Parity games and reachability in infinite-state systems with parameters

Mathieu Hilaire

université
PARIS-SACLAY

Parity games and reachability in infinite-state systems with parameters

Jeux de parité et problème d'accessibilité dans des systèmes à infinité d'états avec paramètres

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 : sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : informatique
Graduate School : Informatique et sciences du numérique. Référent : ENS Paris-Saclay

Thèse préparée dans le **Laboratoire Méthodes Formelles** (Université Paris-Saclay, CNRS), sous la direction de **Stefan GÖLLER**, Professeur à l'Université de Kassel, Allemagne, et le co-encadrement de **Benedikt BOLLIG**, directeur de recherche au CNRS.

Thèse soutenue à Paris-Saclay, le 13 Décembre 2022, par

Mathieu HILAIRE

Composition du jury

Membres du jury avec voix délibérative

Nathalie Bertrand Directrice de recherche à Inria Rennes	Présidente
Véronique Bruyère Professeure à l'Université de Mons, Belgique	Rapporteur & Examinatrice
Joël Ouaknine Scientific Director, Max Planck Institute for Software Systems, Allemagne	Rapporteur & Examinateur
Étienne André Professeur à l'Université de Lorraine	Examinateur
Jiri Srba Professeur à l'Université de Aalborg, Danemark	Examinateur

THESE DE DOCTORAT

NNT : 2022UPASG095

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Jeux de parité et problème d'accessibilité dans des systèmes à infinité d'état avec paramètres
Mots clés : Automates temporisés paramétriques, automates à compteur paramétriques, automates à pile paramétriques, problème d'accessibilité, jeux de parité.

Résumé :

Les approches standard de vérification de modèle se limitent à des spécifications concrètes, par exemple "est-il possible d'atteindre une configuration après que plus de 10 unités de temps se soient écoulées ?". Néanmoins, pour certains types de programmes informatiques, comme les technologies embarquées, les contraintes dépendent de l'environnement. De là émerge la nécessité de spécifications paramétriques, par exemple "est-il possible d'atteindre une configuration après que plus de p unités de temps se soient écoulées ?" où p désigne un paramètre dont la valeur reste à spécifier.

Dans cette thèse nous étudions des variantes paramétriques de trois modèles classiques, les automates à pile, à compteur, et temporisés. En plus d'exprimer des contraintes concrètes (sur la pile, le compteur ou les horloges), ces modèles peuvent exprimer des contraintes paramétriques via des comparaisons avec des paramètres. Le problème de l'accessibilité consiste à demander s'il existe une assignation des paramètres telle que il existe une exécution acceptante dans l'automate concret en résultant. Nous étudions également des jeux de parité paramétriques, où deux joueurs choisissent une évaluation pour chaque paramètre chacun leur tour, puis déplacent un jeton dans le graphe de l'automate concret résultant. Nous nous intéressons au problème de décider quel joueur dispose d'une stratégie gagnante.

Les automates temporisés paramétriques ont été introduit dans les années 90 par Alur, Henzinger et Vardi, qui ont démontré que le problème de l'accessibilité était indécidable, même avec seulement trois horloges pouvant être comparées à des paramètres, ou horloges paramétriques — mais décidable dans le cas d'une seule horloge paramétrique. Des résultats récents de Bundala et Ouaknine démontrent que dans le cas de deux horloges paramétriques et un paramètre, le problème est décidable ainsi que PSPACE$^{\text{NEXP}}$-dur. Un des principaux résultats de cette thèse consiste à démontrer le caractère EXPSPACE-complet du problème. La borne inférieure EXPSPACE repose sur des résultats récents de complexité avancée. Inspirés par Göller, Haase, Ouaknine, et Worrell, nous visualisons la classe EXPSPACE comme un langage de feuille (vision qui repose sur le théorème de Barrington). Nous introduisons un langage de programmation qui peut reconnaître ce genre de langage de feuille. Nous utilisons ensuite une représentation des entiers sous forme de restes chinois dont Chiu, Davida et Litow ont montré qu'elle permettait de calculer en LOGSPACE la représentation binaire, et démontrons grâce à elle que les automates temporisés paramétriques à deux horloges paramétriques et un paramètre peuvent simuler notre langage de programmation. Pour la borne supérieure, à la façon de Bundala et Ouaknine, nous réduisons le problème au problème de l'accessibilité dans un type particulier d'automates à compteur paramétrique. Nous démontrons que ce problème est dans PSPACE. Étant donné que notre réduction se fait en temps exponentiel, cela conduit à une borne supérieure EXPSPACE pour le problème de l'accessibilité dans les automates temporisés paramétriques à deux horloges et un paramètre.

En ce qui concerne les jeux de parités paramétriques pour les automates à piles paramétriques, nous démontrons que le problème de déterminer quel joueur a une stratégie gagnante est dans $(n+1)$-EXP si le nombre de paramètres est fixé à n, mais nonélémentaire sinon. Le caractère nonélémentaire du problème est démontré par réduction du problème de la satisfaisabilité des formules de la logique du premier ordre. La borne supérieure est obtenue via une réduction aux automates à pile de piles, pour lesquels le problème des jeux de parités est n-EXP-complet dans le cas de n niveaux de piles.

Title : Parity games and emptiness in infinite-state systems with parameters
Keywords : Parametric timed automata, parametric one-counter automata, parametric pushdown automata, reachability, parity games.

Abstract :

The most standard model checking approaches are limited to verifying concrete specifications, such as "can we reach a configuration with more than 10 time units elapsing ?". Nevertheless, for certain computer programs, like embedded systems, the constraints depend on the environment. Thus arises the need for parametric specifications, such as "can we reach a configuration with more than p time units elapsing ?" where p is a parameter which takes values in the non-negative integers.

In this thesis, we study parametric pushdown, counter and timed automata and extensions thereof. In addition to expressing concrete constraints (on the stack, on the counter or on clocks), these can employ parametric constraints. The reachability problem for a parametric automaton asks for the existence of an assignment of the parameters such that there exists an accepting run in the underlying concrete automaton. In addition to the reachability problem, we consider parametric parity games, two player games where players alternate choosing assignments for each parameters, then alternate moving a token along the configurations of the concrete automaton resulting from their choice of parameter assignment. We consider the problem of deciding which player has a winning strategy.

Parametric timed automata (PTA for short) were introduced in the 90s by Alur, Henzinger and Vardi, who showed that the reachability problem for PTA was undecidable, already when only three clocks can be compared against parameters, and decidable in the case only one clock can. We call such clocks that can be compared to parameters parametric clocks. A few years ago, Bundala and Ouaknine proved that, for parametric timed automata with two parametric clocks and one parameter ($(2,1)$-PTA for short), the reachability problem is decidable and also provided a PSPACE$^{\text{NEXP}}$ lower bound. One of the main results of this thesis states that reachability for $(2,1)$-PTA is in fact EXPSPACE-complete. For the EXPSPACE lower bound, inspired by previous work by Göller, Haase, Ouaknine, and Worrell, we rely on a serializability characterization of EXPSPACE (in turn originally based on Barrington's Theorem). We provide a programming language and we show it can simulate serializability computations. Relying on a logspace translation of numbers in Chinese Remainder Representation to binary representation due to Chiu, Davida, and Litow, we then show that small $(2,1)$-PTA can simulate our programming language. For the EXPSPACE upper bound on $(2,1)$-PTA, we first give a careful exponential time reduction towards a variant of parametric one-counter automata over one parameter based on a minor adjustment of a construction due to Bundala and Ouaknine. We solve the reachability problem for this parametric one-counter automata with one parameter variant, by providing a series of techniques to partition a fictitious run into several carefully chosen subruns. This allows us to prove that it is sufficient to consider a parameter value of exponential magnitude only, which in turn leads to a doubly-exponential upper bound on the value of the only parameter of the $(2,1)$-PTA. We hope that extensions of our techniques lead to finally establishing decidability of the long-standing open problem of reachability in PTA with two parametric clocks and arbitrarily many parameters.

Concerning parametric pushdown automata, our main result states that deciding the winner of a parametric parity game is in $(n+1)$-EXP in the case the number of parameters n is fixed, but nonelementary otherwise. We provide the nonelementary lower bound via a reduction of the FO satisfiability problem on words. For the upper bound, we reduce parametric pushdown parity games to higher-order pushdown automata parity games, which are known to be n-EXP complete in the case of stacks of level n.

Remerciements

Je voudrais tout d'abord remercier mon directeur de thèse Stefan Göller ainsi que mon co-encadrant de thèse Benedikt Böllig, pour leurs conseils précieux, leur soutien et leurs encouragements. Nos discussions m'ont apporté beaucoup d'expérience et de connaissances d'une valeur inestimable.

Je voudrais également remercier Véronique Bruyère et Joël Ouaknine d'avoir accepté d'être rapporteurs de ce document, ainsi que Nathalie Bertrand, Étienne André et Jiri Srba pour avoir accepté d'être examinateurs pour mon jury.

Je tiens à remercier le LSV, ou plutôt devrais-je dire maintenant le LMF, pour son cadre de travail agréable et bienveillant. Je remercie également l'université de Kassel de m'avoir accueilli chaleureusement à de nombreuses reprises pour ce qui fut des semaines de travail intense. Merci à Da-Jung pour nous avoir parlé de l'extension d'edition de dessins tex ipe qui il faut le dire est bien pratique.

Je tiens à remercier les reviewers anonymes de STACS 2021 et FSTTCS 2022 pour leurs commentaires utiles et leurs suggestions instructives.

Cette thèse n'aurait pas pu être terminée sans le financement généreux de l'Agence Nationale de la Recherche (financement no. ANR-17-CE40-0010).

Je remercie aussi mes amis, en particulier ceux avec qui j'ai passé des semaines de vacances à Saumur et qui m'ont aidé à me changer les idées dans les périodes difficiles, ainsi que tous ceux qui sont venu me rendre visite pour passer un bon moment ensemble même après qu'on se soit tous éparpillés aux quatre coins de la france.

Je remercie ma famille, mes parents et mes frères pour leur support et leur confiance tout au long de longues années au cours desquelles cela n'a pas été toujours facile pour eux.

Merci à ma petite boule de poil Paige pour sa douceur réconfortante. Merci enfin à Camille pour avoir accepté de partager ma vie, pour avoir été là tout au long, et pour son amour et son soutien incommensurable.

Résumé en Français

Les approches standard de vérification de modèle se limitent à des spécifications concrètes, par exemple “est-il possible d'atteindre une configuration après que plus de 10 unités de temps se soient écoulées?”. Néanmoins, pour certains types de programmes informatiques, comme les technologies embarquées, les contraintes dépendent de l'environnement. De là émerge la nécessité de spécifications paramétriques, par exemple “est-il possible d'atteindre une configuration après que plus de p unités de temps se soient écoulées?” où p désigne un paramètre dont la valeur reste à spécifier.

Dans cette thèse nous étudions des variantes paramétriques de trois modèles classiques, les automates à pile, à compteur, et temporisés. En plus d'exprimer des contraintes concrètes (sur la pile, le compteur ou les horloges), ces modèles peuvent exprimer des contraintes paramétriques via des comparaisons avec des paramètres. Le problème de l'accessibilité consiste à demander s'il existe une assignation des paramètres telle que il existe une exécution acceptante dans l'automate concret en résultant. Le problème de l'accessibilité n'est pas le seul qui nous intéresse et nous étudions également des jeux de parité paramétriques, où deux joueurs choisissent une évaluation pour chaque paramètre chacun leur tour, puis déplacent un jeton dans le graphe de l'automate concret résultant. Dans ce cas de figure, nous nous intéressons au problème de décider quel joueur dispose d'une stratégie gagnante.

Les automates temporisés paramétriques ont été introduit dans les années 90 par Alur, Henzinger et Vardi [5], qui ont démontré que le problème de l'accessibilité était indécidable, même avec seulement trois horloges pouvant être comparées à des paramètres, mais décidable dans le cas où une seule horloge peut l'être. Nous appelons les horloges pouvant être comparées à des paramètres horloges paramétriques. Des résultats récents de Bundala et Ouaknine [17] démontrent que dans le cas de deux horloges paramétriques et un paramètre, le problème est décidable ainsi que $\mathsf{PSPACE}^{\mathsf{NEXP}}$-dur. Un des principaux résultats de cette thèse consiste à démontrer le caractère EXPSPACE-complet du problème. La borne inférieure EXPSPACE repose sur des résultats récents de complexité avancée. Inspirés par Göller, Haase, Ouaknine, et Worrell [47, 49], nous visualisons la classe EXPSPACE comme un langage de feuille (vision qui repose sur le théorème de Barrington [9]). Nous introduisons un langage de programmation qui peut reconnaître ce genre de langage de feuille. Nous utilisons ensuite une représentation des entiers sous forme de restes chinois dont Chiu, Davida et Litow [24] ont montré qu'elle permettait de calculer en LOGSPACE la représentation binaire, et démontrons grâce à elle que les automates temporisés paramétriques à deux horloges paramétriques et un paramètre peuvent simuler notre langage de programmation. Pour la borne supérieure, à la façon de Bundala et Ouaknine [17], nous réduisons le problème au problème de l'accessibilité dans un type particulier d'automates à compteur paramétrique. Nous démontrons que ce problème est dans PSPACE à l'aide d'une série de techniques pour partitionner une exécution éventuelle en différentes sous-parties que nous modifions ensuite bout à bout. Ces techniques nous permettent de prouver qu'il est suffisant de considérer un paramètre dont la magnitude est exponentiellement bornée. Étant donné que notre réduction se fait en temps exponentiel, cela conduit à une borne supérieure EXPSPACE pour le problème de l'accessibilité dans les automates temporisés paramétriques à deux horloges et un paramètre.

En ce qui concerne les jeux de parités paramétriques pour les automates à piles paramétriques, nous démontrons que le problème de déterminer quel joueur a une stratégie gagnante est dans $(n+1)$-EXP si le nombre de paramètres est fixé à n, mais nonélémentaire sinon. Le caractère nonélémentaire du problème est démontré par réduction du problème de la satisfaisabilité des formules de la logique du premier ordre. La borne supérieure est obtenue via une réduction aux automates à pile de piles, pour lesquels le problème des jeux de parités est n-EXP-complet dans le cas de n niveaux de piles [19].

Contents

Introduction

Background

The goal of specification and verification is to study mathematical models that allow us to analyse the behaviors of programs. Indeed, in order to make sure that a program functions as intended, even for something as straightforward as a sorting algorithm, there can only be two approaches. The first approach consists in making sure that the program functions as intended for every possible input by executing the program for each of them. This is not in general possible since programs often have an unbounded number of potential input, and, even when they do have a bounded number of potential entries, this approach remains extremely time-consuming. The other approach is to build a mathematical framework representing the program, translate the properties one wants the program to meet into logical formulas, and verify that the mathematical structures statisfy the formulas. This approach is known as model checking, and was initially proposed by Clarke and Emerson [35, 26] and independently by Quielle and Sifakis [86, 87]. The model checking problem asks, given a requirement expressed as a logical formula ϕ, and a program represented as an abstract structure $\mathfrak{A}$, whether the program meets the requirement i.e. whether the structure satisfies the formula. The requirements considered are often *safety* properties, stating that a "bad" state never occurs, or *liveness* properties, stating that a "good" state will happen eventually [2]. In particular, the requirements considered can usually be expressed using temporal logics such as linear temporal logic (LTL) [85, 103] or computational tree logic (CTL) [26, 27], which can express safety and liveness properties [96, 70, 61, 82]. Another branching time logic is the modal μ-calculus [92, 69], which provides a unifying framework subsuming many other logics of interest including both LTL, CTL and their superset CTL* [37, 34]. With this approach, making sure a program meets critical design requirements becomes a mathematical question. Since our dependence on programs is only ever increasing, it is crucial to be able to provide guarantees that programs function as intended.

There are two competing goals for mathematical models for specification and verification. The first is expressivity: one wants models to represent a large body of possible programs, otherwise they lack applications. For instance, the model of Turing machines is capable of implementing any computer algorithm. The other requirement for mathematical models is to have desirable decidability properties. As mentioned, the goal of model checking is to verify that some mathematical structure satisfy some logical formula. Thus, one wishes for model checking formula for common logics (LTL, CTL, CTL* or the modal μ-calculus for instance) to be decidable for the mathematical structures one uses to model programs. A simpler question than that of decidability of the μ-calculus model checking over mathematical structures is that of the reachability problem, which asks for the existence of an execution that ends in a state belonging to a set of final states. Perhaps one of the most important problems of verification, the reachability problem can already be used to express safety properties, that is, to rule out the existence of an execution to a "bad" state. Turing machines, despite or rather because of their high expressivity, are known to have bad decidability properties. For instance the halting problem, that is the problem of determining, given a Turing machine and an input, whether the execution of the Turing machine on the given input will finish running or continue to run forever, is already undecidable [101, 31]. Automata are another example of a mathematical framework used to model programs. They have good decidability properties, but lack expressivity and cannot be used to represent certain programs. For instance when real-time systems are considered, it is desirable to prove quantitative properties of such systems, for example

that a certain action is always performed within a certain amount of time. Other examples include recursive programs, and programs with variables over integers or other infinite domains. To provide formalisms for such systems, various extensions of automata have been developed and studied. Although such extensions are in general infinite objects, we are interested in infinite-state systems that can finitely be described.

