbit-finite for every n) in [13], where most of the subclasses of PDA defined in this paper become expressively equivalent. This covers many examples, such as total order atoms $(\mathbb{Q}, \leq)$, partial order atoms, tree order atoms, and many more [14].

References

[1] R. Alur and D. L. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, pp. 183–235, April 1994.

[2] A. Bouajjani, R. Echahed, and R. Robbana, "On the automatic verification of systems with continuous variables and unbounded discrete data structures," in *Hybrid Systems'94*, 1994, pp. 64–85.

[3] A. Trivedi and D. Wojtczak, "Recursive timed automata," in *Proc. of ATVA'10*. Springer, 2010, pp. 306–324.

[4] M. Benerecetti, S. Minopoli, and A. Peron, "Analysis of timed recursive state machines," in *Proc. of TIME'10*, sept. 2010, pp. 61–68.

[5] P. A. Abdulla, M. F. Atig, and J. Stenman, "Dense-timed pushdown automata," in *Proc. of LICS'12*, june 2012, pp. 35–44.

[6] M. Kaminski and N. Francez, "Finite-memory automata," *Theor. Comput. Sci.*, vol. 134, no. 2, pp. 329–363, 1994.

[7] M. Bojańczyk, B. Klin, and S. Lasota, "Automata theory in nominal sets," *Logical Methods in Computer Science*, vol. 10, no. 3:4, 2013.

[8] E. Y. C. Cheng and M. Kaminski, "Context-free languages over infinite alphabets," *Acta Inf.*, vol. 35, no. 3, pp. 245–267, 1998.

[9] M. Bojanczyk and S. Lasota, "A machine-independent characterization of timed languages," in *In Proc. of ICALP'12*. Springer, 2012, pp. 92–103.

[10] A. Jeż and A. Okhotin, "Complexity of equations over sets of natural numbers," *Theory Comput. Syst.*, vol. 48, no. 2, pp. 319–342, 2011.

[11] B. Bérard, A. Petit, V. Diekert, and P. Gastin, "Characterization of the expressive power of silent transitions in timed automata," *Fundam. Inf.*, vol. 36, no. 2-3, pp. 145–182, Nov. 1998.

[12] A. Jeż and A. Okhotin, "Conjunctive grammars over a unary alphabet: Undecidability and unbounded growth," *Theory Comput. Syst.*, vol. 46, no. 1, pp. 27–58, 2010.

[13] L. Clemente and S. Lasota, "Reachability analysis of first-order definable pushdown systems," University of Warsaw, Tech. Rep., 2015, submitted to CSL'15. [Online]. Available: http://arxiv.org/abs/1504.02651

[14] D. Macpherson, "A survey of homogeneous structures," *Discrete Mathematics*, vol. 311, no. 15, p. 15991634, 2011.

[15] J. Hopcroft, R. Motwani, and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2000.

[16] L. J. Stockmeyer and A. R. Meyer, "Word problems requiring exponential time (preliminary report)," in *Proc. of STOC'73*, ser. STOC '73. New York, NY, USA: ACM, 1973, pp. 1–9.

[17] E. Kopczyński and A. W. To, "Parikh images of grammars: Complexity and applications," in *Proc. of LICS'10*, July 2010, pp. 80–89.

APPENDIX A
PROOF OF THEOREM II.1

A. Preliminaries

A *configuration* c of a dtPDA as above is a tuple (p, μ, r), where $p \in L$ is a control location, $\mu : X \to \mathbb{Q}^{\geq 0}$ is a *clock valuation* for the clocks in X, and $r \in (\Gamma \times \mathbb{Q}^{\geq 0})^*$ is the *stack valuation*, recording stack symbols and their current age. For a clock valuation $\mu : X \to \mathbb{Q}^{\geq 0}$ and a number $k \in \mathbb{Q}^{\geq 0}$, we denote by $\mu + k$ the valuation which adds k to every clock, i.e., for every $x \in X$, $(\mu + k)(x) = \mu(x) + k$. Similarly, for a stack valuation $r = (\gamma_1, h_1) \dots (\gamma_n, h_n)$ and $k \in \mathbb{Q}^{\geq 0}$, we write $r + k$ for the new valuation $(\gamma_1, h_1 + k) \dots (\gamma_n, h_n + k)$. Given a clock valuation $\mu : X \to \mathbb{Q}^{\geq 0}$, a set of clocks $Y \subseteq X$, and a new value $k \in \mathbb{Q}^{\geq 0}$, we denote with $\mu[Y \leftarrow k]$ the new clock valuation which is the same as μ except that assigns value k to all clocks in Y, i.e., $\mu[Y \leftarrow k](y) = k$ if $y \in Y$ and $\mu[Y \leftarrow k](y) = \mu(y)$ if $y \notin Y$. Similarly, we denote with $\mu[z \leftarrow k] : (X \cup \{z\}) \to \mathbb{Q}^{\geq 0}$ the extended clock valuation where the value of the special clock z is k. Given an extended clock valuation $\mu' : (X \cup \{z\}) \to \mathbb{Q}^{\geq 0}$ and a formula φ, we write $\mu' \models \varphi$ if φ holds when clock variables are replaced by values given by μ'. As usual, we distinguish transitions representing elapse of time, which are labelled by some $t \in \mathbb{Q}^{\geq 0}$, and discrete transitions, which for convenience are labelled by tuples of the form (a, φ, Y, op). Formally, for every $t \in \mathbb{Q}^{\geq 0}$ we have a timed transition $(p, \mu, r) \xrightarrow{t} (p, \mu + t, r + t)$, and, if $p \xrightarrow{a,\varphi,Y,op} q$, then we have a discrete transition $(p, \mu, r) \xrightarrow{a,\varphi,Y,op} (q, \mu', r')$, whenever $\mu \models \varphi$ holds, $\mu' = \mu[Y \leftarrow 0]$, and, depending on the kind of operation op,

- Case $op = \mathsf{nop}$: $r' = r$.
- Case $op = \mathsf{push}(\alpha \models \psi)$: $r' = r(\alpha, k)$ if $\mu[z \leftarrow k] \models \psi$.
- Case $op = \mathsf{pop}(\alpha \models \psi)$: $r = r'(\alpha, k)$ if $\mu[z \leftarrow k] \models \psi$.

A run π is an alternating sequence $c_0 tr_0 \dots c_k$ of configurations c_i's and transitions tr_i's s.t., for every $0 \leq i < k$, $c_i \xrightarrow{tr_i} c_{i+1}$.

B. Simplifications

The following simplifying assumptions can be shown not to reduce the recognizing power of the model. They are either standard, or very easy to show.

a) Only pop constraints of the form $z - x \sim k$: Constraints of the form $x - z \sim k$ can be converted to the form $z - x \sim k$ by negating both sides and by flipping the inequality.

b) No pop constraints of the form $z \sim k$: We introduce a new clock x_0 which is ensured to be 0 when a pop transition is taken. A constraint $z \sim k$ can thus be replaced by $z - x_0 \sim k$. Formally, a pop transition $p \xrightarrow{a,\varphi,T,\mathsf{pop}(\alpha \models \psi)} q$ is simulated in two steps:

$$p \xrightarrow{a,\varphi,\{x_0\},\mathsf{nop}} (p, \alpha, \psi', q) \qquad (1)$$

$$(p, \alpha, \psi', q) \xrightarrow{\varepsilon,\varphi \wedge x_0 = 0,T,\mathsf{pop}(\alpha \models \psi')} q \qquad (2)$$

where ψ' is the same as ψ where all constraints of the form $z \sim k$ are replaced by $z - x_0 \sim k$. Transition (1) mimics exactly the original transition, except that the pop operation is not performed. Transition (2) is ensured to be taken with delay 0 after (1), and the pop operation is thus performed under the condition that x_0 is 0.

c) No resets on push and pop transitions: We wish that clocks are reset only on nop transitions. To achieve this, we introduce an extra clock x^* and some auxiliary intermediate control states. A push/pop transition of the form $p \xrightarrow{a,\varphi,T,op} q$ is split into three consecutive transitions

$$p \xrightarrow{a,\varphi,\{x^*\},\mathsf{nop}} (p, a, \varphi, T, op, 0) \qquad (1)$$

$$(p, a, \varphi, T, op, 0) \xrightarrow{\varepsilon,x^*=0,\emptyset,op} (p, a, \varphi, T, op, 1) \qquad (2)$$

$$(p, a, \varphi, T, op, 1) \xrightarrow{\varepsilon,x^*=0,T,\mathsf{nop}} q \qquad (3)$$

Transition (1) reads the input, checks the clock constraint φ, but it does not reset clocks T, neither perform the stack operation. Transition (2) performs the actual stack operation (without resetting any clock), and the constraint on x^* ensures that no time elapsed since (1). Finally, transition (3) resets the clocks in T, and again no time is allowed to elapse. Clocks in T cannot be reset directly by transition (1) since in the semantics of push/pop operations we must compare the age of the stack symbol to the value of clocks prior to their reset.

d) The initial age is 0: A push operation $\mathsf{push}(\alpha \models \psi_1)$ can be restricted to have the push constraint ψ_1 of the trivial form $z = 0$, i.e., the initial age is always 0. We omit a trivial push constraint by just writing $\mathsf{push}(\alpha)$.

A conjunctive constraint ψ_1 involving only clock z is equivalent to either a punctual constraint $z = k$ for an integer k, or to an interval constraint $z \in (a, b)$ for a lower bound $a \in \mathbb{Z} \cup \{-\infty\}$ and an upper bound $b \in \mathbb{Z} \cup \{+\infty\}$. The idea is to push this information on the stack, which is used at the time of pop to update the pop constraint.

The function $update(\psi, I)$ for I equal to either k or (a, b) as above, is defined by structural induction on ψ and it works by shifting all constraints by the amount specified by I:

$$update(z - x \sim h, k) = z - x \sim h - k \text{ for } \sim \in \{\leq, <, >, \geq\}$$
$$update(z - x \lesssim h, (a, b)) = z - x < h - a \text{ for } \lesssim \in \{\leq, <\}$$
$$update(z - x \gtrsim h, (a, b)) = z - x > h - b \text{ for } \gtrsim \in \{\geq, >\}$$
$$update(\mathbf{t}, I) = \mathbf{t}$$
$$update(\psi_0 \wedge \psi_1, I) = update(\psi_0) \wedge update(\psi_1)$$

A push transition $p \xrightarrow{a,\varphi,Y,\mathsf{push}(\alpha \models \psi)} q$ becomes $p \xrightarrow{a,\varphi,Y,\mathsf{push}((\alpha, I))} q$ where I is either k or (a, b) as implied by the constraint ψ, and a pop transition $p \xrightarrow{a,\varphi,Y,\mathsf{pop}(\alpha \models \psi)} q$ is replaced by several transitions of the form $p \xrightarrow{a,\varphi,Y,\mathsf{pop}((\alpha, I) \models \psi')} q$, for every shift I, and where $\psi' = update(\psi, I)$ is the pop constraint updated by I. Correctness for a punctual constraint is immediate. For $z \in (a, b)$ and a pop constraint $z - x \lesssim h$, the semantics asks whether there exists an initial age $z_0 \in (a, b)$ s.t. $z - x \lesssim h$

holds at the time of pop, which is the same as requiring $(z + z_0) - x \lesssim h$ when the initial age is 0 instead, that is, $z - x \lesssim h - z_0$. This constraint is easier to satisfy for smaller values of $z_0 > a$, and thus we obtain $z - x < h - a$. The reasoning for $z - x \gtrsim h$ is analogous.