Notice that the question mentioned above "is a certain action always perfomed within a certain amount of time t ?" is a concrete question. Such concrete timing questions can be answered using concrete automata extensions, for instance timed automata [3, 4], automata extended with a finite set of clocks that progress at the same rate, can be reset to zero and compared against concrete timing constraints. In real life, however, the requirements one wants to ensure often depend on the environment. Programs in embedded systems are an example of such a situation. This leads to the need to reframe verification problems in parametric terms, i.e. ask rather the question "is there a threshold $N \in \mathbb{N}$ such that a certain action will always be performed within N units of time ?". For studying such problems, Alur, Henzinger and Vardi laid the foundations for parametric reasoning about real time in their seminal paper [5]. They introduced parametric timed automata, which extend timed automata with a set of parameters and the ability to employ parametric constraints. This parametric extension can be generalized to other models, that is, we can incorporate a set of parameters P to the finite presentation of, for example, a counter automaton. A parameter valuation for the parametric automaton is then a function from P to some set M (in the example, $\mathbb{N}$), which leads to a concrete automaton. In the case of an automaton with parameters (e.g. a parametric timed automaton), a finite presentation represents a family of infinite-state systems, as there is one concrete infinite-state system for every possible parameter valuation. Verification questions can then be asked in parametric terms. Parametric verification has applications for instance for the study of embedded systems [25], where the constraints depend on the environment, and to capture uncertainties in timing behaviors [45]. A *parametric reachability problem*, given an automaton with parameters, asks whether there is some parameter valuation such that the resulting system can reach a final state. We will generally omit "parametric" and write simply "reachability problem" when it is clear from context that the input automaton has parameters.

We want to reframe other model checking problems in parametric terms as well. Several model checking problems can be encoded in terms of infinite two-player games, hence we are interested in solving two player *parametric games*. In general a game is composed of an arena — a graph where configurations are assigned to one of the two players, player 0 and player 1 — and a winning condition — a set of (possibly infinite) sequences of configurations. A play is a (possibly infinite) sequence of configurations, and is winning for player 0 if it lies in the winning condition. In a reachability game, the winning condition is the set of all plays containing a final configuration. In a parity game, the winning condition is rather given by a priority function assigning a color to every configuration of the graph. The winner of a finite play is the player whose opponent is unable to move, and the winner of an infinite play is determined by the priorities appearing in the play, with player 0 winning if the largest priority that occurs infinitely often in the play is even, and player 1 winning if the largest priority that occurs infinitely often in the play is odd. Two player parity games have applications to model checking properties, since the modal μ-calculus model checking is polynomially equivalent to the problem of solving parity games [16]. We reframe parity games in parametric terms too. In the case of an automaton with parameters, the choice of parameter valuation then becomes another component of the game. Players alternate choosing values for the parameters, and then alternate choosing successor configurations in the induced concrete system. Winning conditions are defined as in the concrete case.

Scope and Contributions

We are interested in analysing parametric pushdown, one-counter and timed automata, i.e. automata extended with a stack, counter or clocks that can be compared with parameters. These parameters can take unspecified values over infinite domains, which here consists in the set of words (for parametric pushdown automata) or the non-negative integers (for parametric one-counter automata and parametric timed automata). Our interests lies in the parametric variants of requirements

such as safety or liveness properties, in the form of the parametric reachability problem and the problem of solving two player parametric games.

Parametric pushdown automata

Pushdown automata are an extension of automata allowing access to an infinite memory in the form of a stack that can be manipulated in a "first-in/last-out" fashion. Despite having access to an infinite memory, they retain desirable algorithmic properties. The reachability problem for pushdown automata indeed can be solved in polynomial time [14, 40], and the μ-calculus model checking problem can be solved in exponential time [104]. Pushdown automata are however more expressive than classical automata, with applications to modeling recursive programs or to reason about quantitative properties of systems [29, 41].

We define parametric pushdown automata (PPDA for short) as pushdown automata extended with a finite set of parameters, and such that the stack can be checked against the parameters, that can take values over the set of words over stack symbols. The study of games over graphs generated by pushdown automata have been essential, in particular, the proof that the μ-calculus model checking problem for pushdown automata is EXP-complete makes use of a reduction to pushdown parity games [104]. Since parity games played on graphs generated by pushdown automata have been essential to the study of model checking pushdown automata, we will study games over graphs generated by PPDA.

Our main result concerning parametric pushdown automata is that the problems of solving parametric reachability games and parametric parity games are in $(n+1)$-NEXP in case the number of parameters n is fixed, but are nonelementary in the case of arbitrarily many parameters. For the nonelementary lower bound on parametric pushdown reachability games, we reduce the FO satisfiability problem on words, known to be nonelementary from [99], to the problem of deciding whether player 0 has a winning strategy for the parametric reachability game generated by a parametric pushdown automaton. For the upper bound, we start by replacing parameters by pebbles acting as registers, leading to a more general model. We then reduce our problem to higher-order pushdown automata parity games, where every pebble is simulated by using an additional stack level. Since solving parity games on higher-order pushdown automata with level n stack is n-EXP-complete [19, 20], this provides an $(n+1)$-EXP upper bound for solving parametric parity games on parametric pushdown automata with n parameters.

Parametric one-counter automata

Counter automata are an extension of automata allowing access to an infinite memory, this time in the form of counters that can store integers that can be incremented, decremented and tested for being zero. Unfortunately, this extension quickly leads to undecidability. Indeed, for every Turing machine, there is a two counter automaton that simulates it [79], thus the reachability problem for two-counter automata is undecidable. On the positive side the reachability problem for one-counter automata with unary updates can be solved in nondeterministic linear time [32], and it can be solved in nondeterministic polynomial time in the case of one-counter automata allowing updates by constants written in binary, also called succinct one-counter automata [52]. One-counter automata retain desirable algorithmic properties, even when they are extended to allow new types of tests, such as testing that the value of the counter belongs to a specific interval. Counter automata are a useful formalism to reason about programs with pointers and linked lists [13, 97]. Of note is that one-counter automata can be seen as a special case of pushdown automata with only one stack symbol plus a bottom-of-stack symbol.

Parametric one-counter automata (POCA for short) provide a formalism to reason about programs with behaviors that make use of parametric constraints or updates. Here, the counter can additionally be incremented or decremented by parameters that can take unspecified non-negative integer values. POCA are used in various synthesis problems, to model resources like time or memory being consumed by transitions [107] and to model open programs whose behavior is parameterized by some input values. For instance it is useful to model procedures embedded in a

larger program. The reachability problem for POCA asks for the existence of an assignment of the parameters to the non-negative integers such that there exists an execution that ends in a state belonging to a set of final states in the underlying one-counter automaton. The reachability problem in POCA is NP-hard since POCA includes succinct one-counter automata. The problem has also been shown to be reducible to satisfiability in quantifier-free Presburger arithmetic with divisibility [52], providing a NEXP upper bound [75].

We consider an extension of POCA where we allow the counter to be compared against parameters or against constants, which we call parametric threshold one counter automata (PTOCA). A recent construction by Bundala and Ouaknine [17] ensures that reachability in PTOCA is decidable in the case of one parameter. We study PTOCA with one parameter and consider, for an assignment of the parameter to the non-negative integers, a fictitious execution where the counter values are bounded by the value assigned to the parameter multiplied by some constant h that depend only on the PTOCA. By a careful analysis, we prove that the existence of such an execution implies the existence of an execution for an assignment of the parameter of lesser magnitude. This in turn allow us to prove a PSPACE upper bound for the reachability problem for a (slight subclass of) PTOCA with one parameter. We hope that extensions of our techniques lead to establishing decidability of reachability in PTOCA with arbitrarily many parameters, and, in case decidability holds, determining its precise computational complexity.

Parametric timed automata

Timed automata [4] are an extension of automata allowing access to an infinite memory in the form of clocks that all progress at the same rate, that can be compared against specific constants, and that can be reset to zero. They too retain desirable decidability properties despite the addition of an infinite memory. The reachability problem for timed automata is indeed decidable and can be solved using only polynomial space [3]. Timed automata provide maybe the most popular formalism to reason about the behavior of real-time systems [98, 18].

Parametric timed automata (PTA for short) are an extension of timed automata in which clocks can additionally be compared against parameters that can take unspecified non-negative integer values. A clock of a PTA that is being compared to at least one parameter is called *parametric*. The reachability problem for parametric timed automata asks for the existence of an assignment of the parameters to the non-negative integers such that there exists an execution that ends in a state belonging to a set of final states in the underlying timed automaton. On the negative side, it has been shown in [5] that already for PTA that contain *three parametric clocks* reachability is undecidable — even in the presence of a single parameter [10]. For PTA over *one parametric clock*, the problem is NEXP-complete [17, 10]. Decidability of reachability in PTA over two parametric clocks (without parameter restrictions) is still considered to be a challenging open problem to the best of our knowledge. For instance, as already remarked in [5], there is an easy reduction from the existential fragment of Presburger Arithmetic with divisibility to reachability in PTA over two parametric clocks. In the presence of one parameter the problem has been shown to be decidable and $\mathsf{PSPACE}^{\mathsf{NEXP}}$-hard [17].

Our main result concerning parametric timed automata is the EXPSPACE-completeness of the reachability problem for parametric timed automata with two parametric clocks and one parameter.

For the EXPSPACE lower bound, inspired by [47, 49], we rely on a serializability characterization of EXPSPACE (in turn originally based on Barrington's Theorem [9]). We provide a programming language that we show can simulate serializability computations. Relying on a logspace translation of numbers in Chinese Remainder Representation to binary representation due to Chiu, Davida, and Litow [24], we then show that with small PTA over two parametric clocks and one parameter one can simulate the programming language.

For the EXPSPACE upper bound, we first give a careful exponential time reduction from PTA over two parametric clocks and one parameter to the (slight subclass of) PTOCA over one parameter mentioned above. Our construction is based on a minor adjustment of a construction due to Bundala and Ouaknine [17]. In solving the reachability problem for parametric one-counter automata with one parameter, we refer to our results on PTOCA reachability, that allows us to

prove that it is sufficient to consider a parameter value of exponential magnitude. This allows us to show a doubly-exponential upper bound on the value of the only parameter of PTA with two parametric clocks and one parameter. We hope that extensions of our techniques lead to finally establishing decidability of the long-standing open problem of reachability in PTA with two parametric clocks (and arbitrarily many parameters) and, if decidability holds, determining its precise computational complexity.

Overview of the thesis

In Chapter 1, we provide general notations and preliminary definitions. Chapter 2 will deal with the introduction and motivation of infinite-state systems. Chapter 3 will introduce the notion of automata extended with parameters, and parametric variants of classic problems such as reachability, reachability games and parity games. Chapter 4 will deal with the model of parametric one-counter automata. Chapter 5 will deal with a closely related model, that of parametric timed automata. More precisely, we will discuss the complexity of the parametric reachability problem, which we show to be EXPSPACE-complete in the case of parametric timed automata with one parameter and two parametric clocks. Chapter 6 will deal with the model of parametric pushdown automata. We concern ourselves here with parametric reachability games and parametric parity games. We determine that the problems are in $(n+1)$-EXP in the case the number of parameters n is fixed, and provide a nonelementary lower bound in the general case.

Publications

Some of the results presented in this thesis have already been published in the article [48] coauthored with my supervisor Stefan Göller, namely the results from Chapter 4 and Chapter 5.

Chapter 1

Definitions

In this chapter, we introduce general notations and preliminary definitions. First, we concern ourselves with basic mathematical objects and notations: numbers, words, (partial) functions. Secondly, we introduce the general definition of transition systems. Thirdly, we define parity games. Fourth, we define the complexity classes that will be of interest to us. Lastly, we recall the definition of automata.

1.1 Preliminaries

For any two sets X and S, let X^S denote the set of all functions from S to X. For any set S let $\mathscr{P}(S) = \{X \mid X \subseteq S\}$ denote the *power set* of S. If a set S is finite, we denote its *size* by $|S|$.

1.1.1 Numbers

By $\mathbb{Z}$ we denote the *integers* and by $\mathbb{N} = \{0, 1, \ldots\}$ we denote the *non-negative integers*. For every $a, b \in \mathbb{Z}$ with $a \leq b$ we define $[a, b] = \{k \in \mathbb{Z} \mid a \leq k \leq b\}$. For every $n \geq 1$ we define $n\mathbb{Z} = \{n \cdot z \mid z \in \mathbb{Z}\}$. For every number $n \in \mathbb{N}$ we define $\log(n) = \min\{i + 1 \mid i \in \mathbb{N}, n \leq 2^i\}$, which is the number of bits necessary to write down n in binary. For every finite set $M \subset \mathbb{N} \setminus \{0\}$ let $\mathsf{LCM}(M) = \min\{n \geq 1 \mid \forall m \in M : m|n\}$ denote the least common multiple of the elements in M. For any $j \in \mathbb{N}$ let $\mathsf{LCM}(j) = \mathsf{LCM}([1, j])$ denote the *least common multiple* of the numbers $\{1, \ldots, j\}$. For a real $r \in \mathbb{R}$, $\lfloor r \rfloor$ is the greater integer z such that $z \leq r$.

1.1.2 Alphabets and words

For every set A we denote by A^* the set of finite sequences of elements of A, and we denote by A^ω the set of infinite sequences of elements of A. If the set A is finite, A will be called an *alphabet*, and we call the elements of A^* *words* over A, and the elements of A^ω *infinite words* over A. We denote the union of A^* and A^ω by A^∞. We denote the *empty word* by ε. For all $a \in A$ and all $w \in A^*$ let $|w|_a$ denote the number of occurrences of the letter a in w, while $|w|$ denotes the *length* of the word. We denote by A^n the set of words of length $n \in \mathbb{N}$ over A.

For two words $u, v \in A^*$, we denote by $u \cdot v$ (often abreviated uv) the concatenation of the two sequences. We say u is a *prefix* (resp. a *suffix*) of v if there exists a word $w \in A^*$ such that $v = u \cdot w$ (resp. $v = wu$).

For a word $w = a_0 a_1 a_2 \ldots a_{n-1}$ where $a_i \in A$ for all $i \in [0, n-1]$, we denote a_i by $w[i]$ for $0 \leq i \leq n-1$ and call it the *letter* at *position* i. For each language $L \subseteq A^*$ let $\chi_L : A^* \to \{0, 1\}$ denote its characteristic function defined as

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L, \\ 0 & \text{otherwise.} \end{cases}$$

1.1.3 Partial functions

A *partial function* f from a set S to a set X that is defined on a subset $C \subseteq S$ is denoted by $f : S \rightharpoonup X$. We call C (resp. $f(C)$) the *domain* of f (resp. *image*) and write it $\mathrm{Dom}(f)$ (resp. $\mathrm{Im}(f)$). If $C = S$ then f is called *total*.

For a partial function $f : S \rightharpoonup X$, for notational purposes, we will consider some element $\perp_X \notin X$ and f can alternatively be associated with the function returning the bottom element $\perp_X$ when it is undefined. Thus we write $(X \uplus \{\perp_X\})^S$ for the set of all partial functions from S to X, which we sometimes abbreviate as $(X \uplus \{\perp\})^S$.

For a partial function $f : S \rightharpoonup X$, the *inverse image* of an element $x \in X$, denoted by $f^{-1}(x)$, is the set $\{s \in S \mid f(s) = x\}$.

We say a function $f : \mathbb{N} \to \mathbb{N}$ is *monotonic non-decreasing* if f for all $i, j \in \mathbb{N}$ such that $i \leq j$, one also has $f(i) \leq f(j)$. Let f and g denote total functions from $\mathbb{N}$ to $\mathbb{N}$. We say f is *asymptotically bounded by* g and write $f(n) \in O(g(n))$ if there exists a constant $M \in \mathbb{N}$ and a natural number n_o such that

$$f(n) \leq M \cdot g(n)$$

holds for all $n \geq n_o$. We extend the notation to composition of functions i.e. if f, g, h are all total functions from $\mathbb{N}$ to $\mathbb{N}$, h is monotonic non-decreasing and $f(n) \in O(g(n))$, then we sometimes write $h(f(n)) \in h(O(g(n)))$ for $h(f(n)) \in O(h(g(n)))$.

1.2 Transition systems

A *labeled transition system* (LTS for short) is a tuple $T = (S, \Lambda, \to)$ where S is a set of *configurations*, Λ is a set of *labels*, and $\to \subseteq S \times \Lambda \times S$ is a ternary relation, denoted as the set of *labeled transitions*. We prefer to use infix notation and $(s, a, s') \in \to$ will be abbreviated as $s \xrightarrow{a} s'$ to represent a transition from configuration s to configuration s' with label a.

Labels can be used to represent the reading of an input, but also to represent an action performed during the transition or conditions that must hold in order to allow the use of the transition.

A *path* in a labeled transition system from a *source configuration* s_0 to a *target configuration* s_n is a sequence $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} s_n$. We define the *concatenation* $\pi_1\pi_2$ of two paths π_1 and π_2 when the source configuration of π_2 is equal to the target configuration of π_1 as expected. The *length* of $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} s_n$ is defined as $|\pi| = n$. We say the path is *labeled* by $a_0 a_1, \ldots a_{n-1}$. For all $w \in \Lambda^*$, all $s, s' \in S$, we will write $s \xrightarrow{w} s'$ if there exists a path from s to s' labeled by w.

An *infinite path* is an infinite sequence $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$. For each infinite (resp. finite) path $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$ (resp. $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} s_n$) and $i, j \in \mathbb{N}$ (resp. $i, j \in [0, n]$) with $i < j$ we denote by $\pi[i, j]$ the path $s_i \xrightarrow{a_i} s_{i+1} \xrightarrow{a_{i+1}} \cdots \xrightarrow{a_{j-1}} s_j$ and by $\pi[i]$ the configuration s_i. As expected, a *prefix* of a finite or infinite path π is a finite path of the form $\pi[0, j]$, and a *suffix* of a finite path π is a path of the form $\pi[i, n]$.
Given an infinite path $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$ let $\mathit{Inf}(\pi) = \{s \in S \mid \forall i \ \exists j > i \ s_j = s\}$.
The set of *successors* of a configuration $s \in S$ is defined as $\{s' \in S \mid \exists a \in \Lambda \ s \xrightarrow{a} s'\}$. A configuration without successors is called a *dead end*.