C. Untiming the stack

We are now ready present the formal construction for untiming the stack of a dtPDA. Let $\mathcal{T} = (Q, q_0, \Sigma, \Gamma, X, z, \Delta)$ be a dtPDA. We construct a dtPDA with timeless stack $\mathcal{U} = (Q', q_0', \Sigma, \Gamma', X', \Delta')$. Let

$$\mathcal{C}_\sim = \{(x, \sim, k) \mid z - x \sim k \in \psi \text{ for a pop constraint } \psi\}$$

The new set of clocks X' is obtained by adding to X a clock $\hat{x}_{\sim k}$ for every $(x, \sim, k) \in \mathcal{C}$, where $\mathcal{C} = \bigcup_\sim \mathcal{C}_\sim$. A control state in $\mathcal{U}$ is of the form (p, R, O), where p is a control state in $\mathcal{T}$. The set $R \subseteq \mathcal{C}_< \cup \mathcal{C}_\leq$ represents active reset restrictions, and the set $O \subseteq \mathcal{C}_> \cup \mathcal{C}_\geq$ represents active reset obligations. The initial control state of $\mathcal{U}$ is $q_0' = (q_0, \emptyset, \emptyset)$. The stack alphabet of $\mathcal{U}$ consists of tuples (α, ψ, R, O), where $\alpha \in \Gamma$ is a stack symbol of $\mathcal{T}$, ψ is a clock constraint, and R, O are as above.

Transitions in $\mathcal{U}$ are defined as follows. If we have a push transition $p \xrightarrow{a,\varphi,\emptyset,\mathsf{push}(\alpha)} q$ in $\mathcal{T}$, then we have several push transitions in $\mathcal{U}$ of the form

$$(p, R, O) \xrightarrow{a,\varphi',T',\mathsf{push}(\alpha')} (q, R', O_1') \quad \text{with } \alpha' = (\alpha, \psi, R, O_0')$$

for every $R, R' \subseteq \mathcal{C}_< \cup \mathcal{C}_\leq$, $O, O_0', O_1' \subseteq \mathcal{C}_> \cup \mathcal{C}_\geq$, and constraints ψ, φ' satisfying the conditions below. Constraint ψ is guessed to be the constraint that will hold at the time of the corresponding pop. The other components are determined as follows. We first consider reset restrictions. Let $M = \{(x, \lesssim, k) \mid z - x \lesssim k \in \psi\}$ be the new reset restrictions as implied by the guessed pop constraint ψ. Since restrictions in R subsume those in M, we reset $\hat{x}_{\lesssim k}$ only for restrictions in $M - R$.

$$\begin{aligned} R' &= R \cup M \\ T_0 &= \{\hat{x}_{\lesssim k} \mid (x, \lesssim, k) \in M - R\} \\ \varphi_0 &= \varphi \wedge \bigwedge_{(x,\lesssim,k) \in M - R} -x \lesssim k \end{aligned}$$

We now address reset obligations. Let $N = \{(x, \gtrsim, k) \mid z - x \gtrsim k \in \psi\}$ be the new reset obligations. New reset obligations in N always subsume previous ones in O. Let $O_0' \subseteq O - N$ be any set of previous reset obligations not subsumed by new ones. Intuitively, we guess that obligations in O_0' will be satisfied after the matching pop, thus we push them on the stack. Let $N' \subseteq N$ be those new reset obligations which are already satisfied by a previous reset, and thus need not be added.

$$\begin{aligned} O_1' &= (O - O_0') \cup (N - N') \\ T' &= T_0 \cup \{\hat{x}_{\gtrsim k} \mid (x, \gtrsim, k) \in N - N'\} \\ \varphi' &= \varphi_0 \wedge \bigwedge_{(x,\gtrsim,k) \in N'} -x \gtrsim k \end{aligned}$$

For every pop transition $p \xrightarrow{a,\varphi,\emptyset,\mathsf{pop}(\alpha \models \psi)} q$ in $\mathcal{T}$, we have an "untimed" pop transition in $\mathcal{U}$ of the form

$$(p, R, \emptyset) \xrightarrow{a,\varphi,\emptyset,\mathsf{pop}(\alpha')} (q, R', O') \quad \text{with } \alpha' = (\alpha, \psi, R', O')$$

for every $R, R' \subseteq \mathcal{C}_< \cup \mathcal{C}_\leq$ and $O' \subseteq \mathcal{C}_> \cup \mathcal{C}_\geq$. We require the set of reset obligations to be empty in order to ensure that all clocks that were under a reset obligation have been indeed reset.

For every nop transition $p \xrightarrow{a,\varphi,T,\mathsf{nop}} q$ in $\mathcal{T}$, we have a nop transition in $\mathcal{U}$ of the form

$$(p, R, O) \xrightarrow{a,\varphi',T,\mathsf{nop}} (q, R, O - O')$$

for every $R \subseteq \mathcal{C}_< \cup \mathcal{C}_\leq$, $O, O' \subseteq \mathcal{C}_> \cup \mathcal{C}_\geq$, and for every set of reset obligations $O' \subseteq \{(x, \gtrsim, k) \in O \mid x \in T\}$ which are satisfied by a reset in this transition:

$$\varphi' = \varphi \wedge \bigwedge_{x \in T, (x,\lesssim,k) \in R} (\hat{x}_{\lesssim k} \lesssim k) \wedge \bigwedge_{(x,\gtrsim,k) \in O'} (\hat{x}_{\gtrsim k} \gtrsim k)$$

Theorem II.1. *A dtPDA $\mathcal{T}$ can be effectively transformed into a dtPDA $\mathcal{U}$ with timeless stack recognizing the same timed language. Moreover, $\mathcal{U}$ has linearly many clocks w.r.t. $\mathcal{T}$, and exponentially many control locations.*

Proof. Let π be an accepting run in $\mathcal{T}$,

$$\pi = (p_0, \nu_0, v_0) tr_0 (p_1, \nu_1, v_1) \cdots (p_{k+1}, \nu_{k+1}, v_{k+1}) \quad \text{with } tr_i = (a_i, \varphi_i, T_i, op_i)$$

We construct an accepting run π' in $\mathcal{U}$,

$$\pi' = (r_0, \mu_0, u_0) tr_0' (r_1, \mu_1, u_1) \cdots (r_{k+1}, \mu_{k+1}, u_{k+1}), \quad \text{with } r_i = (p_i, R_i, O_i) \text{ and } tr_i' = (a_i, \varphi_i', T_i', op_i')$$

where $tr_i' = tr_i$ if $tr_i \in \mathbb{R}$, and otherwise it is determined as follows. For every $i \leq j$, let $t_{i,j}$ be the total time elapsed from transition i to transition j, i.e.,

$$t_{i,j} = \sum_{h=i}^{j} \begin{cases} tr_h & \text{if } tr_h \in \mathbb{R} \\ 0 & \text{otherwise} \end{cases}$$

If $i > j$, we define $t_{i,j} = -t_{j,i}$. The construction of π' is based on the following two observations.

1) For any reset restriction $z - x \lesssim k \in \psi_{j_i}$, whenever $x \in T_h$ is reset at transition tr_h with $h < j_i$, the time elapsed between tr_i and tr_h is $t_{i,h} \lesssim k$.
2) For any reset obligation $z - x \gtrsim k \in \psi_{j_i}$, there exists a *minimal* index $h < j_i$ s.t. $x \in T_h$ is reset at transition tr_h and $t_{i,h} \gtrsim k$. (Minimality is important to construct a run

in $\mathcal{U}$, in order to mimic the fact that new reset obligations truly subsume old ones.)

We proceed by a case analysis on op_i. Let $op_i = \mathsf{push}(\alpha_i)$. The corresponding pop operation has $op_{j_i} = \mathsf{pop}(\alpha_{j_i} \models \psi_{j_i})$, with $\alpha_{j_i} = \alpha_i$. By assumption, $T_i = \emptyset$. Take $tr'_i = (a_i, \varphi'_i, T'_i, op'_i)$ with $op'_i = (\alpha_i, \psi_{j_i}, R_i, O^0)$, where φ'_i, T'_i, and O^0 are defined as follows. We first analyse reset restrictions. Let $M = \{(x, \lesssim, k) \mid z - x \lesssim k \in \psi_{j_i}\}$ be the set of reset restrictions due to ψ_{j_i}, and let

$$T^0 = \left\{\hat{x}_{\lesssim k} \mid (x, \lesssim, k) \in M - R_i\right\}$$
$$\varphi^0 = \varphi_i \wedge \bigwedge_{(x,\lesssim,k)\in M - R_i} -x \lesssim k$$

We show $\mu_i \models \varphi^0$. First, $\mu_i \models \varphi_i$ holds because π is a valid run in $\mathcal{T}$. Let $(x, \lesssim, k) \in M - R_i$. Then, $\mu_{j_i} \models z - x \lesssim k$. If $k \geq 0$, then $\mu_i \models -x \lesssim k$ immediately holds. If $k < 0$, let h be the last transition before tr_i when x is reset. By Point 1) above, $t_{i,h} \lesssim k$, i.e., the last reset of x is more than $-k$ time units before transition i. Thus, $\mu_i \models x \gtrsim -k$.

We now analyse reset obligations. Let $N = \{(x, \gtrsim, k) \mid z - x \gtrsim k \in \psi_{j_i}\}$ be the reset obligations due to ψ_{j_i}, and let $N' = \{(x, \gtrsim, k) \in N \mid \mu_i \models -x \gtrsim k\}$ be those obligations in N which are already satisfied by a past reset. (Necessarily $k \leq 0$ for $(x, \gtrsim, k) \in N'$.) For $(x, \gtrsim, k) \in O_i$, let l be the largest index $< i$ s.t. $(x, \gtrsim, k) \in O_{l+1} - O_l$. Then, tr_l is a push transition with matching pop transition tr_{j_l} with $(x, \gtrsim, k) \in \psi_{j_l}$. By Point 2) above, there exists $h < j_l$ s.t. $x \in T_h$ and $t_{l,h} \gtrsim k$. Let $O^0 = \{(x, \gtrsim, k) \in O \mid h > j_i\}$ be those obligations which will be satisfied after the matching pop transition tr_{j_i}. Obligations in O^0 are pushed on the stack. Then, let

$$O_{i+1} = (O_i - O^0) \cup (N - N')$$
$$T'_i = T^0 \cup \left\{\hat{x}_{\gtrsim k} \mid (x, \gtrsim, k) \in N - N'\right\}$$
$$\varphi'_i = \varphi_0 \wedge \bigwedge_{(x,\gtrsim,k)\in N'} -x \gtrsim k$$

Clearly, $\mu_i \models \varphi'_i$ holds, since we proved above $\mu_i \models \varphi_0$, and by the definition of N'.

Let's now analyse the corresponding pop operation $op_{j_i} = \mathsf{pop}(\alpha_{j_i} \models \psi_{j_i})$. Once again, $T_{j_i} = \emptyset$ by assumption. By construction of π' (cf. the push transition above), tr'_i pushed a symbol of the form $(\alpha_i, \psi_i, R_i, O^0)$, with $\alpha_i = \alpha_{j_i}$ and $\psi_i = \psi_{j_i}$. Therefore, take $tr'_{j_i} = (a_{j_i}, \varphi_{j_i}, \emptyset, op'_{j_i})$ with $op'_{j_i} = \mathsf{pop}((\alpha_i, \psi_i, R_i, O^0))$ and define $R_{j_i+1} := R_i$ and $O_{j_i+1} := O^0$. By construction, reset obligations added to O_{j_i} are removed as soon as they can be satisfied (cf. the definition of O' in the nop rule below). All reset obligations can be satisfied by Point 2) above. Thus, $O_{j_i} = \emptyset$, and tr'_{j_i} is a valid transition.