A labeled transition system $(S, \Lambda, \to)$ is *deterministic* if for all configurations $s_1, s_2, s_3 \in S$ and all $a \in \Lambda$, $s_1 \xrightarrow{a} s_2$ and $s_1 \xrightarrow{a} s_3$ implies $s_2 = s_3$.

An *(unlabeled) transition system* is a pair $T = (S, \to)$ where S is a set of *configurations* and $\to \subseteq S \times S$ is a binary relation on the set of configurations, denoted as the set of *transitions*. We again prefer to use infix notation and write $s \to s'$ to denote a *transition* from configuration s to configuration s' (i.e., $(s, s') \in \to$).

Note that an unlabeled transition system can be seen as a labeled transition system where the set of labels consists of only one element. Determinism, (infinite) paths, their length, and concatenation in unlabeled transition systems are then defined as expected.

1.3 Games

A game is composed of an arena and a winning condition. We will first study arenas and then introduce common winning conditions that will be of interest to us.

1.3.1 Arenas

An *arena* is a tuple $A = (S_0, S_1, \to)$ which is composed of

- two disjoint sets of configurations, S_0 and S_1, whose disjoint union $S_0 \cup S_1$ we denote by S, and
- a relation of transitions $\to \subseteq S \times S$.

Note that $(S, \to)$, where $S = S_0 \uplus S_1$, forms a transition system, and, in particular, given a transition system, one needs only to provide a partition of the set of configurations S into two sets S_0 and S_1 to obtain an arena.

The games we are interested in are played by two players, called *player* 0 and *player* 1. We will often write player i to denote a general player for $i \in \{0, 1\}$, and we will call player $1-i$ its *opponent*.

1.3.2 Plays

A *play* in an arena $A = (S_0, S_1, \to)$ is a path in $(S, \to)$ that is *maximal* in the following sense: it is either infinite or finite and if it is finite, then the target configuration is a dead end. A *partial play* in an arena $A = (S_0, S_1, \to)$ is a prefix of a play in A.

Essentially plays can be seen as this: the two players, player 0 and player 1, move along the labeled transition system, taking turns either infinitely often or until a dead end is reached.

1.3.3 Strategies

A *strategy* for player $i \in \{0, 1\}$ is a function $\sigma_i : S^* S_i \to S$ such that $\sigma_i(ws)$ is a successor of s. A strategy is called *memoryless* if its output only depends on the final configuration of the sequence $s \in S_i$, i.e. if $\sigma_i(ws) = \sigma_i(w's)$ for all $w, w' \in S^*$. Thus, a memoryless strategy for player i can be written as a function $\sigma_i : S_i \to S$.

A partial play $\pi = s_0 \to s_1 \to \ldots \to s_n$ is *consistent* with a strategy σ_i if sequences of configurations that end in S_i along this play have all successors according to the strategy, i.e. if for all $j \in [0, n-1]$, for all element $v_j \in V_i$ in the sequence π, $v_{j+1} = \sigma_i(v_j)$. Given a configuration s, a strategy σ_0 for player 0 and a strategy σ_1 for player 1, the play starting with s that is consistent with both strategies is unique and is called the *resulting play* of σ_0 and σ_1 starting from s. It is denoted by $\pi(s, \sigma_0, \sigma_1)$.

1.3.4 Winning conditions

Once we have provided players with an arena, it remains to define properly what the winning condition is in order to define a game. Given an arena $A = (S_0, S_1, \to)$, a *winning condition* Win is a subset of the set of maximal plays for A. A *game* is a pair $\mathcal{G} = (A, \text{Win})$ which is composed of

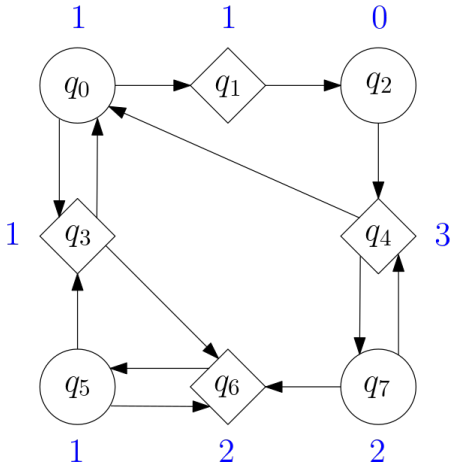

Figure 1.1: An example of an arena. Circles belong to player 0, whereas diamonds belong to player 1. Arrows represent transitions from a configuration to the next. The priority assigned by the priority mapping to each configuration is a number in $\{0, 1, 2, 3\}$, next to the configuration it is assigned to. Note that we do not have a dead end in our example.

an arena A and a winning condition Win. Although a winning condition is in general an infinite object we are interested in winning conditions that can finitely be described.

A strategy σ_0 for player 0 from position $s \in S$ is a *winning strategy from* s for player 0 if every maximal play starting from s and consistent with σ_0 is in Win. On the other hand, a strategy σ_1 for player 1 from position $s \in S$ is a *winning strategy from* s for player 1 if every maximal play starting from s and consistent with σ_1 is not in Win.

Reachability Games We first consider reachability games: the winning condition $\text{Win}_F(S)$ is given by a set of final configurations $F \subseteq S$, and $\text{Win}_F(S)$ is the set of maximal plays in S^*FS^∞. We simply write Win_F in case the set S is obvious from context. To indicate that the winning condition of a game is a reachability winning condition, we will speak of *reachability games*.

With regards to reachability games, we are going to concern ourselves with games over potentially infinitely large arenas, and we are interested in the following decision problem.

Reachability game
INPUT: A reachability game $\mathcal{G} = ((S_0, S_1, \to), \text{Win}_F)$, an initial configuration $s \in S_0 \cup S_1$.
QUESTION: Does player 0 have a winning strategy from s in $\mathcal{G}$?

Parity Games Secondly, we consider min-parity games: the winning condition $\text{Win}_\Omega(S)$ is given by a *priority function* $\Omega : S \to [0, m]$. An infinite path $\pi = s_0 s_1 s_2 \ldots$ is in $\text{Win}_\Omega(S)$ if the smallest priority appearing infinitely often in the sequence is even, i.e. if $\min(\{\Omega(s) |\ s \in Inf(\pi)\})$ is even. A finite play is in $\text{Win}_\Omega(S)$ if its target configuration is in S_1. We simply write Win_Ω in case the set $S = S_0 \uplus S_1$ is obvious from context. To indicate that the winning condition of a game is a parity winning condition, we will speak of *parity games*.

With regards to parity games, we are going to concern ourselves with games over potentially infinitely large arenas, and we are interested in the following decision problem.

Parity game
INPUT: A parity game $\mathcal{G} = ((S_0, S_1, \to), \text{Win}_\Omega)$, an initial configuration $s \in S_0 \cup S_1$.
QUESTION: Does player 0 have a winning strategy from s in $\mathcal{G}$?

Example 1. *Let us consider the arena presented in Figure 1.1. Configurations are partitioned into two sets: the circles, belonging to player* 0*, and the diamonds, belonging to player* 1*. The priorities are* $\{0,1,2,3\}$*. Consider the game on the arena, where the winning condition is a parity condition. Plays* $\pi_1 = q_0q_1q_2(q_4q_7)^\omega$ *and* $\pi_2 = (q_0q_1q_2q_4)^\omega$ *are both winning for player* 0*. Indeed, in* π_1*, the minimal priority encountered infinitely often is* 2*, whereas in* π_2*, it is* 0*. In both cases, it is an even number. On the other hand, the plays* $\pi_3 = (q_5q_3q_6)^\omega$ *and* $\pi_4 = (q_5q_6)^\omega$ *are both winning for player* 1*: the minimal priority encountered is odd, since it is* 1 *for both.*

One can take matters further: on this arena, player 0 *has a winning strategy for the parity game from* q_0*,* q_1*,* q_2*,* q_4 *and* q_7*, but not from* q_5*,* q_3 *nor* q_6*, where player* 1 *has a winning strategy.*

One important property of parity games is that of *memoryless determinacy*: for a parity game $\mathcal{G} = ((S_0, S_1, \to), \text{WIN}_\Omega)$ and an initial configuration $s \in S_0 \cup S_1$, one of the players has a memoryless winning strategy from s [108].

Of note is that several model checking problems can be expressed as decision problems for games: the most famous example is that the modal μ-calculus model checking problem is polynomially equivalent to solving the parity game problem.

1.4 Complexity

We assume the reader is familiar with Turing machines. A proper definition of space bounded deterministic Turing machines is provided in Section 5.2.1, where a more thorough analysis of their behavior is required.

We now provide an overview of time and space complexity classes. We say a Turing machine is $f(n)$*-time bounded* if for any input word w of size n, the length of any computation on w is at most $f(n)$. We say a Turing machine is $f(n)$*-space bounded* if $f(n)$ is the size of its working tape for any computation of the Turing machine on an input of length n.

By P, and EXP we denote the classes of all problems that can be decided by a deterministic Turing machine who is polynomially or exponentially time bounded, respectively and by NP, and NEXP we denote the classes of all problems that can be decided by a non-deterministic Turing machine that is polynomially or exponentially time bounded, respectively.

By L, PSPACE, and $\mathsf{EXPSPACE}$ we denote the classes of all problems that can be decided by a deterministic Turing machine that is logarithmically, polynomially, exponentially space bounded, respectively. We do not explicitly define $\mathsf{NPSPACE}$ and $\mathsf{NEXPSPACE}$ since by Savitch's theorem [90] they are equivalent to PSPACE and $\mathsf{EXPSPACE}$ respectively. We do however define NL as the class of all problems that can be decided by a nondeterministic Turing machine that is logarithmically space bounded.

We define the tower function $T : \mathbb{N} \times \mathbb{R} \to \mathbb{R}$ by $T(0,r) = r$ and $T(h+1,r) = 2^{T(h,r)}$ for all $h \in \mathbb{N}, r \in \mathbb{R}$. Thus $T(h,r)$ is a tower of 2s of height h with an r sitting on top, i.e.

$$T(h,r) = 2^{2^{2^{\cdot^{\cdot^{\cdot^{2^r}}}}}} \Big\} \text{height } h$$

Observe that for all $n, h \in \mathbb{N}$ with $n \geq 1$, we have $T(h, log^{(h)}(n)) = n$. Here a problem is in $\mathsf{ELEMENTARY}$ if there exists $h \in \mathbb{N}$ such that it can be solved in time $O(T(h,0))$. By h-EXP we denote the class of all problems that can be decided by a deterministic Turing machine who is $T(h, f(n))$-time bounded for some polynomial function f. The complexity class h-$\mathsf{EXPSPACE}$ is defined analogously.

Finally, by $\mathsf{PSPACE}^{\mathsf{NEXP}}$ we denote the class of all problems that can be decided by a deterministic Turing machine that is polynomially space bounded, and that has access to results from an oracle in NEXP.

A true/false problem is *decidable* if there exists a time bounded Turing Machine that answers the question asked by the problem. If there exists no such Turing Machine, the problem is *undecidable*.

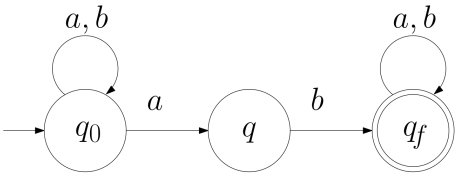

Figure 1.2: A finite automaton that accepts $\{a,b\}^*ab\{a,b\}^*$. The automaton consists of three states, the input alphabet is $\{a,b\}$. The transitions are represented by arrows labeled with the corresponding input symbol. The initial state is q_0 and the set of final state is composed of the singular state q_f.

For a complexity class $C \in \{\mathsf{NP}, \mathsf{PSPACE}, \mathsf{NEXP}, \mathsf{PSPACE}^{\mathsf{NEXP}}, \mathsf{EXPSPACE}, h\text{–}\mathsf{EXP}, h\text{–}\mathsf{EXPSPACE}\}$, we say a problem P is *C-hard* when every problem in C can be reduced in polynomial time to P. For a complexity class $C \in \{\mathsf{NL}, \mathsf{P}\}$ we say a problem P is *C-hard* when every problem in C can be reduced to P using logarithmic space only. When a problem P is both C-hard and in C, we call it *C-complete*.

For more information on Turing machines in the broader definition, we refer the reader to [84, 7], which provide further details on complexity theory as well.

1.5 Automata

A *finite automaton* is a tuple $\mathcal{A} = (Q, \Sigma, R, q_{init}, F)$, where

- Q is a finite *set of states*,
- Σ is a *finite input alphabet*,
- $R \subseteq Q \times \Sigma \times Q$ is a *transition relation*,
- $q_{init} \in Q$ is the *initial state*, and
- $F \subseteq Q$ is a *set of final states*.

A finite automaton $\mathcal{A} = (Q, \Sigma, R, q_{init}, F)$ induces the finite labeled transition system (Q, Σ, R).

The *size* of $\mathcal{A}$ is defined as $|\mathcal{A}| = |Q| + |\Sigma| + |R|$.

A *run* from q_0 to q_n in $\mathcal{A}$ is a path in the labeled transition system (Q, Σ, R) induced by $\mathcal{A}$, and will be noted $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} q_n$, for $a_0, \ldots, a_{n-1} \in \Sigma$. We sometimes use the abbreviation $q \to^* q'$ to denote a run of arbitrary length from q to q'. We say π is *accepting* if $q_0 = q_{init}$ and $q_n \in F$.

We say a word $w \in \Sigma^*$ is accepted by $\mathcal{A}$ if $q_{init} \xrightarrow{w} q$ in (Q, Σ, R) for some $q \in F$. The *language* of $\mathcal{A}$ is $L(\mathcal{A})$ the set of elements accepted by $\mathcal{A}$. We say a language L is *regular* if there exists a finite automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.

Example 2. *See Figure 1.2 for an example of an automaton that accepts the language $\{a,b\}^*ab\{a,b\}^*$.*

We are interested in the following decision problem.

Automata reachability
INPUT: A finite automaton $\mathcal{A}$.
QUESTION: Does there exists an accepting run in $\mathcal{A}$?

We refer the reader to [57] for more details on finite automata and regular languages.

Chapter 2

Models of infinite-state systems

Automata, as defined above, are an example of finite-state machines. In general however transition systems can have an infinite number of configurations. It is possible nonetheless for some transition systems with an infinite number of configurations to have a finite presentation: this is the case for instance for the transition systems induced by a Turing machines. However Rice [88] showed that there is no general algorithm that, given a Turing machine, can determine whether the Turing machine meets any nontrivial semantic specification, where a specification is called *trivial* if it is either true for every Turing machine or false for every Turing machine.

Several other cases of transition systems with an infinite number of states but finite representation exist, with less expressive power, but better decidability properties. In particular, stack-based automata provide a mathematical framework for modeling the sequential behavior of computer programs. They are essentially finite automata that have access to an infinite memory that can be manipulated in a "first-in/last-out" fashion. Pushdown automata are a widely-used formalism with applications in, e.g., inter-procedural control-flow analysis of recursive programs [29, 41] and model checking [14].

A similar example is that of counter automata, that is, automata extended with a set of counters with integer values. One of the earliest results about counter automata was obtained by Minsky who showed that already reachability in counter automata is undecidable even when restricted to two counters only [79]. For this reason, we restrict ourselves to the consideration of one-counter automata. There are other restrictions that lead to decidability for instance flatness (see [28, 76]) or the removal of zero tests, which corresponds to vector addition systems with states (see [78, 68, 89]). Note that a one-counter automaton is essentially a pushdown automaton with only two stack symbols one of which is used as a bottom-of-stack symbol only. In the case of one-counter automata it is a classical result that the reachability problem is NL-complete [102, 71]. When one-counter automata are extended with counter updates encoded in binary the reachability problem becomes NP-complete [53].

Another particular framework of infinite-state systems is that of timed automata. Introduced by Alur and Dill [4] in the 1990's, they too are essentially finite automata that have access to an infinite memory, this time however in the form of clocks that all progress at the same rate, that can be compared against constants and that can be individually reset to zero. While traces of runs in the transition system generated by a finite-state machine only allow for reasoning about the relative order of events, timed automata additionally incorporate timing information between them. They provide a popular formalism to reason about the behavior of real-time systems with desirable algorithmic properties; for instance the reachability problem is decidable and in fact PSPACE-complete [3].

Since we do not concern ourselves with the study of the languages accepted by such systems, but rather with the complexity of decision problems such as reachability games and parity games, we omit the input alphabet both from our considerations and from the automata definitions. We then discuss the introduction of a set of parameters to these three models, whose consequences on the complexity results are going to be studied in more detail in corresponding sections.

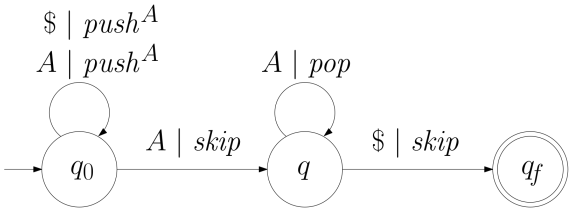

Figure 2.1: A pushdown automaton. The automaton consists of three states, the stack alphabet is $\{\$, A\}$. The rules are represented by arrows labeled with the corresponding pair of topmost stack symbols | stack operation. The initial state is q_0 and the set of final state is composed of the singular state q_f. To reach q_f, the automaton first adds to the stack a certain number of A's in q_0, then removes from the stack the same number of A's while in q.

2.1 Pushdown automata

A pushdown automaton is an automaton extended with a stack that can be manipulated by pushing or popping stack symbols from some stack alphabet. Moreover, the automaton can use the top of the stack to decide which transition to take next.

A *stack over an alphabet* Γ (or *stack content*) is simply a word in Γ^*. We denote the *base set of stack operations* on a stack over Γ as $\mathsf{Op}(\Gamma) = \{push^\gamma \mid \gamma \in \Gamma\} \cup \{pop, skip\}$.