Finally, Let $op_i = \mathsf{nop}$. Let O' be defined as

$$O' = \left\{(x, \gtrsim, k) \in O_i \mid x \in T_i \text{ and } \mu_i \models \hat{x}_{\gtrsim k} \gtrsim k\right\}$$

Take $tr'_i = (a_i, \varphi'_i, T'_i, \mathsf{nop})$, with $T'_i = T_i$ and

$$\varphi'_i = \varphi_i \wedge \varphi^0 \wedge \varphi^1, \text{ where}$$
$$\varphi^0 = \bigwedge_{x\in T_i,(x,\lesssim,k)\in R_i} \hat{x}_{\lesssim k} \lesssim k$$
$$\varphi^1 = \bigwedge_{(x,\gtrsim,k)\in O'} \hat{x}_{\gtrsim k} \gtrsim k$$

and let $R_{i+1} = R_i$ and $O_{i+1} = O_i - O'$. We show that $\mu_i \models \varphi'_i$.

- $\mu_i \models \varphi_i$ since π is a valid run in $\mathcal{T}$.
- $\mu_i \models \varphi^0$: Let $x \in T_i$ and $(x, \lesssim, k) \in R_i$. We show $\mu_i \models \hat{x}_{\lesssim k} \lesssim k$. Let h^* be the largest index $h < i$ s.t. $(x, \lesssim, k) \in R_{h+1} - R_h$. Then, tr_{h^*} is a push transition, and $\hat{x}_{\lesssim k} \in T_{h^*}$ is reset at transition tr_{h^*}. Moreover, since h^* is maximal and $(x, \lesssim, k) \in R_i$, by construction $(x, \lesssim, k) \in R_h$ for every $h^* \leq h \leq i$. Thus, $\hat{x}_{\lesssim k} \notin T_h$ for every $h^* < h \leq i$. Therefore, $\mu_i(\hat{x}_{\lesssim k}) = t_{h^*,i}$. Since at the matching pop transition $tr_{j_{h^*}}$ we have $z - x \lesssim k \in \psi_{j_{h^*}}$, and $x \in T_i$ is reset now, by Point 1) above we have $t_{h^*,i} \lesssim k$. Consequently, $\mu_i \models \hat{x}_{\lesssim k} \lesssim k$.
- $\mu_i \models \varphi^1$: Immediately by the choice of O'.

Thus, tr'_i is a valid transition.

For the other inclusion, let $w = (a_0, t_0) \cdots (a_k, t_k)$ be a timed word accepted by $\mathcal{U}$, and let π' be an accepting run:

$$\pi' = (r_0, \mu_0, u_0) tr'_0 (r_1, \mu_1, u_1) \cdots (r_{k+1}, \mu_{k+1}, u_{k+1}),$$
$$\text{with } r_i = (p_i, R_i, O_i)$$

We obtain an accepting run π in $\mathcal{T}$ by removing the extra components in the control state and stack alphabet, and by adding back pop constraints (as given by the symbol popped). To show that π is an accepting run, we argue that $\mu_i \models \psi_i$ holds for a pop transition

$$tr'_i = (a_i, \varphi_i, T_i, \mathsf{pop}(\alpha'_i))$$
$$\text{with } \alpha'_i = (\alpha_i, \psi_i, R'_i, O'_i)$$

Let tr'_j with $j < i$ be the corresponding push transition, i.e.,

$$tr'_j = (a_j, \varphi_j, T_j, \mathsf{push}(\alpha'_j)) \quad (13)$$
$$\text{with } \alpha'_j = \alpha'_i$$

Notice that the symbol popped at time i matches the one pushed at time j. We begin with reset restrictions. Let $z - x \lesssim k \in \psi_i$ any reset restriction on clock x with $\lesssim \in \{\leq, <\}$. We argue that $\mu_i \models z - x \lesssim k$ holds. The claim follows from the following observations. Let $j < h \leq i$ in the following:

1) Except possibly for the first push transition j, clock $\hat{x}_{\lesssim k}$ is never reset before, and including, transition i. This is follows from the fact that, once a reset obligation is added, it always subsumes new ones. Thus, $\mu_h(\hat{x}_{\lesssim k})$ is at least the age of α'_i at index h.
2) $(x, \lesssim, k) \in R_h$. Indeed, $(x, \lesssim, k) \in R_{j+1}$ by construction. Moreover, nop operations do not change $R_h = R_{h+1}$, push operations put R_h on the stack and $R_h \subseteq R_{h+1}$, and pop operations restore the R_h of the corresponding push.

3) By the previous point, if x is reset (necessarily at a nop operation by assumption), then $\mu_h \models \hat{x}_{\lesssim k} \lesssim k$ holds by the definition of nop operation.
4) Finally, $\mu_h \models -x \lesssim k$, for $h = j$. If $k \geq 0$, this is trivial. Otherwise, let $k < 0$, and let j^* be the largest index $j^* \leq j$ s.t. $(x, \lesssim, k) \in R_{j^*} - R_{j^*-1}$ is last added to reset obligations. Then, $\mu_{j^*} \models -x \lesssim k$ holds by definition of push operation. Since $k < 0$, x is not reset ever since (cf. Point 3), and thus $\mu_j \models -x \lesssim k$.

There are two cases. If x is reset between transition j and i, then by 1) and 3) the age of α'_i is $\lesssim k$ the last time x was reset. Consequently, at transition i, $\mu_i \models z - x \lesssim k$. If x is not reset at all, then $\mu_j \models -x \lesssim k$ (by Point 4) immediately implies $\mu_i \models z - x \lesssim k$.

We now consider reset obligations. Let $z - x \gtrsim k \in \psi_i$ with $\gtrsim \in \{\geq, >\}$. We argue that $\mu_i \models z - x \gtrsim k$ holds. There are two cases. If $(x, \gtrsim, k) \notin O_{j+1}$, then $\mu_j \models -x \gtrsim k$ holds by construction, i.e., the constraint must have been satisfied by a previous reset of x, which directly implies $\mu_i \models z - x \gtrsim k$. Now let $(x, \gtrsim, k) \in O_{j+1}$. We make the following observation.

5) $\hat{x}_{\gtrsim k}$ is at most the age of α'_i. This is obvious, since $\hat{x}_{\gtrsim k}$ is reset at transition j by construction.

Since the pop at transition i satisfies $O_i = \emptyset$, constraint $(x, \gtrsim, k)$ must be eventually removed. The only way to remove $(x, \gtrsim, k)$ from O_h is to either push it on the stack (cf. O'_0), or to reset x when $\mu_h \models \hat{x}_{\gtrsim k} \gtrsim k$ holds (by definition of nop operation). In the former case, $(x, \gtrsim, k)$ will reappear in O_h at the matching pop operation and still be pending. In the latter case, the age of α'_i was at least k when x was reset by Point 5) above, which directly implies $\mu_i \models z - x \gtrsim k$. □

APPENDIX B
PROOFS MISSING IN SEC. III

Lemma III.1. *An equivariant subset $X \subseteq \mathbb{Q}^n$ is orbit-finite if, and only if, it has uniformly bounded span, i.e., it admits a common bound on the spans of all its elements.*

Proof. Every orbit, being defined by a minimal constraint, has uniformly bounded span. Therefore every orbit-finite set also does.

For the opposite direction, if the span of the elements of an equivariant set $X \subseteq \mathbb{Q}^n$ is bounded by k, then X is a subset of

$$\{(x_1, \ldots, x_n) \ : \ \bigwedge_{i \neq j} x_i - x_j \leq k\}.$$

The latter set can be equivalently defined by a finite disjunction of minimal constraints, and hence it is orbit-finite, which implies orbit-finiteness of X. □

Lemma III.2 (Normal Form Lemma)**.** *Every definable set X decomposes into a finite union of orbits $O \subseteq X$ and of extensions of orbits $O \subseteq X$. A decomposition can be effectively computed in* `ExpTime`.

Proof. Fix a definable set X. Let L be its indexing set. For each index $l \in L$ separately, we compute a decomposition of X_l.

Fix the index l. Let K be larger than the largest absolute value of any integer constant used in the defining constraint of X_l, and let d be the dimension of X_l. Enumerate all minimal constraints φ that define an orbit of span bounded by $(d-1)\cdot K$. Note that such constraints do not need to use integer constants of absolute value greater than $(d-1)\cdot K$.

Claim B.0.1. *It is decidable in polynomial time whether $[\varphi] \subseteq X_l$.*

Proof. Indeed, for every pair of variables x, y the minimal constraint φ determines an interval $I_{x,y}$ of the form

$$\{z\} \qquad \text{or} \qquad (z, z+1),$$

for $z \in \mathbb{Z}$, of possible values of $x - y$. In order to determine whether $[\varphi] \subseteq X_l$, we evaluate the constraint ψ defining X_l over the minimal constraint φ, very much like a boolean formula is evaluated over a valuation of its variables. Atomic sub-formulae of ψ are evaluated on the basis of the intervals $I_{x,y}$; for instance

$$x - y < z$$

evaluates to true if and only if $I_{xy} \subseteq \mathbb{Z}_{<z}$. $[\varphi] \subseteq X_l$ iff the constraint ψ evaluates to true over φ. □

Thanks to the claim, we compute all minimal constraints satisfying $[\varphi] \subseteq X$ and add them to the decomposition of X.

The next claim formulates the weakness of constraints that we build upon:

Claim B.0.2. *let $O \subseteq \mathbb{Q}^d$ be an orbit. If $O \subseteq X$ and O admits a gap K then the K-extension of O is included in X.*

Therefore, for every minimal constraint φ satisfying the above claim, we add to the decomposition of X its K-extension. The decomposition is computed in `ExpTime`, and its correctness follows by following last claim:

Claim B.0.3. *Every orbit $O \subseteq X$ of span larger than $(d-1)\cdot K$ is included in the K-extension of some orbit $O' \subseteq X$ of span at most $(d-1)\cdot K$.*

This completes the proof of Lemma III.2. □

APPENDIX C
PROOFS MISSING IN SEC. IV

Theorem IV.1. *Non-emptiness of trPDA is undecidable.*

Proof. The proof works for transition rules satisfying $n = 1$, $m = 2$ in (4). The idea is to simulate a 2-counter Minsky machine M by a trPDA $\mathcal{A}_M$: one counter is simulated using the stack and two stack symbols $\bot, \top$, while the other counter is simulated using the difference between the time values stored in the top-most stack symbol and in the state. It is enough if state space and stack alphabet are 1-dimensional, i.e., store exactly one time value. A configuration of M with

a control state p and values of counters n_1, n_2 is represented by the following configuration of $\mathcal{A}_M$:

$$(\ (p, t+n_1+n_2),\quad (\top, t+n_1)\dots(\top, t+1)(\bot, t)\ )$$

for an arbitrary $t \in \mathbb{Q}$ chosen nondeterministically by $\mathcal{A}_M$ in the beginning of the simulation. The simulation assumes that the time values stored in consecutive stack symbols increase by 1, thus a push operation needs to see the current top-most stack symbol. Then increment (resp. decrement) of the first counter is simulated by a simultaneous push (resp. pop), and increment (resp. decrement) of the state by 1, e.g.: if M increments the first counter and changes state from p to p', the PDA has the following transition rule ($\texttt{inc}_1$ is an input letter):

$$((p,t),(\top,u),\texttt{inc}_1,(p',t+1),(\top,u+1)(\top,u)).$$

Operations on the second counter are performed exclusively on the time value stored in the state. Zero test $n_1 = 0$ of the first counter is done by checking if the top-most stack symbol is $(\bot, t)$ for an arbitrary $t \in \mathbb{Q}$; while zero test $n_2 = 0$ of the second counter is done by an equality test $t = u$ of the time values stored in the state and in the top-most symbol.