Formally, a *pushdown automaton* (PDA for short) is a tuple $\mathcal{Z} = (Q, \Gamma, R, q_{init}, \gamma_{init}, F)$ where

- Q is a non-empty finite *set of states*,
- Γ is a non-empty finite *stack alphabet*,
- $R \subseteq Q \times \Gamma \times Q \times \mathsf{Op}(\Gamma)$ is finite *set of rules*,
- $q_{init} \in Q$ is the *initial state*,
- $\gamma_{init} \in \Gamma$ is the *initial stack symbol* and
- $F \subseteq Q$ is a *set of final states*.

The *size* of $\mathcal{Z}$ is defined as $|\mathcal{Z}| = |Q| + |\Gamma| + |R|$. Unlike standard notation we write the top of the stack at the rightmost letter of the word. By $\mathsf{Conf}(\mathcal{Z}) = Q \times \Gamma^*$ we denote the set of *configurations* of $\mathcal{Z}$. We rather write $q(w)$ instead of (q, w) to denote elements of $\mathsf{Conf}(\mathcal{Z})$.

A PDA $\mathcal{Z} = (Q, \Gamma, R, q_{init}, \gamma_{init}, F)$ induces a transition system $T_\mathcal{Z} = (\mathsf{Conf}(\mathcal{Z}), \to_\mathcal{Z})$ where for all $q, q' \in Q$, for all $w, w' \in \Gamma^*$, for all $a \in \Gamma$, $q(wa) \to_\mathcal{Z} q'(w')$ if there exists some rule $(q, a, q', op) \in R$ such that either of the following holds

- $op = push^\gamma$ and $w' = wa\gamma$,
- $op = pop$ and $w' = w$, or
- $op = skip$ and $w' = wa$.

A *run* π from $q_0(w_0)$ to $q_n(w_n)$ in $\mathcal{Z}$ is a path $q_0(w_0) \to_\mathcal{Z} q_1(w_1) \to_\mathcal{Z} \cdots \to_\mathcal{Z} q_n(w_n)$ in $T_\mathcal{Z}$. We say π is *accepting* if $q_0(w_0) = q_{init}(\gamma_{init})$ and $q_n \in F$. We refer to Figure 2.1 for an instance of a PDA for which there exists an accepting run.

PDA reachability We are first concerned with the following problem.

PDA REACHABILITY
INPUT: A PDA $\mathcal{Z}$.
QUESTION: Does an accepting run exist in $\mathcal{Z}$?

It is known that the reachability problem for pushdown automata can be solved in polynomial time, as shown by Bouajjani, Esparza, and Maler in [14]. See for instance [91] for more details.

Theorem 3. *[14]* PDA REACHABILITY *is in* P.

PDA games Pushdown automata have also been studied from the point of view of model checking several logics, such as LTL or the modal μ-calculus. As mentioned, the μ-calculus model checking problem is polynomially equivalent to solving the parity game problem, both for finite arenas [36], and also in the case of infinite arena induced by pushdown automata [105]. Hence, we are naturally interested in games on the transition systems generated by pushdown automata.

Given a transition system, one needs only to provide a partition of the set of configurations S into two sets S_0 and S_1 to obtain an arena.

Given a pushdown automaton $\mathcal{Z}$ and a partition of Q into Q_0 and Q_1, we partition the configurations of the transition system $T_{\mathcal{Z}}$ into $\mathsf{Conf}_{\mathcal{Z},0} = Q_0 \times \Gamma^*$ and $\mathsf{Conf}_{\mathcal{Z},1} = Q_1 \times \Gamma^*$.

With these notations in mind one can define the arena

$$A_{(\mathcal{Z},Q_0,Q_1)} = (\mathsf{Conf}_{\mathcal{Z},0}, \mathsf{Conf}_{\mathcal{Z},1}, \rightarrow_{\mathcal{Z}})$$

induced by a PDA $\mathcal{Z}$ and a partition of its set of states.

Similarly, given a priority function $\Omega : Q \to [0, m]$, we naturally extend the function as follows, by setting $\Omega_{\Gamma^*} : \mathsf{Conf}(\mathcal{Z}) \to [0, m]$ and $\Omega_{\Gamma^*}(q, w) = \Omega(q)$ for all $w \in \Gamma^*$.

We recall results for games over pushdown automata's transition systems, namely, reachability and parity games.

PUSHDOWN REACHABILITY GAME
INPUT: A pushdown automata $\mathcal{Z} = (Q, \Gamma, R, q_{init}, \gamma_{init}, F)$, where $Q = Q_0 \uplus Q_1$.
QUESTION: Does player 0 have a winning strategy from $q_{init}(\gamma_{init})$ in the reachability game $\mathcal{G} = (A_{(\mathcal{Z},Q_0,Q_1)}, \text{WIN}_{F \times \Gamma^*})$?

PUSHDOWN PARITY GAME
INPUT: A pushdown automata $\mathcal{Z} = (Q, \Gamma, R, q_{init}, \gamma_{init}, F)$, where $Q = Q_0 \uplus Q_1$, and a priority function $\Omega : Q \to [0, m]$.
QUESTION: Does player 0 have a winning strategy from $q_{init}(\gamma_{init})$ in the parity game $\mathcal{G} = (A_{(\mathcal{Z},Q_0,Q_1)}, \text{WIN}_{\Omega_{\Gamma^*}})$?

For parity games for pushdown automata, it is known from [105] that determining the winner is an EXP-complete problem. An important corollary of this result is that the μ-calculus model checking problem for pushdown automata is EXP-complete. Since the lower bound provided in [105] makes use of a reachability winning condition, the following is known.

Theorem 4. *[105]* PUSHDOWN PARITY GAME *and* PUSHDOWN REACHABILITY GAME *are both* EXP*-complete.*

2.2 One-counter automata

A one-counter automaton is an automaton extended with a counter that can be manipulated by incrementing or decrementing the counter. Moreover, the automaton can compare the value of the counter against zero to decide which transition to take next.

We denote the *base set of counter operations*, as $\mathsf{Op}_\pm \cup \mathsf{Op}_0$, where $\mathsf{Op}_\pm = \{-1, 0, +1\}$ and $\mathsf{Op}_0 = \{= 0, > 0\}$.

A *one-counter automaton* (OCA for short) is a tuple $\mathcal{C} = (Q, R, q_{init}, F)$, where

- Q is a non-empty finite *set of states*,
- $R \subseteq Q \times (\mathsf{Op}_\pm \cup \mathsf{Op}_0) \times Q$ is a finite *set of rules*,
- q_{init} is an *initial state*, and
- $F \subseteq Q$ is a *set of final states*.

The *size* of $\mathcal{C}$ is defined as $|\mathcal{C}| = |Q| + |R|$. By $\mathsf{Conf}(\mathcal{C}) = Q \times \mathbb{Z}$ we denote the set of *configurations* of $\mathcal{C}$. We prefer however to abbreviate a configuration (q, z) by $q(z)$.

Being slightly non-standard we define configurations to take counter values over $\mathbb{Z}$ rather than over $\mathbb{N}$ for notational convenience. This does not cause any loss of generality as we allow guards that enable us to test if the value of the counter is greater or equal to zero.

Previously, such as in [17] or [52], slightly different sets of operations have been used such as operations to increment the counter by a constant represented in binary, i.e. with operations in $\mathsf{Op}_{\pm\mathbb{N}} = \{+c \mid c \in \mathbb{Z}\}$, where $|+c| = \log(|c|)$, or operations to compare the counter against some natural number, i.e. with operations in $\mathsf{Op}_\bowtie = \{\bowtie c \mid \bowtie \in \{<, \leq, =, \geq, >\}, c \in \mathbb{N}\}$, where $|\bowtie c| = \log(|c|)$ for $\bowtie \in \{<, \leq, =, \geq, >\}$. One-counter automata which allow binary updates in $\mathsf{Op}_{\pm\mathbb{N}}$ and tests in Op_0 will be called *Succinct one-counter automata* (SOCA for short). The *size* of a SOCA $\mathcal{C}$ is defined slightly differently as the size of an OCA as $|\mathcal{C}| = |Q| + |R| + \sum_{(q,op,q') \in R} |op|$.

A one-counter automaton $\mathcal{C} = (Q, R, q_{init}, F)$ induces the labeled transition system $T_\mathcal{C} = (\mathsf{Conf}(\mathcal{C}), \Lambda_\mathcal{C}, \rightarrow_\mathcal{C})$ where $\Lambda_\mathcal{C} = \mathsf{Op}$ and where $\rightarrow_\mathcal{C}$ is defined such that for all $q, q' \in Q$, for all $z, z' \in \mathbb{Z}$, for all $op \in \mathsf{Op}$, $q(z) \xrightarrow{op}_\mathcal{C} q'(z')$ if $(q, op, q') \in R$ and either of the following holds

- $op = c \in \mathsf{Op}_\pm$ and $z' = z + c$, or
- $op = \bowtie 0 \in \mathsf{Op}_0$, $z = z'$ and $z \bowtie 0$.

A *run* in $\mathcal{C}$ from $q_0(z_0)$ to $q_n(z_n)$ is a path in $T_\mathcal{C}$, i.e. a sequence, possibly empty (i.e. $n = 0$), of the form

$$\pi \quad = \quad q_0(z_0) \xrightarrow{\pi_0}_\mathcal{C} q_1(z_1) \quad \cdots \quad \xrightarrow{\pi_{n-1}}_\mathcal{C} q_n(z_n).$$

We say π is *accepting* if $q_0 = q_{init}$, $z_0 = 0$, and $q_n \in F$.

Given a run $\pi = q_0(z_0) \xrightarrow{op_0}_\mathcal{C} q_1(z_1) \quad \cdots \quad \xrightarrow{op_{n-1}}_\mathcal{C} q_n(z_n)$, we define the *counter effect* of π as $\Delta(\pi) = z_n - z_0$. We define $\textsc{Values}(\pi) = \{z_i \mid i \in [0, n]\}$ to denote the *set of counter values* of the configurations of π. We define a non-empty run π's *maximum* as $\max(\pi) = \max(\textsc{Values}(\pi))$ and the *minimum* as $\min(\pi) = \min(\textsc{Values}(\pi))$.

As mentioned, OCA can be seen as a particular case of PDA.

Remark 5. *For every OCA $\mathcal{C}$ one can compute in polynomial time a PDA $\mathcal{Z}$ inducing an isomorphic transition system.*

Proof. In order to use its stack to model integers in $\mathbb{Z}$, a pushdown automaton needs only three stack symbols one of which serves as a bottom-of-stack symbol only: +1, −1 and \$. The symbol \$ corresponds to 0, while $\$(+1)^n$ and $\$(-1)^n$ correspond to integers n and $-n$ respectively. The translation is straightforward: $\mathcal{Z}$ uses the same set of states, to any rule $(q, +1, q')$ of $\mathcal{C}$ correspond rules $(q, \$, q', push^{+1}), (q, +1, q', push^{+1})$, and $(q, -1, q', pop)$ in $\mathcal{Z}$, to any rule $(q, -1, q')$ of $\mathcal{C}$ correspond rules $(q, \$, q', push^{-1}), (q, -1, q', push^{-1})$, and $(q, +1, q', pop)$ in $\mathcal{Z}$, to any rule

$(q, +0, q')$ of $\mathcal{C}$ correspond rules $(q, \$, q', \mathit{skip}), (q, -1, q', \mathit{skip})$, and $(q, +1, q', \mathit{skip})$ in $\mathcal{Z}$, rules of the form $(q, = 0, q')$ of $\mathcal{C}$ correspond to rule $(q, \$, q', \mathit{skip})$ in $\mathcal{Z}$ and finally rules of the form $(q, > 0, q')$ of $\mathcal{C}$ correspond to rule $(q, +1, q', \mathit{skip})$ in $\mathcal{Z}$. Thus the construction has polynomial size. □

OCA and SOCA reachability We consider the following decision problem.

OCA REACHABILITY
INPUT: An OCA $\mathcal{C}$.
QUESTION: Does an accepting run exist in $\mathcal{C}$?

We write SOCA REACHABILITY when the input is a succinct one-counter automaton rather than a one-counter automaton.

As OCA can be seen as special cases of PDA, it follows that solving the reachability problem can be done in polynomial time. The problem is actually in NL, see e.g. the works by Lafourcade et al. [71]. NL-hardness then trivially follows from NL-hardness of reachability in directed graphs.

Theorem 6. *[71, 102]* OCA REACHABILITY *is* NL*-complete.*

Succinct one-counter automata, i.e. one-counter automata that allow counter increments and decrements by constants written in binary, and zero tests, have however yielded very different results for this problem.

Theorem 7. *[53]* SOCA REACHABILITY *is* NP*-complete.*

OCA and SOCA games As with pushdown automata, given an OCA $\mathcal{C} = (Q, R, q_{init}, F)$, and a partition of Q into Q_0 and Q_1, we partition the configurations of the transition system $T_{\mathcal{C}}$ into $\mathsf{Conf}_{\mathcal{C},0} = Q_0 \times \mathbb{Z}$ and $\mathsf{Conf}_{\mathcal{C},0} = Q_0 \times \mathbb{Z}$.

With these notations in mind one can define the arena

$$A_{(\mathcal{C},Q_0,Q_1)} = (\mathsf{Conf}_{\mathcal{C},0}, \mathsf{Conf}_{\mathcal{C},1}, E_{\mathcal{C}})$$

induced by a OCA $\mathcal{C}$ and a partition of its set of states, where $E_{\mathcal{C}}$ is the binary relation such that for all $q(z), q'(z') \in \mathsf{Conf}(\mathcal{C})$, $(q(z), q'(z')) \in E_{\mathcal{C}}$ iff $q(z) \xrightarrow{op}_{\mathcal{C}} q'(z')$ for some $op \in \mathsf{Op}_{\pm} \cup \mathsf{Op}_0$. The construction of an arena for a SOCA follows the same pattern.

Again, given a priority function $\Omega : Q \to [0, m]$, we naturally extend the function as follows, by setting $\Omega_{\mathbb{Z}} : \mathsf{Conf}(\mathcal{C}) \to [0, m]$ and $\Omega_{\mathbb{Z}}(q, z) = \Omega(q)$ for all $z \in \mathbb{Z}$.

We recall results for games over OCA and SOCA transition systems, namely, reachability and parity games.

OCA REACHABILITY GAME
INPUT: A OCA $\mathcal{C} = (Q, R, q_{init}, F)$, where $Q = Q_0 \uplus Q_1$.
QUESTION: Does player 0 have a winning strategy from $q_{init}(0)$ in the reachability game $\mathcal{G} = (A_{(\mathcal{C},Q_0,Q_1)}, \mathrm{WIN}_{F \times \mathbb{Z}})$?

OCA PARITY GAME
INPUT: A OCA $\mathcal{C} = (Q, R, q_{init}, F)$, where $Q = Q_0 \uplus Q_1$, and a priority function $\Omega : Q \to [0, m]$.
QUESTION: Does player 0 have a winning strategy from $q_{init}(0)$ in the parity game $\mathcal{G} = (A_{(\mathcal{C},Q_0,Q_1)}, \mathrm{WIN}_{\Omega_{\mathbb{Z}}})$?

As before we write SOCA REACHABILITY GAME and SOCA PARITY GAME when the input is a SOCA rather than a OCA.

PSPACE-hardness of the OCA REACHABILITY GAME problem was proved in 1995 by Holzer [60]. A more self-contained proof was later provided by Jančar and Sawa in [65]. As OCA can be seen as special cases of PDA, it follows that deciding the winner in a parity game played on the transition graph of an OCA can be achieved in EXP. This upper bound has been improved by Serre [95], who provided a matching PSPACE upper bound.

Theorem 8. *[95, 65]* OCA REACHABILITY GAME *and* OCA PARITY GAME *are both* PSPACE*-complete.*

This problem is again more difficult to solve in the case of SOCA.

Theorem 9. *[63]* SOCA REACHABILITY GAME *and* SOCA PARITY GAME *are both* EXPSPACE*-complete.*

2.2.1 Bounded one-counter automata

Note that even without syntactic $\mathsf{Op}_{\bowtie}$ test, one can still perform "$> c$" tests by using a gadget consisting of a $-c$, followed by a > 0 test, followed by a $+c$ and, similarly, one can still perform "$= c$" tests by using a gadget consisting of a $-c$, followed by a $= 0$ test, followed by a $+c$. However note that "$< c$" tests cannot be performed in a similar way. In order to enforce upper bounds on the counter values, bounded counter automata have been introduced. A bounded one-counter automaton has a single counter that can store values between 0 and some bound $b > 0$. The automaton may increase or decrease the counter by constants written in binary as long as this doesn't make the counter strictly larger than b or strictly smaller than 0. It may also compare the counter against constants written in binary.

A *bounded one-counter automaton* (BOCA for short) is a tuple $\mathcal{C} = (Q, R, b, q_{init}, F)$, where

- Q is a non-empty finite *set of states*,
- $R \subseteq Q \times (\mathsf{Op}_{\pm\mathbb{N}} \cup \mathsf{Op}_{\bowtie}) \times Q$ is a finite *set of rules*,
- $b \in \mathbb{N} \setminus \{0\}$ is a *bound*,
- q_{init} is an *initial state*, and
- $F \subseteq Q$ is a *set of final states*.

The *size* of $\mathcal{C}$ is defined as $|\mathcal{C}| = |Q| + |R| + log(b) + \sum_{(q,op,q')\in R} |op|$. *Configurations* of a bounded counter automata belong to $Q \times [0, b]$. Let $\mathsf{Consts}(\mathcal{C})$ denote the constants that appear in the operations $op \in \mathsf{Op}_{\bowtie}$ for some rule (q, op, q') in R. *Accepting runs* are defined analogously as for OCA.

BOCA reachability We consider the following problem.

BOCA REACHABILITY
INPUT: A BOCA $\mathcal{C}$.
QUESTION: Does an accepting run exist in $\mathcal{C}$?

It was shown in [42] that the reachability problem for bounded one-counter automata is PSPACE-complete.

Theorem 10. *[42]* BOCA REACHABILITY *is* PSPACE*-complete.*

The presence of "$< c$" tests is of interest to us since they serve to provide polynomial inter-reducibility between reachability in timed automata and BOCA as established in [54].

2.3 Timed automata

A *guard* over a finite set of clocks Ω is a comparison of the form $\omega \bowtie c$, where $\omega \in \Omega$, $c \in \mathbb{N}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$. We denote by $\textsc{Guards}(\Omega)$ the *set of guards* over the set of clocks Ω. The *size* of a guard $g = \omega \bowtie c$ is defined as $|g| = \log(c)$. A *clock valuation* is a function from Ω to $\mathbb{N}$; we write $\vec{0}$ to denote the clock valuation $\omega \mapsto 0$ whenever the set Ω is clear from the context. For each clock

valuation v and each $t \in \mathbb{N}$ we denote by $v+t$ the clock valuation $\omega \mapsto v(\omega)+t$. For each guard $g = \omega \bowtie c$ with $c \in \mathbb{N}$, we write $v \vDash g$ if $v(\omega) \bowtie c$.

A timed automaton is a finite automaton extended with a finite set of clocks Ω that all progress at the same rate and that can individually be reset to zero. Moreover, every transition is labeled by a guard over Ω and by a set of clocks to be reset.

Formally, a *timed automaton* (*TA* for short) is a tuple $\mathcal{A} = (Q, \Omega, R, q_{init}, F)$, where

- Q is a non-empty finite *set of states*,
- Ω is a non-empty finite *set of clocks*,
- $R \subseteq Q \times \mathcal{G}(\Omega) \times \mathscr{P}(\Omega) \times Q$ is a finite *set of rules*,
- $q_{init} \in Q$ is an *initial state*, and
- $F \subseteq Q$ is a *set of final states*.

We also refer to $\mathcal{A}$ as an n-TA if $|\Omega| = n$. The *size* of $\mathcal{A}$ is defined as

$$|\mathcal{A}| \;=\; |Q| + |\Omega| + |R| + \sum_{(q,g,U,q') \in R} |g|.$$

Let $\mathsf{Consts}(\mathcal{A}) = \{c \in \mathbb{N} \mid \exists (q,g,U,q') \in R,\ \exists \omega \in \Omega, \bowtie \in \{<, \leq, =, \geq, >\} : g = \omega \bowtie c\}$ denote the set of constants that appear in the guards of the rules of $\mathcal{A}$.

By $\mathsf{Conf}(\mathcal{A}) = Q \times \mathbb{N}^{\Omega}$ we denote the set of *configurations* of $\mathcal{A}$. We prefer however to abbreviate a configuration (q, v) by $q(v)$.

A TA $\mathcal{A} = (Q, \Omega, R, q_{init}, F)$ induces the labeled transition system $T_{\mathcal{A}} = (\mathsf{Conf}(\mathcal{A}), \Lambda_{\mathcal{A}}, \rightarrow_{\mathcal{A}})$ where $\Lambda_{\mathcal{A}} = R \times \mathbb{N}$ and where $\rightarrow_{\mathcal{A}}$ is defined such that, for all $(\delta, t) \in R \times \mathbb{N}$ with $\delta = (q, g, U, q') \in R$, for all $q(v), q'(v') \in \mathsf{Conf}(\mathcal{A})$, $q(v) \xrightarrow{\delta,t}_{\mathcal{A}} q'(v')$ if $v+t \vDash g$, $v'(u) = 0$ for all $u \in U$ and $v'(\omega) = v(\omega)+t$ for all $\omega \in \Omega \setminus U$.

A *run* from $q_0(v_0)$ to $q_n(v_n)$ in $\mathcal{A}$ is a path in the transition system $T_{\mathcal{A}}$, that is, a sequence $\pi = q_0(v_0) \xrightarrow{\delta_1,t_1}_{\mathcal{A}} q_1(v_1) \cdots \xrightarrow{\delta_n,t_n}_{\mathcal{A}} q_n(v_n)$; it is called *reset-free* if for all $i \in \{1, \ldots, n\}$, $\delta_i = (g_i, \varnothing)$ for some guard g_i.

We say π is *accepting* if $q_0(v_0) = q_{init}(\vec{0})$ and $q_n \in F$.

It is worth mentioning that there are further modes of time valuations and guards which exist in the literature, we refer to [6] for a recent overview. Notably, we consider in this thesis only the case of timed automata over discrete time. It is worth mentioning that in the case of timed automata over continuous time (i.e. with clocks having values in $\mathbb{R}_{\geq 0}$), techniques [58, 83] exist for reducing the reachability problem to discrete time in the case of closed (i.e. non-strict) clock constraints ranging over integers.

TA reachability We consider the following problems.

n-TA REACHABILITY
INPUT: An n-TA $\mathcal{A}$.
QUESTION: Does an accepting run exist in $\mathcal{A}$?

TA REACHABILITY
INPUT: A TA $\mathcal{A}$.
QUESTION: Does an accepting run exist in $\mathcal{A}$?

Using a technique called *region abstraction*, Alur and Dill proved the following theorem.

Theorem 11. *[4]* TA REACHABILITY *is* PSPACE*-complete.*

This result was later refined by Courcoubetis and Yannakakis who showed that PSPACE-hardness already holds if $\mathcal{A}$ is a 3-TA.

Theorem 12. *[30]* 3-TA REACHABILITY *is* PSPACE*-complete.*

The cases with less than three clocks were left out in [30] and later discussed by Laroussinie, Markey, and Schnoebelen, who showed that reachability in one-clock timed automata is NL-complete and NP-hard for two clocks [72].

Theorem 13. *[72]* 1-TA REACHABILITY *is* NL*-complete.*

The gap in the case of two clocks was left open for some time before Fearnley and Jurdziński showed that reachability in two-clock timed automata is PSPACE-complete [42].

Theorem 14. *[42]* 2-TA REACHABILITY *is* PSPACE*-complete.*

Timed automata have been tied to the study of automata extended with counters. For instance, the undecidability proof of the universality problem for nondeterministic timed automata by Alur and Dill [4] proceeds via a reduction from reachability in two-counter automata. On a different note, it has been shown that BOCA reachability is polynomial time inter-reducible to reachability in 2-TA automata [54].

Theorem 15. *[54]* 2-TA REACHABILITY *is polynomial time inter-reducible with* BOCA REACHABILITY.

Chapter 3

Parametric systems

The infinite-state systems defined in the precedent chapter are widely used to model the behavior of computer programs. Essentially these approaches however address only concrete specifications. For instance, one can ask, given a program, whether a certain set of states is reachable without some counter exceeding a threshold value 2048, or one can ask whether a certain set of states is reachable after having more than 10 time units elapse. For certain types of computer programs however, such as embedded systems, the constraints depend on some environment, and concrete constraints are useful only when restricting oneself to a given concrete environment. In real life, it thus makes sense to rather ask, for instance, whether there exists a threshold value t such that a certain set of states is reachable without some counter exceeding t, or to ask whether for all threshold t a certain set of states is reachable after having more than t time units elapse. Hence the need to study under-specified systems, and parametric constraints.

In order to understand the behavior of such under-specified systems, Alur, Henzinger and Vardi have introduced parametric timed automata in their seminal paper [5]. In addition to expressing concrete timing constraints, these can employ parametric constraints. Verification of the desired behavior of the system is then performed without concrete values. Their model laid the foundations for parametric reasoning about real time. With similar considerations, it has also been natural to consider extensions of one-counter automata allowing updates that increase or decrease the counter by integer parameters [64, 15].

In this thesis, we consider the problem of analysing parametric pushdown, one-counter and timed automata, i.e. automata extended with a stack, counter or clocks that can be compared with parameters. These parameters can take unspecified values over infinite domains, which here consist in the set of words (for parametric pushdown automata) or the non-negative integers (for parametric one-counter automata and parametric timed automata). An M-*parameter valuation* is then a function μ from a set P of parameters to a given set M. We will omit the name of the set M in case it is obvious from context. Our interests lies in the parametric variants of requirements such as safety or liveness properties: for instance, finding constraints on the parameters defining the set of all possible values for which the system satisfies a safety or liveness property, or verifying that the model satisfies some safety or liveness property for all possible values of the parameters.

3.1 Parametric reachability

The *parametric reachability problem* for an automaton with parameters (e.g. for a PTA) asks whether there exists a parameter valuation μ such that a final configuration of the resulting concrete automaton becomes reachable. From here onwards, we omit "parametric" and write simply "reachability problem" when it is clear from context that the input automaton has parameters. Solving such a problem can be useful to verify safety properties of automaton with parameters, i.e. to answer the question "is a bad state reachable for some valuation of the parameters"?

We are interested in how adding a set of parameters influences decidability and complexity

properties. In particular, notice that the effects of adding a set of parameters depend on the model. Indeed, the reachability problem for parametric timed automata is long known to be undecidable [5], whereas the reachability problem for timed automata is PSPACE-complete [4]. On the other hand, reachability for succinct one-counter automata is NP-complete [53] and reachability for parametric one-counter automata is both NP-hard and in NEXP [52].

3.2 Parametric games

As noted precedently, several model checking problems can be expressed as decision problems for games. For instance as mentioned in the case of pushdown automata, the modal μ-calculus model checking problem is polynomially equivalent to solving the parity game problem. Since we are interested in parametric model checking and specification, we are naturally interested in parametric variants of reachability games and parity games.

One goal may be to characterize the set of all possible parameter valuations for which a player has a winning strategy in a reachability game or a parity game played on the arena induced by the resulting concrete automaton and some partition of its set of states. For determining such a set of parameter valuations, we incorporate the choice of the parameter valuation to the game. Each player then wants the parameter valuation to lead to an arena where it has a winning strategy.

In keeping with the spirit of alternation between players choosing successors in the arena, we consider here games where players alternate choosing values for the parameters before alternating choosing successor configurations in the game on the induced arena. For simplicity's sake, the values are assigned to the parameters one by one, alternating between values assigned by player 0 and by player 1.

A winning strategy for player 0 allows to synthesize a controller that restricts the environment and ensures that the property expressed by the winning condition always holds.

Parameter valuation arenas We introduce here arenas over the set of partial M-parameter valuations for a set of parameters $P = P_0 \uplus P_1$. The values are assigned to the parameters one by one, alternating between values assigned by player 0 (corresponding to parameters in P_0) and by player 1 (corresponding to parameters in P_1). Other options could have been considered: for instance, we could consider a fully existential approach (i.e. all parameters have their values assigned by player 0), or a fully universal approach (i.e. all parameters have their values assigned by player 1). Note however that the approach considered here encompasses both of the latter approaches: the fully existential approach consists in the case where $P_1 = \varnothing$ and the fully universal one, the case $P_0 = \varnothing$.

Assume an ordering of the parameters $P_0 \uplus P_1 = \{p_0, p_1, \dots, p_k\}$ for some $k \in \mathbb{N}$. Then given a set M, a *parameter valuation arena* is a tuple $A_{P_0,P_1,\mathsf{M}} = (S_0, S_1, \to_P)$ where

- S_0 is the set of all partial M-parameter valuations with domain $\{p_0, p_1, \dots, p_j\} \setminus \{p_j\}$ where $\mathsf{p}_j \in P_0$,
- S_1 is the set of all partial M-parameter valuations with domain $\{p_0, p_1, \dots, p_j\} \setminus \{p_j\}$ where $\mathsf{p}_j \in P_1$,
- $\mu \to_P \mu'$, if $\mu' \in (\mathsf{M})^{\{p_0,\dots,p_j\}}$ for some $j \in [0,k]$, $\mathrm{Dom}(\mu) = \mathrm{Dom}(\mu') \setminus \{p_j\}$ and $\mu'(p) = \mu(p)$ for $p \in \mathrm{Dom}(\mu)$.

Note that the arena has dead ends, and that these are the configurations that correspond to parameter valuations. Given a strategy σ_0 for player 0 and a strategy σ_1 for player 1, the last configuration of the resulting play $\pi(\mu_\perp, \sigma_0, \sigma_1)$ is called *the resulting parameter valuation* and is denoted by μ_{σ_0,σ_1}, where $\mu_\perp$ is the partial parameter valuation with domain $\varnothing$.

Parametric valuation arenas are used as prefixes in parametric games. Once the resulting parameter valuation is produced by the two players, it serves to instantiate the transition system for continuing the game. Indeed, where a concrete automaton induces a transition system, a parametric

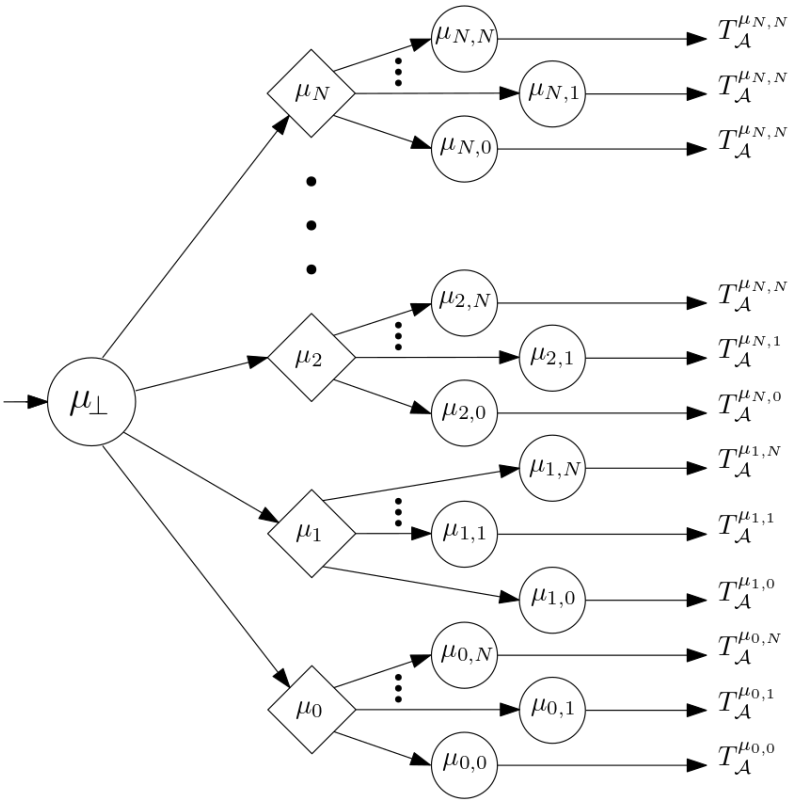

Figure 3.1: An illustration of the arena for a parametric automaton $\mathcal{A}$ with $P_0 = \{p_0\}$, $P_1 = \{p_1\}$ and $\mathsf{M} = [0, N]$ for some $N \in \mathbb{N}$. Here $\mu_\perp$ is the totally undefined function and can be viewed rather as the function that assigns $\perp$ to both parameters. For $i, j \in [0, N]$, μ_i is the partial parameter assignment with domain $\{p_0\}$ that maps p_0 to i and $\mu_{i,j}$ is the parameter assignment that maps p_0 to i and p_1 to j. As before, the circles belong to player 0 and the diamonds to player 1.

automaton induces one transition system for every possible parameter valuation. We use $T_{\mathcal{A}}^{\mu}$ to denote the transition system induced by the concrete automaton corresponding to a parametric automaton $\mathcal{A}$ with parameters P taking values in M and a parameter valuation $\mu : P \to \mathsf{M}$.

Given a parametric automaton $\mathcal{A}$ with a set of parameters $P = P_0 \uplus P_1$ that can take values in the set M, a *parametric game* for $\mathcal{A}$ is a game played on the arena $A_{\mathcal{A}}$ that include both the parameter valuation arena $A_{P_0,P_1,\mathsf{M}}$ and, for all parameter valuations $\mu : P \to \mathsf{M}$, the transition system $T_{\mathcal{A}}^{\mu}$ — albeit with configurations additionally indexed by μ to avoid confusion. Moreover, for every dead end μ in $A_{P_0,P_1,\mathsf{M}}$, we add a transition from the configuration $\mu \in \mathsf{M}^P$ to $T_{\mathcal{A}}^{\mu}$. See Figure 3.1 for an illustration of such an arena. Solving parametric games on an automaton with parameters (e.g. a POCA) then consists in answering whether there exists a winning strategy for player 0 from the initial configuration of the parametric game. Remark that the parametric reachability problem for an automaton with parameters can be seen as solving the parametric reachability game where $P_1 = \varnothing$, and $Q_1 = \varnothing$.

Chapter 4

Parametric one-counter automata

This chapter studies the computational complexity of reachability in classes of parametric one-counter automata. Parametric one-counter automata (POCA for short) provide a formalism to reason about programs with behaviors that make use of parametric constraints or updates. They extend one-counter automata with the ability to increment or decrement the counter by parameters that can take unspecified non-negative integer values. POCA are used in various synthesis problems, and to model open programs, whose behavior depends on values input from the environment [5], or to model resources like memory or time being consumed by transitions [107]. The *reachability problem for POCA* in turn asks for the existence of an assignment of the parameters to the non-negative integers such that an accepting run exists in the resulting one-counter automaton.

The reachability problem for POCA is known to be NP-hard and in NEXP [52]. More recently however, new extensions of one-counter automata by parameters have arisen, motivated notably by the study of parametric timed automata and by the relationship between bounded one-counter automata and timed automata. Bundala and Ouaknine [17] introduced a new model which extends bounded one-counter automata by relaxing the assumption that the counter values remain in a bounded interval, by allowing updates and comparisons to be parametric as well, and by allowing modulo tests. Since the counter values in this new model can be unbounded, but can be tested against certain thresholds, we will call this variant parametric threshold one-counter automata (PTOCA for short).