$\mathcal{A}_M$ accepts if M halts from the control state p. Thus the language $L(\mathcal{A}_M)$ is non-empty iff M halts. □

Lemma IV.2. *The untiming of a timed register context-free language is effectively context-free.*

Proof. Let $\mathcal{G}$ be a trCFG with transition relation ρ, recognizing a timed language L. We show the untiming of L can be recognized by a CFG $\mathcal{G}'$ of size exponential in $\mathcal{G}$. Enumerate all orbits O of S; this can be done effectively by Corollary III.3. $\mathcal{G}'$ will have a non-terminal X_O for every orbit O of S. For every non-terminals $X_O, X_{O_1}, \dots, X_{O_n}$ and for every orbit P of A_ε, a production

$$(X_O, P, X_{O_1}, \cdots, X_{O_n})$$

is included in $\mathcal{G}'$ whenever $\exists x \in O, a \in P, x_1 \in O_1, \dots, x_n \in O_n \cdot \rho(x, a, x_1, \dots, x_n)$ holds. The latter condition can be checked in EXPTIME, similarly like in the proof of Lemma III.2. Then $\mathcal{G}$ recognizes a timed word if, and only if, $\mathcal{G}'$ recognizes its untiming. □

Theorem IV.3. *Non-emptiness problem of trCFG is* ExpTime*-complete.*

Proof. The EXPTIME upper-bound follows immediately from Lemma IV.2: From a trCFG $\mathcal{G}$ recognizing a timed language L, we derive an exponentially larger context-free grammar $\mathcal{G}'$ recognizing the untiming of L, for which non-emptiness is decidable in PTIME. Correctness follows since L is non-empty if, and only if, its untiming is non-empty.

For the lower-bound, we reduce from the non-emptiness problem of the intersection of the languages recognized by n (untimed) NFAs $\mathcal{A}_1, \dots, \mathcal{A}_n$ and a (untimed) CFG $\mathcal{G}$. (A similar reduction from this same problem was used in [5] to show EXPTIME-hardness of dtPDA). It is folklore that the latter problem is EXPTIME-hard; this can be shown by a direct reduction from linearly bounded alternating Turing machines.

We adapt the textbook construction for intersection of a regular language and a context-free one [15] in order to define a timed register context-free grammar $\mathcal{G}'$. We use timed registers to succinctly represent control states of the NFAs $\mathcal{A}_i$'s. Let P_i be the set of control states of $\mathcal{A}_i$. For simplicity, we assume that $P_i = \{1, \dots, k\}$, that 1 is the unique initial state of each NFA, and that 2 is the unique final state of each NFA. A tuple of states of NFAs may be encoded as $r \in \{1, \dots, k\}^n$. We write $r \xrightarrow{a} r'$ if for every i, the pair of states at coordinate i in r and r', is related in the automaton $\mathcal{A}_i$ by an a-labelled transition. We will represent a pair of such tuples $(r, r') \in \{1, \dots, k\}^{2n}$ as an orbit in $\mathbb{Q}^{2n+1}$; one additional component will serve as reference point, and the others will be interpreted as the difference w.r.t. the reference point. Thus, we encode (r, r') as the following orbit $O_{r,r'}$ in $\mathbb{Q}^{2n+1}$:

$$O_{r,r'} \;=\; \bigcup_{t \in \mathbb{Q}} (t, r+t, r'+t).$$

Let symbols S' of $\mathcal{G}'$ be

$$S' \;=\; S \times O_{r,r'}$$

for S the symbols of $\mathcal{G}$. Thus symbols in S' are of the form

$$(X, t, r+t, r'+t).$$

Notice that S' is orbit-finite. From the initial symbols, $\mathcal{G}'$ goes to one of the symbols

$$(X_0, t, \vec{1}+t, \vec{2}+t), \qquad \text{for } t \in \mathbb{Q},$$

where X_0 is the initial symbol of $\mathcal{G}$, and $\vec{1}$, $\vec{2}$ are constant tuples. Assume for simplicity that $\mathcal{G}$ is in Chomsky normal form. For every production $X \longrightarrow a$ in $\mathcal{G}$, the grammar $\mathcal{G}'$ has productions

$$(X, t, r+t, r'+t) \longrightarrow a$$

for every $t \in \mathbb{Q}$, whenever $r \xrightarrow{a} r'$. Moreover, for every production $X \longrightarrow YZ$ of $\mathcal{G}$, the grammar $\mathcal{G}'$ has productions

$$(X, t, r+t, r'+t) \longrightarrow (Y, t, r+t, r''+t)\ (Z, t, r''+t, r'+t),$$

for every $t \in \mathbb{Q}$ and for every three tuples r, r', r''.

The productions above are definable with (only equality) constraints of polynomial size. It is an easy exercise to check that the grammar $\mathcal{G}'$ recognizes the same language as the intersection of languages of $\mathcal{A}_1, \dots, \mathcal{A}_n$ and $\mathcal{G}$. □

Lemma IV.4. *An orbit-finite trPDA can be transformed into a language-equivalent trPDA in short form* (5) *of polynomially larger size.*

Proof. Let $\mathcal{A}$ be an orbit-finite trPDA. We define an orbit-finite trPDA $\mathcal{B}$ in short form recognizing the same language. Intuitively, $\mathcal{B}$ keeps in the state a prefix of the stack long enough to apply rules of $\mathcal{A}$ without directly looking at the stack. Thus, states in $\mathcal{B}$ are pairs (q, v) where $v \in S^*$ is a prefix of a lhs/rhs of a rule of $\mathcal{A}$. Since projection and finite union preserve orbit-finiteness, $\mathcal{B}$ has an orbit-finite set

of states. By Lemma III.4 the set is definable and effectively computable. For every rule (q, v, a, q', v') in $\mathcal{A}$ there exists a rule $\text{NOP}((q, v), a, (q', v'))$ in $\mathcal{B}$. Moreover, for every state (q, vs) in $\mathcal{B}$, there exist rules $\text{POP}((q, v), s, \varepsilon, (q, vs))$ and $\text{PUSH}((q, vs), \varepsilon, (q, v), s)$ in order to load/unload the local buffer of $\mathcal{B}$. The language is preserved by this transformation, and the size of $\mathcal{B}$ in short form grows only polynomially with respect to the size of $\mathcal{A}$. □