We provide formal definitions of parametric threshold one-counter automata in Section 4.1. We give an overview of our contribution in Section 4.1.1. The contribution consists in a series of techniques to partition a fictitious run into several carefully chosen subruns and manipulate them to obtain a new run with parameter value of lesser magnitude. We provide the proof of our result, which stretches along Section 4.4, Section 4.5, Section 4.6, and Section 21. In Section 4.8 we close the chapter with a discussion about the methods used, with some directions for future work.

4.1 Definitions

Given a set of parameters P we denote by $\mathsf{Op}(P)$ the *set of operations* over the set of parameters P, being of the form $\mathsf{Op}(P) = \mathsf{Op}_{\pm} \cup \mathsf{Op}_{\pm P} \cup \mathsf{Op}_{\mathsf{mod}} \cup \mathsf{Op}_{\bowtie} \cup \mathsf{Op}_{\bowtie P}$, where

- $\mathsf{Op}_{\pm} = \{-1, 0, +1\}$,
- $\mathsf{Op}_{\pm P} = \{+p, -p \mid p \in P\}$,
- $\mathsf{Op}_{\mathsf{mod}} = \{\mathsf{mod}\ c \mid c \in \mathbb{N}\}$,
- $\mathsf{Op}_{\bowtie} = \{\bowtie c \mid \bowtie \in \{<, \leq, =, \geq, >\}, c \in \mathbb{N}\}$, and
- $\mathsf{Op}_{\bowtie P} = \{\bowtie p \mid \bowtie \in \{<, \leq, =, \geq, >\}, p \in P\}$.

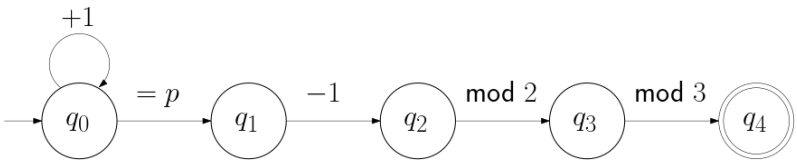

Figure 4.1: An example of a PTOCA. The automaton consists of five states and the set of parameters is $\{p\}$. The rules are represented by arrows labeled with the corresponding operations. A parameter valuation $\mu : \{p\} \to \mathbb{N}$ witnesses that an accepting μ-run exists in the above PTOCA if, and only, if $\mu(p) \equiv 1 \bmod 6$.

The *size* $|op|$ of an operation op is defined as

$$|op| = \begin{cases} \log(c) & \text{if } op = \mathsf{mod}\ c \text{ or } op = \bowtie c \text{ with } c \in \mathbb{N}, \\ 1 & \text{otherwise.} \end{cases}$$

We denote by *updates* those operations that lie in $\mathsf{Op}_{\pm} \cup \mathsf{Op}_{\pm P}$ and by *tests* those operations that lie in $\mathsf{Op}_{\mathsf{mod}} \cup \mathsf{Op}_{\bowtie} \cup \mathsf{Op}_{\bowtie P}$.

A *parametric threshold one-counter automaton* (PTOCA for short) is a tuple

$$\mathcal{C} = (Q, P, R, q_{init}, F),$$

where

- Q is a non-empty finite *set of states*,
- P is a non-empty finite *set of parameters* that can take non-negative integer values,
- $R \subseteq Q \times \mathsf{Op}(P) \times Q$ is a finite *set of rules*,
- q_{init} is an *initial state*, and
- $F \subseteq Q$ is a *set of final states*.

We refer to $\mathcal{C}$ as an n-PTOCA if $n = |P|$ is the number of parameters of $\mathcal{C}$. The *size* of $\mathcal{C}$ is defined as

$$|\mathcal{C}| \;=\; |Q| + |P| + |R| + \sum_{(q,op,q') \in R} |op|.$$

Again let $\mathsf{Consts}(\mathcal{C})$ denote the constants that appear in the operations $op \in \mathsf{Op}_{\mathsf{mod}} \cup \mathsf{Op}_{\bowtie}$ for some rule (q, op, q') in R. As usual, by $\mathsf{Conf}(\mathcal{C}) = Q \times \mathbb{Z}$ we denote the set of *configurations* of $\mathcal{C}$, and prefer to abbreviate a configuration (q, z) by $q(z)$.

Being slightly non-standard we define configurations to take counter values over $\mathbb{Z}$ rather than over $\mathbb{N}$ for notational convenience. This does not cause any loss of generality as we allow guards that enable us to test if the value of the counter is greater or equal to zero.

Previously, such as in [17] or [52], slightly different sets of operations have been used, such as operations to increment the counter by a constant represented in binary, and tests of the form $\{=0, >0\}$, but not tests in $\mathsf{Op}_{\mathsf{mod}}$ or $\mathsf{Op}_{\bowtie}$. We rather refer to the models using these, i.e. with

$$\mathsf{Op}'(P) = \mathsf{Op}_{\pm\mathbb{N}} \cup \mathsf{Op}_{\pm P} \cup \mathsf{Op}_0,$$

as *Parametric one-counter automata* (POCA for short). Moreover, Bundala and Ouaknine [17] extend PTOCA with some operations of the form $+[0,p]$ that allow to nondeterministically add to the counter a value that lies in $[0,\mu(p)]$, where $\mu(p)$ is the parameter valuation of some parameter p. They do so in order to provide an exponential time reduction from the reachability problem of parametric timed automata with two parametric clocks to the reachability problem of parametric one counter automata. We shall later show in Section 5.3 that, when reducing the reachability problem for parametric timed automata with two parametric clocks and only one parameter, one does not require these $+[0,p]$-transitions nor binary increments.

A parametric threshold one-counter automaton $\mathcal{C} = (Q, P, R, q_{init}, F)$, and a parameter valuation $\mu : P \to \mathbb{N}$ induces the labeled transition system $T^{\mu}_{\mathcal{C}} = (\mathsf{Conf}(\mathcal{C}), \lambda_{\mathcal{C}}, \to_{\mathcal{C},\mu})$, where $\lambda_{\mathcal{C}} = \mathsf{Op}(P)$ and where $\to_{\mathcal{C},\mu}$ is defined such that for all $q(z), q'(z') \in \mathsf{Conf}(\mathcal{C})$, for all $op \in \mathsf{Op}(P)$, $q(z) \xrightarrow{op}_{\mathcal{C},\mu} q'(z')$ if there exists some $(q, op, q') \in R$ such that either of the following holds

(1) $op = c \in \mathsf{Op}_{\pm}$ and $z' = z + c$,

(2) $op \in \mathsf{Op}_{\pm P}$, and either

- $op = +p$ and $z' = z + \mu(p)$, or
- $op = -p$ and $z' = z - \mu(p)$,

(3) $op = \mathsf{mod}\ c \in \mathsf{Op}_{\mathsf{mod}}$, $z = z'$ and $z' \equiv 0 \mod c$,

(4) $op = \bowtie c \in \mathsf{Op}_{\bowtie}$, $z = z'$ and $z' \bowtie c$, and

(5) $op = \bowtie p \in \mathsf{Op}_{\bowtie P}$, $z = z'$ and $z' \bowtie \mu(p)$.

Let $\mu : P \to \mathbb{N}$ be a parameter valuation. A *μ-run* in $\mathcal{C}$ (from $q_0(z_0)$ to $q_n(z_n)$) is a path in $T^{\mu}_{\mathcal{C}}$, that is, a sequence, possibly empty (i.e. $n = 0$), of the form

$$\pi \quad = \quad q_0(z_0) \xrightarrow{\pi_0}_{\mathcal{C},\mu} q_1(z_1) \quad \cdots \quad \xrightarrow{\pi_{n-1}}_{\mathcal{C},\mu} q_n(z_n).$$

As with OCA, we say π is *accepting* if $q_0 = q_{init}$, $z_0 = 0$, and $q_n \in F$. We refer to Figure 4.1 and Figure 4.2 for instances of PTOCA for which there exists an accepting μ-run for some $\mu \in \mathbb{N}^P$. For any two $i, j \in [0,n]$ we naturally define the *subrun* $\pi[i,j]$ *from* $q_i(z_i)$ *to* $q_j(z_j)$ as the μ-run

$$q_i(z_i) \xrightarrow{\pi_i}_{\mathcal{C},\mu} q_{i+1}(z_{i+1}) \quad \cdots \quad \xrightarrow{\pi_{j-1}}_{\mathcal{C},\mu} q_j(z_j).$$

As expected, a *prefix* (resp. *suffix*) of π is a μ-run of the form $\pi[0,j]$ (resp. $\pi[i,n]$).

We define $\Delta(\pi) = z_n - z_0$ as the *counter effect* of the run π and for each $i \in [0, n-1]$ let $\Delta(\pi, i) = \Delta(\pi[i, i+1])$ denote the counter effect of the i-th transition of π.

In the particular case where $P = \{p\}$ is a singleton for some parameter p and $\mu(p) = N$, we prefer to write $q(z) \xrightarrow{op}_{\mathcal{C},N} q'(z')$ to denote $q(z) \xrightarrow{op}_{\mathcal{C},\mu} q'(z')$ and prefer to call a μ-run an *N-run*. In case the automaton $\mathcal{C}$ is obvious from context, we write $\to_{\mu}$ (resp. $\to_N$) instead of $\to_{\mathcal{C},\mu}$ (resp. $\to_{\mathcal{C},N}$).

Given a μ-run $\pi = q_0(z_0) \xrightarrow{\pi_0}_{\mathcal{C},\mu} q_1(z_1) \quad \cdots \quad \xrightarrow{\pi_{n-1}}_{\mathcal{C},\mu} q_n(z_n)$, we define $\textsc{Values}(\pi) = \{z_i \mid i \in [0,n]\}$ to denote the *set of counter values* of the configurations of π. We define a run π's *maximum* as $\max(\pi) = \max(\textsc{Values}(\pi))$ and the *minimum* as $\min(\pi) = \min(\textsc{Values}(\pi))$.

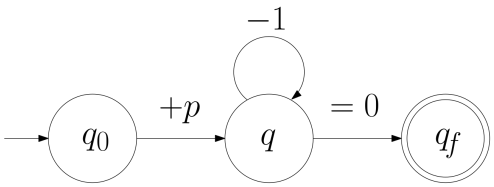

Figure 4.2: Another example of a PTOCA. Remark that for any instantiation of the parameter p with a natural number N, an accepting N-run has length $N + 2$ and is thus not of size bounded by the size of the automaton.

We are interested in the following decision problem.

n-PTOCA REACHABILITY
INPUT: An n-PTOCA $\mathcal{C}$.
QUESTION: Does an accepting μ-run for some $\mu \in \mathbb{N}^P$ exist in $\mathcal{C}$?

We just write PTOCA REACHABILITY when $n = |P|$ is a priori not fixed. Similarly, we write POCA REACHABILITY when the input is a parametric one-counter automaton rather than a parametric threshold one-counter automaton.

Most results have been concerned with parametric one-counter automata. In [52], Haase provides a polynomial time reduction from POCA REACHABILITY to satisfiability in quantifier-free Presburger arithmetic with divisibility, while believing the latter to be in NP. To the author's knowledge, the best upper bound currently known for satisfiability in quantifier-free Presburger arithmetic with divisibility is NEXP [75]. For a more detailed discussion on the misplaced folklore belief that existential Presburger arithmetic with divisibility is NP-complete, we refer to [73].

Theorem 16. *[52, 75]* POCA REACHABILITY *is* NP*-hard and in* NEXP*.*

Note that even without syntactic parametric test, one can still perform $> p$ tests by using a gadget consisting of a $-p$, followed by a > 0 test, and, similarly, one can still perform $= p$ tests by using a gadget consisting of a $-p$, followed by a $= 0$ test. However note that $< p$ tests cannot be performed in this manner. It was shown in [42] that allowing $\leq c$ tests in SOCA makes the reachability problem PSPACE-complete.

A natural question, since the complexity for POCA REACHABILITY is so close to that of SOCA REACHABILITY, would be to ask whether or not reachability in POCA allowing $\leq c$ and $\leq p$ tests is PSPACE-complete as well.

Note that our PTOCA model englobes POCA allowing $\leq c$ and $\leq p$ tests, since parametric updates can be used to simulate updates by constants written in binary: one simply needs to have as many additional parameters as there are constants one wishes to use in binary updates. Then it's possible to check that these parameters' valuations are equal to corresponding constants (since we allow $= c$ tests) and these parametric updates then correspond to updates by constants written in binary. This in particular means that PSPACE-hardness follows from the result from [42].

Theorem 17. PTOCA REACHABILITY *is* PSPACE*-hard.*

On the positive side, Bundala and Ouaknine showed decidability in the case of 1 parameter via a reduction to existential Presburger arithmetic with divisibility (∃PAD for short). Since our PTOCA model is a subset of the one introduced in [17] the following upper bound follows from theirs.

Theorem 18 (Theorem 10.18 in [17])**.** 1-PTOCA REACHABILITY *is decidable.*

Bundala and Ouaknine first prove decidability for reachability in 1-PTOCA with VALUES$(\pi) \subseteq [0, 2 \cdot N]$ for all accepting N-runs for all $N \in \mathbb{N}$, then relax the restriction. Similarly, we are interested

in 1-PTOCA that adhere to a generalization of this restriction. For all $h \in \mathbb{N}$, an *h-bounded* 1-PTOCA is a 1-PTOCA $\mathcal{C}$ such that for all $N \in \mathbb{N}$, all N-runs π in $\mathcal{C}$ satisfy $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$. We define the reachability problem for this subset as the following problem.

h-BOUNDED 1-PTOCA REACHABILITY
INPUT: An h-bounded 1-PTOCA $\mathcal{C}$.
QUESTION: Does an accepting N-run for some $N \in \mathbb{N}$ exist in $\mathcal{C}$?

4.1.1 Contribution

This chapter's main contribution is the Small Parameter Theorem (Theorem 21), which tells us that for every PTOCA over one parameter, for every constant $h \in \mathbb{N}$ and every sufficiently large parameter value N, accepting N-runs with counter values all in $[0, h \cdot N]$ can be turned into accepting N'-runs for some smaller N'. The theorem relies on a series of techniques in order to partition a fictitious N-run with counter values at most $h \cdot N$ into several carefully chosen subruns and in turn allow us to prove that it is sufficient to consider a parameter value of lesser magnitude. In the case of h-bounded 1-PTOCA, a repeated application of the Small Parameter Theorem (Theorem 21) allows us to conclude that an accepting N-run all of whose counter values lie in $[0, h \cdot N]$ exists for some $N \in \mathbb{N}$ if, and only if, there exists an accepting N'-run for some N' that is at most exponential in h and the size of the PTOCA. This exponential upper bound on the parameter value leads to a PSPACE upper bound for h-BOUNDED 1-PTOCA REACHABILITY.

Theorem 19. *h-BOUNDED 1-PTOCA REACHABILITY is in* PSPACE.

We hope that our techniques can be extended to work on 1-PTOCA and, furthermore, on PTOCA with more than one parameter.

4.1.2 Overview

In order to prove the Small Parameter Theorem, we first introduce the notion of semiruns and give several techniques for manipulating them in Section 4.4. We then introduce several lemmas and prove the Small Parameter Theorem (Theorem 21) by carefully factorizing a potential N-run into subsemiruns that can be treated by the lemmas. We refer the reader to Section 4.3 for a thorough overview of the proof of the Small Parameter Theorem.

4.2 The Small Parameter Theorem

In this section we state the Small Parameter Theorem (Theorem 21) which tells us that for every PTOCA over one parameter and every sufficiently large parameter value N, accepting N-runs with counter values all in $[0, h \cdot N]$ can be turned into accepting N'-runs for some smaller N'.

We provide an overview of the proof of the Small Parameter Theorem in Section 4.3, whose actual proof will stretch over Sections 4.4, 4.5, 4.6, and 4.7.

For each PTOCA $\mathcal{C} = (Q, P, R, q_{init}, F)$ and every $h \in \mathbb{N}$ we define the following constants.

$$
\begin{aligned}
Z_{\mathcal{C}} &= \mathsf{LCM}(\mathsf{Consts}(\mathcal{C})) \\
\Gamma_{\mathcal{C},h} &= \mathsf{LCM}((4h+1) \cdot |Q|) \cdot Z_{\mathcal{C}} \\
\Upsilon_{\mathcal{C},h} &= (4h+1) \cdot |Q| \cdot \mathsf{LCM}(4h+1 \cdot |Q|) \cdot ((4h+1) \cdot |Q| \cdot Z_{\mathcal{C}} + 2) \\
M_{\mathcal{C},h} &= 30 \cdot (h+1) \cdot (\Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h} + 1)
\end{aligned}
$$

Since for every non-empty finite set $U \subseteq \mathbb{N} \setminus \{0\}$ we have $\mathsf{LCM}(U) \leq \max(U)^{|U|}$, the following lemma is straightforward.

Lemma 20. *The above constants are asymptotically bounded by* $2^{\mathrm{poly}(|\mathcal{C}|+h)}$, *where* $\mathrm{poly}(x) = x^{O(1)}$.

The main result of this section is the following theorem.

Theorem 21 (Small Parameter Theorem)**.** *Let $\mathcal{C} = (Q, \{p\}, R, q_{init}, F)$ be a PTOCA with one parameter p. If there exists an accepting N-run in $\mathcal{C}$ with values all in $[0, h \cdot N]$ for some $N > M_{\mathcal{C},h}$, then there exists an accepting $(N - \Gamma_{\mathcal{C},h})$-run in $\mathcal{C}$.*

Let us first establish that this theorem is enough to prove the desired PSPACE upper bound.

Corollary 22. *h-*BOUNDED 1-PTOCA REACHABILITY *is in* PSPACE.

Proof. Let us fix a PTOCA $\mathcal{C} = (Q, P, R, q_0, F)$ with $P = \{p\}$, such that for all $N \in \mathbb{N}$, if there is an accepting N-runs in $\mathcal{C}$, there exists one satisfying $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$.

We first claim that if there exists an accepting N-run π in $\mathcal{C}$, then there exists one satisfying $N \in [0, M_{\mathcal{C},h}]$ and $\textsc{Values}(\pi) \subseteq [0, h \cdot M_{\mathcal{C},h}]$. All accepting N-runs π of $\mathcal{C}$ satisfy $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$, so if $N > M_{\mathcal{C},h}$, then there exists some accepting $(N - \Gamma_{\mathcal{C},h})$-run in $\mathcal{C}$ by Theorem 21. Remarking that in case $N > M_{\mathcal{C},h}$ we have $N - \Gamma_{\mathcal{C},h} > M_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} > 0$, one can repeat the above argument for $N - \Gamma_{\mathcal{C},h}$ and possibly for $N - 2\Gamma_{\mathcal{C},h}$ and so on, thus implying the desired existence.

Thus it suffices to check in polynomial space in $|\mathcal{C}| + h$ whether there exists some accepting N-run π in $\mathcal{C}$ satisfying $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$ for some $N \in [0, M_{\mathcal{C},h}]$. Since $M_{\mathcal{C},h} \in 2^{\mathrm{poly}(|\mathcal{C}|+h)}$, the latter is simply a reachability question in an exponentially large finite graph all of whose vertices and edges can be represented using polynomially many bits, and thus decidable in polynomial space. □

4.3 Overview of the proof of the Small Parameter Theorem

For the proof of the Small Parameter Theorem (Theorem 21) we proceed as follows.

- In Section 4.4 we introduce the notion of N-semiruns. These generalize N-runs in that only modulo tests need to hold, not however comparison tests. We define some natural operations on them, like shifting them by some value or cutting out certain infixes. In Subsection 4.4.2 we prove two important lemmas on semiruns that will serve as base tools for subsequent steps in the proof:
  - The Depumping Lemma (Lemma 24) will be our main tool to depump certains semiruns, in the following sense: in case the difference between the number of $+p$-transitions and $-p$-transitions is bounded for all infixes and equal to 0 for the whole semirun and furthermore the absolute counter effect of the semirun is sufficiently large, then one can build — by applying the above-mentioned operations — a new semirun whose absolute counter effect is slightly smaller.
  - The Bracket Lemma (Lemma 25) states that in case the counter effect is sufficiently large and the counter values are all in $[0, h \cdot N]$, then one can find an infix where the counter effect is also large and moreover the difference between the number of $+p$-transitions and $-p$-transitions is bounded for all infixes and equal to 0 for the whole semirun.
- In Section 4.5 we introduce the notion of hills and valleys. Hills are N-semiruns that start and end in configurations with low counter values but where all intermediate configurations have counter values above the source and target configuration. We introduce the dual notion of valleys. The main contribution of the section is the following.
  - The Hill and Valley Lemma (Lemma 30) allows to transform N-semiruns that are hills (resp. valleys) into $(N - \Gamma_{\mathcal{C},h})$-semiruns with the same source and target configuration.
- Making use of all of the above lemmas, we introduce in Section 4.6 the following lemma, which is a main technical ingredient in the proof of Theorem 21.
  - The 5/6-Lemma (Lemma 41) states that N-semiruns with counter effect smaller than $5/6 \cdot N$ can be turned in into $(N - \Gamma_{\mathcal{C},h})$-semiruns.

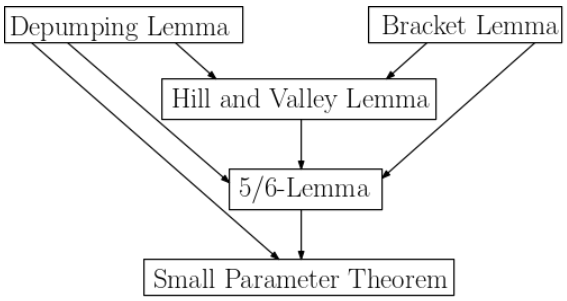

Figure 4.3: Illustration of the dependencies between the lemmas. The presence of an arrow going from a lemma to another means that the lemma in question is used inside the proof of the lemma the arrow points to.

- Finally, in Section 4.7 we prove the Small Parameter Theorem (Theorem 21) by carefully factorizing a potential N-run into subsemiruns that can be treated by the above lemmas.

In Figure 4.3 we give an overview of the dependencies of the above-mentioned lemmas.

4.4 Semiruns, their bracket projection, and embeddings

In this Section we motivate and introduce the notion of semiruns by loosening the conditions on runs, and define basic operations on them. These basic operations possibly change their counter values, length, or counter effect.

The formalism of an N-run is a little bit too restrictive to define operations on them. For instance, subtracting $Z_{\mathcal{C}}$ from all counter values of a μ-run produces an object, where conditions (1),(2), and (3) of the definition of T^{μ}_{C} (page 31) indeed hold — as $Z_{\mathcal{C}} = \mathsf{LCM}(\mathsf{Consts}(\mathcal{C}))$ — but where conditions (4) and (5) might not hold anymore, as comparison guards may be violated. Rather than certifying each time that the application of an operation preserves the property of being an N-run we prefer to loosen the definition in order to avoid tedious case distinctions. This motivates the notion of semitransitions (resp. semiruns), which are a generalization of transitions (resp. runs), in which the comparison tests need not hold.

We introduce semiruns and operations on them in Section 4.4.1. Section 4.4.2 introduces the bracket projection of semiruns, the Depumping Lemma (Lemma 24) and the Bracket Lemma (Lemma 25). Section 4.4.3 introduces the notion of embeddings, which provide a formal means to express when a semirun can structurally be found as a subsequence of another.

4.4.1 Semiruns and operations on them

A parametric one-counter automaton $\mathcal{C} = (Q, P, R, q_{init}, F)$, and a parameter valuation $\mu : P \to \mathbb{N}$ induces a second labeled transition system $\mathcal{T}^{\mu}_{\mathcal{C}} = (\mathsf{Conf}(\mathcal{C}), \lambda_{\mathcal{C}}, \dashrightarrow_{\mathcal{C},\mu})$ where $\lambda_{\mathcal{C}} = \mathsf{Op}(P)$ and where $\overset{\mathcal{C},\mu}{\dashrightarrow}$ is defined such that for all $q(z), q'(z') \in \mathsf{Conf}(\mathcal{C})$, for all $op \in \mathsf{Op}(P)$, $q(z) \overset{op}{\dashrightarrow}_{\mu} q'(z')$ if there exists some $(q, op, q') \in R$ such that conditions (1),(2), and (3) of $T^{\mu}_{\mathcal{C}}$ hold but where conditions (4) and (5) are loosened by the following conditions (4') and (5') respectively

(1) $op = c \in \mathsf{Op}_{\pm}$, and $z' = z + c$,

(2) $op \in \mathsf{Op}_{\pm P}$ and either

- $op = +p$ and $z' = z + \mu(p)$, or
- $op = -p$ and $z' = z - \mu(p)$.

(3) $op = \bmod c \in \mathsf{Op}_{\mathsf{mod}}$, $z = z'$ and $z' \equiv 0 \bmod c$,

(4') $op = \bowtie c \in \mathsf{Op}_{\bowtie}$ and $z = z'$, and

(5') $op = \bowtie p \in \mathsf{Op}_{\bowtie P}$, and $z = z'$.

Thus, in a nutshell, when writing $q(z) \overset{op}{\dashrightarrow}_{\mu} q'(z')$ we do not require that the comparison tests against parameters or against constants hold; however the updates and the modulo tests against constants must be respected. To differentiate $q(z) \overset{op}{\dashrightarrow}_{\mu} q'(z')$ from $q(z) \overset{op}{\longrightarrow}_{\mu} q'(z')$, we will call the former a *semitransition*. This naturally gives rise to the definition of *μ-semiruns* as expected. Note that in particular every μ-run is a μ-semirun. The abbreviation N-semirun, $q(z) \overset{op}{\dashrightarrow}_{N} q'(z')$, the counter effect Δ, VALUES, min, max, subsemirun, prefix, suffix are defined as for runs.

Note that in particular every N-run is an N-semirun. Importantly, note also that semitransitions involving comparison tests are still syntactically present in semiruns. By a careful analysis, one can therefore possibly perform operations on N-semiruns in order to show that they are in fact N-runs.

Example 23. *The* 2*-semirun*

$$\pi \;=\; q_0(0) \overset{+1}{\dashrightarrow}_2 q_1(1) \overset{+1}{\dashrightarrow}_2 q_1(2) \overset{+1}{\dashrightarrow}_2 q_1(3) \overset{\leq p}{\dashrightarrow}_2 q_2(3) \overset{\mathsf{mod}\ 3}{\dashrightarrow}_2 q_3(3)$$

is not a 2*-run, as, in* $q_1(3) \overset{\leq p}{\dashrightarrow}_2 q_2(3)$*, condition (4) of Definition 4.1 does not hold, however condition (4') of* $\mathcal{T}^{\mu}_{\mathcal{C}}$*'s definition on page 36 does.*

Shifting and gluing of semiruns

Let us fix a PTOCA $\mathcal{C}$ and some N-semirun

$$\pi \;=\; q_0(z_0) \overset{\pi_0}{\dashrightarrow}_N q_1(z_1) \cdots \overset{\pi_{n-2}}{\dashrightarrow}_N q_{n-1}(z_{n-1}) \overset{\pi_{n-1}}{\dashrightarrow}_N q_n(z_n).$$

We define the following operations, where we recall that $Z_{\mathcal{C}} = \mathsf{LCM}(\mathsf{Consts}(\mathcal{C}))$.

- For $D \in Z_{\mathcal{C}}\mathbb{Z}$, we define the *shifting* of π by D as

  $$\pi + D \;=\; q_0(z_0 + D) \overset{\pi_0}{\dashrightarrow}_N q_1(z_1 + D) \cdots \overset{\pi_{n-1}}{\dashrightarrow}_N q_n(z_n + D).$$

  Since there are no effective comparison tests and D is an integer that is divisible by all constants appearing in modulo tests in $\mathcal{C}$, it is clear that $\pi + D$ is again an N-semirun.

- For two configurations $q_i(z_i)$ and $q_j(z_j)$ with $0 \leq i < j \leq n$ and where $D = z_j - z_i \in Z_{\mathcal{C}}\mathbb{Z}$ is a multiple of $Z_{\mathcal{C}}$ and $q_i = q_j$, we define the *gluing* of the configurations as

  $$\pi - [i, j] \;=\; q_0(z_0) \;\cdots\; \overset{\pi_{i-1}}{\dashrightarrow}_N q_i(z_i) \overset{\pi_j}{\dashrightarrow}_N q_{j+1}(z_{j+1} - D) \cdots \overset{\pi_{n-1}}{\dashrightarrow}_N q_n(z_n - D).$$

When gluing the leftmost and rightmost configurations of pairwise non-intersecting intervals $I_1 = [a_1, b_1], \ldots, I_k = [a_k, b_k] \subseteq [0, n]$, assuming $b_i < a_{i+1}$ for all $1 \leq i < k$, and $q_{a_i} = q_{b_i}$ and $z_{b_i} - z_{a_i} \in Z_C\mathbb{Z}$ for all $1 \leq i \leq k$, we will use $\pi - I_1 - I_2 \cdots - I_k$ to denote the result corresponding to gluing each interval successively while shifting the others accordingly, instead of writing the more tedious $\pi^{(k)}$, where

$$\begin{aligned}
\pi^{(1)} &= \pi - [a_1, b_1], \\
\pi^{(2)} &= \pi^{(1)} - [a_2 - (|I_1| - 1), b_2 - (|I_1| - 1)], \\
&\cdots \\
\pi^{(k)} &= \pi^{(k-1)} - [a_k - \sum_{1 \leq j < k} (|I_j| - 1), b_k - \sum_{1 \leq j < k} (|I_j| - 1)].
\end{aligned}$$

4.4.2 The bracket projection of semiruns

In this section we define a projection ϕ of semitransitions $\tau = q(z) \xrightarrow{op}_N q'(z')$ to a word over the binary alphabet $\{[,]\}$, where transitions with $op = +p$ are mapped to $[$, transitions with $op = -p$ are mapped to $]$, and all other transitions are mapped to the empty word ε. The projection ϕ is naturally extended to a morphism from semiruns to $\{[,]\}^*$. In this section we will show the following lemmas.

- The Depumping Lemma (Lemma 24) states that for each N-semirun whose ϕ-projection has bounded bracketing properties and that has a counter effect whose absolute value is sufficiently large there exists another N-semirun with a counter effect whose absolute value is slightly smaller. This latter resulting N-semirun has a particular form in that it can be obtained from the original N-semirun by applying the above-mentioned operations of shifting and gluing: notably, the subsemiruns that are being glued themselves have a ϕ-projection that has bounded bracketing properties.
- The Bracket Lemma (Lemma 25) states that if an N-semirun has all its counter values in $[0, h \cdot N]$, has an absolute counter effect that is sufficiently large and has a ϕ-projection that satisfies a suitable threshold condition on the number of occurrences of $[$ and $]$, that there is a subsemirun where the absolute counter effect is also large and whose ϕ-projection has bounded bracketing properties.

Formally, we define a mapping ϕ such that for every semitransition $\tau = q(z) \xrightarrow{op}_N q'(z')$,

$$\phi(\tau) = \begin{cases} [& \text{if } op = +p, \\ ] & \text{if } op = -p, \\ \varepsilon & \text{otherwise.} \end{cases}$$

Note that an N-semirun π can contain several $+p$-transitions and $-p$ transitions. We introduce the notation $\phi(\pi, i) = \phi(\pi[i, i+1])$ to denote the ϕ-projection of the i-th transition of π for all $i \in [0, |\pi| - 1]$. The mapping ϕ is naturally extended to a morphism on semiruns to words over the binary alphabet $\{[,]\}$ as expected $\phi(\pi) = \phi(\pi, 0)\phi(\pi, 1) \cdots \phi(\pi, |\pi| - 1)$.

We are particularly interested in N-semiruns whose projection by ϕ contains as many opening as closing brackets and only a few pending ones (when read from left to right). To make this formal, for all $k \in \mathbb{N}$ we define the regular language

$$\Lambda_k = \left\{ w \in \{[,]\}^* : |w|_[= |w|_], \forall u, v \in \{[,]\}^*.\ uv = w \implies |u|_[- |u|_] \in [-k, k] \right\}.$$

We are interested in analysing N-semiruns with counter values in $[0, h \cdot N]$. Bounding the counter values like this limits the number of $+p$ (resp. $-p$) that can appear in a row. This will be the basis in the Bracket Lemma which amounts to showing the existence of subsemiruns whose ϕ-projection is in Λ_{2h}.

The now following Depumping Lemma will enable us to reduce the counter effect of N-semiruns whose ϕ-projection is in Λ_{2h}. It is worth remarking that $\Gamma_{\mathcal{C},h} \ll \Upsilon_{\mathcal{C},h}$, recalling the definition of our constants on page 33.

Lemma 24 (Depumping Lemma)**.** *For all N-semiruns π satisfying $\phi(\pi) \in \Lambda_{2h}$ and $|\Delta(\pi)| > \Upsilon_{\mathcal{C},h}$ there exists an N-semirun π' such that either*

- *$\Delta(\pi) > \Upsilon_{\mathcal{C},h}$ and $\Delta(\pi') = \Delta(\pi) - \Gamma_{\mathcal{C},h}$, or*
- *$\Delta(\pi) < -\Upsilon_{\mathcal{C},h}$ and $\Delta(\pi') = \Delta(\pi) + \Gamma_{\mathcal{C},h}$.*

Moreover, $\pi' = \pi - I_1 - I_2 \cdots - I_k$ for pairwise disjoint intervals $I_1, \ldots, I_k \subseteq [0, |\pi|]$ such that we have $\phi(\pi[I_i]) \in \Lambda_{4h}$ for all $i \in [1,k]$, and either $\Delta(\pi[I_i]) > 0$ for all $i \in [1,k]$ or $\Delta(\pi[I_i]) < 0$ for all $i \in [1,k]$.

Proof. Let $\pi = q_0(z_0) \xrightarrow{\pi_0}_N q_1(z_1) \xrightarrow{\pi_1}_N \cdots \xrightarrow{\pi_{n-1}}_N q_n(z_n)$ be an N-semirun such that $\phi(\pi) \in \Lambda_{2h}$. We will assume without loss of generality that $\Delta(\pi) > \Upsilon_{\mathcal{C},h}$. The dual case when $\Delta(\pi) < -\Upsilon_{\mathcal{C},h}$ can be proven analogously.
For every position $i \in [0,n]$ let us define

$$\lambda(i) = |\phi(\pi[0,i])|_[- |\phi(\pi[0,i])|_] \qquad \text{and} \qquad \mathsf{pot}(i) = z_i - z_0 - \lambda(i) \cdot N.$$

Note that since $\phi(\pi) \in \Lambda_{2h}$ we have for all $i \in [0,n]$,

$$\lambda(i) \in [-2h, 2h], \tag{4.1}$$

and moreover

$$\phi(\pi[0,i]) \in \Lambda_{2h} \iff \lambda(i) = 0. \tag{4.2}$$

We note the following important properties of pot,

1. $|\mathsf{pot}(i-1) - \mathsf{pot}(i)| \leq 1$ for all $i \in [1,n]$,
2. $\mathsf{pot}(0) = 0$,
3. for all $0 \leq i < j \leq n$, if $\lambda(i) = \lambda(j)$, then $\mathsf{pot}(j) - \mathsf{pot}(i) = z_j - z_i$, and
4. $\mathsf{pot}(n) = z_n - z_0 = \Delta(\pi)$ since $\lambda(0) = \lambda(n) = 0$.

The following claim states that if in a subsemirun the pot increment is sufficiently large, then one can find a subsemirun therein that can potentially be glued.