Lemma C.1. *Non-emptiness of trPDA with timeless stack is* `ExpTime`*-hard.*

Proof. As in [5] (cf. Theorem IV.3), we reduce from the non-emptiness problem of the intersection of the languages recognized by n NFAs $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and a PDA $\mathcal{B}$. This time, timed registers in the state are used to simulate the control states of the NFAs and the PDA, while the untimed pushdown simulates the pushdown of the PDA. We omit the details since they are very similar to Theorem IV.3. □

Appendix D
Proofs missing in Sec. V

A. Systems of equations

Lemma V.2. *The non-emptiness and membership problems of systems of equations are both NP-complete. The membership problem is NP-hard already for intersection-free systems.*

Proof. NP-hardness of the membership problem follows from [16], where it is shown that membership is already NP-hard when restricted to intersection-free systems with only non-negative constants $\{0, 1\}$. Moreover, the membership problem for $k \in \mathbb{Z}$ in X easily reduces in polynomial time to non-emptiness (by using intersection): it suffices to introduce a new variable X' and a new inclusion $X' \supseteq X \cap \{k\}$. Then X contains k in the old system if, and only if, X' is non-empty in the new system. The former inclusion can be simulated with only constants $\{0, 1\}$ by looking at the binary representation of k and by introducing polynomially many new variables and inclusions. Thus, NP-hardness of the non-emptiness problem follows from NP-hardness of membership.

It remains to show an NP upper bound for the non-emptiness problem. The procedure guesses in advance a sequence of inclusions

$$X_1 \supseteq Y_1 \cap \{0\}, \qquad \ldots, \qquad X_n \supseteq Y_n \cap \{0\}$$

from Δ, and then checks correctness of the guess by invoking membership tests. Let Δ' be obtained from Δ by removing all inclusions that use intersection. For every $0 \leq i < n$, let Δ_i be Δ' with the inclusions $X_1 \supseteq \{0\}, \ldots, X_i \supseteq \{0\}$. The procedure checks that 0 is in the least solution for Y_{i+1} in Δ_i. Each of these checks can be done in NP, as they reduce to non-emptiness of the intersection of the Parikh images of two context-free languages:

- the language of a context-free grammar over $\{-1, 0, 1\}$ obtained from Δ_i by replacing addition with concatenation (as in the proof of Lemma V.1), and
- the language over $\{-1, 0, 1\}$ containing words with the same number of -1's and 1's.

The non-emptiness of the intersection can be checked in NP by Kopczyński and To [17].

It remains to argue for correctness. Let ν be the least solution for Δ, and, for every i, let ν_i be the least solution for the guessed Δ_i. By construction we have $\nu_1 \subseteq \nu_2 \subseteq \cdots \subseteq \nu_n \subseteq \nu$, therefore a right guess yields the correct answer. On the other side, suppose $k \in \nu(X)$ for some $k \in \mathbb{Z}$, and let t be a derivation of this fact constructed according to the following rules:

$$\frac{}{k \in X} \quad \text{for every } X \supseteq \{k\}$$

$$\frac{0 \in Y}{0 \in X} \quad \text{for every } X \supseteq Y \cap \{0\}$$

$$\frac{k \in Y \quad l \in Z}{k + l \in X} \quad \text{for every } X \supseteq Y + Z$$

The derivation is finite since we are considering least solutions. Given t, let $t_0, t_1, \ldots, t_n$ be all sub-derivations (subtrees) of t s.t. t_i proves a goal of the form $0 \in Y_{i+1}$. We further assume that t_i is not a subtree of any previous t_j with $1 \leq j < i$. The derivation t_i can be used to show that 0 belongs to $\nu_i(Y_{i+1})$. Thus, the algorithm correctly guesses and verifies $X_1 \supseteq Y_1 \cap \{0\}, \ldots, X_n \supseteq Y_n \cap \{0\}$. □

B. Proof of Decomposition Lemma

Lemma V.8 (Decomposition Lemma)**.** *For a definable subset $X \subseteq Q^2$, its inverse image decomposes into a finite union of sets of the form*

$$\{o\} \times I,$$

where o is an orbit in $\ddot{Q}$, and $I \subseteq \mathbb{Z}$ is one of

$$\mathbb{Z}_{<m} = \{z : z < m\}, \quad \{m\}, \quad \mathbb{Z}_{>m} = \{z : z > m\},$$

for $m \in \mathbb{Z}$. A decomposition of X is computable in `ExpTime`*.*

Proof. The proof proceeds similarly as the proof of the Normal Form Lemma. Consider the set of pairs of states extended with reference points (cf. Sec. V-B):

$$\ddot{X} = \left\{ (q \cdot t, q' \cdot t') \in \dot{Q}^2 \mid (q, q') \in X, \quad t, t' \in \mathbb{Q} \right\}.$$

The set $\ddot{X}$ is definable. As in the proof of the Normal Form Lemma, we decompose $\ddot{X}$ into a finite union of orbits, and K-extensions of orbits O, for a sufficiently large positive integer K, namely greater than the largest span of a state from Q. Thus a state admits no gap K or larger, and therefore a gap K may only be caused by a large distance *between* the reference points of two states.

Consider only those orbits $O \subseteq \ddot{X}$ where the difference of reference points is an integer (the property is an invariant of an orbit); call these orbits *integer-difference orbits*.

Every integer-difference orbit O uniquely determines a pair

$$(o, z_0), \tag{14}$$

for o an orbit in $\ddot{Q}$ and $z_0 \in Z$ the difference of reference points, with $-2 \cdot K \leq z_0 \leq 2 \cdot K$. Furthermore, consider K-extension of an integer-difference orbit O. The integer-difference orbits included in the K-extension jointly determine one of the two sets,

$$\{o\} \times \mathbb{Z}_{\geq z_0} \qquad \text{or} \qquad \{o\} \times \mathbb{Z}_{\leq z_0}. \tag{15}$$

Therefore, the decomposition of the inverse image of X contains singletons of all pairs listed in (14), and the sets listed in (15). $\square$