Claim 1. *For each subsemirun $\pi[a,b]$ that satisfies $\mathsf{pot}(b) - \mathsf{pot}(a) > (4h+1) \cdot |Q| \cdot Z_{\mathcal{C}}$ there exist positions $a \leq s < t \leq b$, such that*

- *$q_s = q_t$,*
- *$\lambda(s) = \lambda(t)$, and*
- *$z_t - z_s = dZ_{\mathcal{C}}$ for some $d \in [1, (4h+1) \cdot |Q|]$.*

Proof of the Claim. Since by assumption $\mathsf{pot}(b) - \mathsf{pot}(a) > (4h+1) \cdot |Q| \cdot Z_{\mathcal{C}}$, by the pigeonhole principle and Point 1 above, there exist two indices $a \leq s < t \leq b$ such that $q_s = q_t$, $\lambda(s) \in [-2h, 2h]$ and $\lambda(t) \in [-2h, 2h]$ are equal, and $\mathsf{pot}(t) - \mathsf{pot}(s) = dZ_{\mathcal{C}}$ for some $d \in [1, (4h+1) \cdot |Q|]$. By Point 3 above, from $\lambda(t) = \lambda(s)$, it follows $z_t - z_s = \mathsf{pot}(t) - \mathsf{pot}(s) = dZ_{\mathcal{C}}$.
(End of the proof of the Claim) □

Since $\mathsf{pot}(i) - \mathsf{pot}(i-1) \leq 1$ for all $i \in [1,n]$ by Point 1 above and

$$\begin{array}{rcl} \mathsf{pot}(n) - \mathsf{pot}(0) & = & z_n - z_0 \\ & = & \Delta(\pi) \\ & > & \Upsilon_{\mathcal{C},h} \\ & \overset{\text{page 33}}{=} & (4h+1)\cdot|Q|\cdot\mathsf{LCM}((4h+1)\cdot|Q|)\cdot((4h+1)\cdot|Q|\cdot Z_{\mathcal{C}}+2)\,, \end{array}$$

by the pigeonhole principle, there exist at least

$$(4h+1)\cdot|Q|\cdot\mathsf{LCM}((4h+1)\cdot|Q|)$$

pairwise disjoint subsemiruns $\pi[a,b]$ satisfying $\mathsf{pot}(b) - \mathsf{pot}(a) > (4h+1)\cdot|Q|\cdot Z_{\mathcal{C}}$. Let

$$L = \mathsf{LCM}((4h+1)\cdot|Q|),$$

and let $\pi[a_1,b_1],\ldots,\pi[a_{(4h+1)\cdot|Q|\cdot L},b_{(4h+1)\cdot|Q|\cdot L}]$ be an enumeration of these latter subsemiruns. We apply the above Claim to all of these $\pi[a_i,b_i]$: there exist positions $a_i \leq s_i \leq t_i \leq b_i$ such that $\lambda(s_i) = \lambda(t_i)$, $q_{s_i} = q_{t_i}$, and $z_{t_i} = z_{s_i} + d_i Z_{\mathcal{C}}$ for some $d_i \in [1,(4h+1)\cdot|Q|]$. From $\lambda(s_i) = \lambda(t_i)$ and (4.1) it follows $\phi(\pi[s_i,t_i]) \in \Lambda_{4h}$. Recall that $\Gamma_{\mathcal{C},h} = \mathsf{LCM}((4h+1)\cdot|Q|)\cdot Z_{\mathcal{C}} = L\cdot Z_{\mathcal{C}}$, cf. page 33. By the pigeonhole principle, among these $(4h+1)\cdot|Q|\cdot L$ pairwise disjoint subsemiruns $\pi[a_i,b_i]$, there exists some $d \in [1,(4h+1)\cdot|Q|]$ such that there are L/d many different $\pi[a_i,b_i]$ all satisfying $d_i = d$. Let $\pi[a_{i_1},b_{i_1}],\ldots,\pi[a_{i_{L/d}},b_{i_{L/d}}]$ be an enumeration of these latter $\pi[a_i,b_i]$. Note that for all of these $\pi[a_i,b_i]$ we have $\Delta(\pi[s_{i_j},t_{i_j}]) = d\cdot Z_{\mathcal{C}}$. Since moreover $q_{s_{i_j}} = q_{t_{i_j}}$ we know that, for all $j \in [1,L/d]$, the gluing $\pi - [s_{i_j},t_{i_j}]$ is an N-semirun with $\Delta(\pi - [s_{i_j},t_{i_j}]) = \Delta(\pi) - dZ_{\mathcal{C}}$. Thus,

$$\pi' \quad = \quad \pi - [s_{i_1},t_{i_1}] - \ldots - [s_{i_{L/d}},t_{i_{L/d}}]$$

is an N-semirun satisfying $\Delta(\pi') = \Delta(\pi) - d\cdot(L/d)\cdot Z_{\mathcal{C}} = \Delta(\pi) - \Gamma_{\mathcal{C},h}$ as required. □

Let us now introduce the Bracket Lemma, which states that in case the absolute value of the counter effect of an N-semirun is sufficiently large, the counter values are all in $[0,h\cdot N]$ and a majority condition holds on the number of occurrences of $[$ and $]$ in its ϕ-projection, that there is a subsemirun where the counter effect is also large and that moreover has good bracketing properties (in the sense of the Depumping Lemma). Roughly speaking, it is based on the idea that if the values of a semirun are all in $[0,h\cdot N]$, there cannot be $h+1$ $+p$-transitions in a row. Technically speaking, the Bracket Lemma can be applied to $(N-\Gamma_{\mathcal{C},h})$-semiruns, where N is sufficiently large: the reason is that the Bracket Lemma will later be applied to N-semiruns in which some of the $+p/-p$-transitions have already been modified ("by hand") to have an effect $(N-\Gamma_{\mathcal{C},h})/-(N-\Gamma_{\mathcal{C},h})$ instead of $N/-N$.

Lemma 25 (Bracket Lemma)**.** *For all $N > M_{\mathcal{C},h}$, all $(N-\Gamma_{\mathcal{C},h})$-semiruns π satisfying* $\textsc{Values}(\pi) \subseteq [0,h\cdot N]$, *$\Delta(\pi) < -\Upsilon_{\mathcal{C},h}$ (resp. $\Delta(\pi) > \Upsilon_{\mathcal{C},h}$) and where $\phi(\pi)$ contains at least as many occurrences of $[$ as occurrences of $]$ (resp. at least as many occurrences of $]$ as occurrences of $[$) there exists a subsemirun $\pi[c,d]$ satisfying $\phi(\pi[c,d]) \in \Lambda_{2h}$ and $\Delta(\pi[c,d]) < -\Upsilon_{\mathcal{C},h}$ (resp. $\Delta(\pi[c,d]) > \Upsilon_{\mathcal{C},h}$).*

Proof. We only prove the case where $\Delta(\pi) < -\Upsilon_{\mathcal{C},h}$ and $\phi(\pi)$ contains at least as many occurrences of $[$ as of $]$. The dual case when $\Delta(\pi) > \Upsilon_{\mathcal{C},h}$ and $\phi(\pi)$ contains at least as many $]$ as of $[$ can be proven analogously.

As in the proof of Lemma 24, for any word $u \in \{[,]\}^*$ let $\lambda(u) = |u|_[- |u|_]$. For the rest of the proof assume by contradiction that there is no such subsemirun $\pi[c,d]$ satisfying $\Delta(\pi[c,d]) < -\Upsilon_{\mathcal{C},h}$ and $\phi(\pi[c,d]) \in \Lambda_{2h}$, or, equivalently, that every subsemirun $\pi[c,d]$ with $\phi(\pi[c,d]) \in \Lambda_{2h}$ satisfies $\Delta(\pi[c,d]) \geq -\Upsilon_{\mathcal{C},h}$.

For all $k \geq 0$ let

$$\Psi_k = \{w \in \{[,]\}^* \mid \forall uv = w : \lambda(u) \in [-k, k]\}$$

denote the set of all words over the alphabet $\{[,]\}$, where for each prefix the absolute difference between the number of occurrences of [and of] is at most k. Note that

$$\Lambda_k = \Psi_k \cap \lambda^{-1}(0). \tag{4.3}$$

Under the above assumptions on π, for the sake of contradiction, we have three claims on properties on the image of ϕ applied to π and subsemiruns thereof.

Claim 1. $\phi(\pi) \in \Psi_h$.

Proof of Claim 1. Let us write $\pi = \pi[0, n]$. Assume by contradiction that $\phi(\pi) \notin \Psi_h$. Let u be a shortest prefix of $\phi(\pi)$ such that $\lambda(u) \notin [-h, h]$. Let us first consider the case when $\lambda(u) > h$.

By definition of u we have $\lambda(u) = h + 1$ and there are indices $0 \leq t_1 < \cdots < t_{h+1} < n$ such that

- $\phi(\pi, t_1) = \ldots = \phi(\pi, t_{h+1}) = [$, and
- $\phi(\pi[t_i + 1, t_{i+1}]) \in \Lambda_h$ for all $i \in [1, h]$.

Recall that by our assumption every subsemirun $\pi[c, d]$ of π with $\phi(\pi[c, d]) \in \Lambda_{2h}$ satisfies $\Delta(\pi[c, d]) \geq -\Upsilon_{\mathcal{C},h}$. Since $\bigcup_{i \in [1,2h]} \Lambda_i = \Lambda_{2h}$ it follows $\Delta(\pi[t_i + 1, t_{i+1}]) \geq -\Upsilon_{\mathcal{C},h}$ for all $i \in [1, h]$. Moreover, bearing in mind that π is an $(N - \Gamma_{\mathcal{C},h})$-semirun, we obtain $\Delta(\pi, t_i) = N - \Gamma_{\mathcal{C},h}$. Altogether, as $N > M_{\mathcal{C},h}$ by assumption, we obtain

$$\begin{aligned}
\Delta(\pi[t_1, t_{h+1} + 1]) &\geq -h \cdot \Upsilon_{\mathcal{C},h} + (h + 1) \cdot (N - \Gamma_{\mathcal{C},h}) \\
&> h \cdot N + N - (h + 1) \cdot (\Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h}) \\
&> h \cdot N + M_{\mathcal{C},h} - (h + 1) \cdot (\Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h}) \\
&> h \cdot N,
\end{aligned}$$

where the last inequality follows from $M_{\mathcal{C},h}$'s definition on page 33, hence contradicting $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$.

Let us now consider the case when $\lambda(u) < -h$. Again, by definition of u, we have $\lambda(u) = -h - 1$. There are hence indices $0 \leq t_1 < \ldots < t_{h+1} < n$ such that

$$\phi(\pi, t_1) = \ldots = \phi(\pi, t_{h+1}) =],$$

and moreover $\phi(\pi[0, t_1]) \in \Lambda_h$ and $\phi(\pi[t_i + 1, t_{i+1}]) \in \Lambda_h$ for all $i \in [1, h]$. By assumption $\phi(\pi)$ contains at least as many [as]. Therefore there must exist $h + 1$ further positions $t'_1, \ldots, t'_{h+1}$ in π satisfying $0 \leq t_1 < \ldots < t_{h+1} < t'_1 < t'_2 < \ldots < t'_{h+1} < n$ such that

$$\phi(\pi, t'_1) = \ldots = \phi(\pi, t'_{h+1}) = [$$

and $\phi(\pi[t'_i + 1, t'_{i+1}]) \in \Lambda_h$ for all $i \in [1, h]$. Again taking into account our assumption that $\Delta(\pi[c, d]) \geq -\Upsilon_{\mathcal{C},h}$ for all subsemiruns $\pi[c, d]$ with $\phi(\pi[c, d]) \in \Lambda_{2h}$, it follows as above, that $\Delta(\pi[t'_1, t'_{h+1} + 1]) > h \cdot N$, contradicting $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$. □

Claim 2. $\phi(\pi[a, b]) \in \Psi_{2h}$ for all subsemiruns $\pi[a, b]$ of π.

Proof of Claim 2. This is an immediate consequence of Claim 1. Indeed, any subsemirun $\pi[a, b]$ of π satisfying $\phi(\pi[a, b]) \notin \Psi_{2h}$ gives rise to a prefix u of $\phi(\pi)$ such that $u \notin \Psi_h$ and hence $\phi(\pi) \notin \Psi_h$. □

Claim 3. For all subsemiruns $\pi[a, b]$ of π, if $\lambda(\phi(\pi[a, b])) > 0$, then $\Delta(\pi[a, b]) > \Upsilon_{\mathcal{C}}$.

Proof of Claim 3. We prove the statement by induction on $\lambda(\phi(\pi[a, b]))$.

For the induction base, assume $\lambda(\phi([a,b])) = 1$. Thus, there exists a position $t \in [a,b]$ such that $\phi(\pi,t) = [$ and $\lambda(\pi[a,t]) = \lambda(\phi(\pi[t+1,b])) = 0$. By Claim 2 and (4.3) we have $\phi(\pi[a,t]), \phi(\pi[t+1,b]) \in \Lambda_{2h}$. Thus, $\Delta(\pi[a,t]), \Delta(\pi[t+1,b]) > -\Upsilon_{\mathcal{C},h}$ by our assumption. Hence, we obtain

$$\begin{aligned}\Delta(\pi[a,b]) &= \Delta(\pi[a,t]) + \Delta(\pi,t) + \Delta(\pi[t+1,b]) \\ &\geq -\Upsilon_{\mathcal{C},h} + (N - \Gamma_{\mathcal{C},h}) - \Upsilon_{\mathcal{C},h} \\ &> M_{\mathcal{C},h} - 2\Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} \\ &> \Upsilon_{\mathcal{C},h},\end{aligned}$$

where the last strict inequality follows from definition of $M_{\mathcal{C},h}$ on page 33.

Assume $\lambda(\phi(\pi[a,b])) > 1$. Consider the smallest position $t \in [a,b]$ such that $\lambda(\phi(\pi[a,t])) = 0$ and $\phi(\pi,t) = [$. By Claim 2 and (4.3) it follows that $\phi(\pi[a,t]) \in \Lambda_{2h}$ and hence $\Delta(\pi[a,t]) \geq -\Upsilon_{\mathcal{C},h}$ by our assumption. Moreover, $\lambda(\phi(\pi[t+1,b])) = \lambda(\phi(\pi[a,b])) - 1$. We can thus apply induction hypothesis to $\pi[t+1,b]$ and obtain

$$\begin{aligned}\Delta(\pi[a,b]) &= \Delta(\pi[a,t]) + \Delta(\pi,t) + \Delta(\pi[t+1,b]) \\ &> \Delta(\pi[a,t]) + \Delta(\pi,t) + \Upsilon_{\mathcal{C},h} \\ &\geq -\Upsilon_{\mathcal{C},h} + (N - \Gamma_{\mathcal{C},h}) + \Upsilon_{\mathcal{C},h} \\ &> M_{\mathcal{C},h} - \Gamma_{\mathcal{C},h}, \\ &> \Upsilon_{\mathcal{C},h},\end{aligned}$$

where the first strict inequality follows from induction hypothesis on $\pi[t+1,b]$ and the last strict inequality follows from definition of $M_{\mathcal{C},h}$ on page 33. □

We will now contradict our initial assumption that there is no subsemirun $\pi[c,d]$ satisfying $\phi(\pi[c,d]) \in \Lambda_{2h}$ and $\Delta(\pi[c,d]) < -\Upsilon_{\mathcal{C},h}$ by making use of the above claims.

Since π itself satisfies $\Delta(\pi) < -\Upsilon_{\mathcal{C},h}$, it follows $\phi(\pi) \notin \Lambda_{2h} = \Psi_{2h} \cap \lambda^{-1}(0)$ by our assumption and (4.3). But since $\phi(\pi) \in \Psi_{2h}$ by Claim 2, it follows $\lambda(\phi(\pi)) \neq 0$.

As $\phi(\pi)$ contains at least as many occurrences of $[$ as occurrences of $]$ by assumption, $\phi(\pi)$ must contain strictly more occurrences of $[$ than of $]$, i.e. $\lambda(\phi(\pi)) > 0$. By Claim 3 it follows $\Delta(\pi) > \Upsilon_{\mathcal{C},h}$, contradicting our assumption that $\Delta(\pi) < -\Upsilon_{\mathcal{C},h}$. □

4.4.3 Embeddings of semiruns

The Small Parameter Theorem (Theorem 21) turns N-runs with values in $[0, h \cdot N]$ into $(N - \Gamma_{\mathcal{C},h})$-runs. In proving this, we prefer to view N-runs as N-semiruns. Indeed, we first view any N-run as an N-semirun and then apply certain of the above-mentioned operations on them to obtain some $(N - \Gamma_{\mathcal{C},h})$-semirun. However, we would then like to claim that the resulting $(N - \Gamma_{\mathcal{C},h})$-semirun is in fact an $(N - \Gamma_{\mathcal{C},h})$-run as desired, in particular the comparison tests need to hold. To do so, we introduce a notion when an N-semirun can be embedded into an M-semirun (possibly $N \neq M$) in the sense that operations are being preserved, source and target states are being preserved, and that with respect to some line $\ell \in \mathbb{Z}$ the counter value of each configuration of the embedding has the same orientation with respect to ℓ as the counter value of the configuration it corresponds to.

Definition 26 (ℓ-embedding)**.** *Let $\ell \in \mathbb{Z}$. An N-semirun*

$$\sigma \quad = \quad s_0(y_0) \xrightarrow{\sigma_0}_N s_1(y_1) \cdots \xrightarrow{\sigma_{n-1}}_N s_n(y_n)$$

is an ℓ-embedding *of an M-semirun*

$$\pi \quad = \quad q_0(z_0) \xrightarrow{\pi_0}_M q_1(z_1) \cdots \xrightarrow{\pi_{m-1}}_M q_m(z_m)$$

if $s_0 = q_0$, $s_n = q_m$ and there exists an order-preserving injective mapping $\psi : [0,n] \to [0,m]$ such that

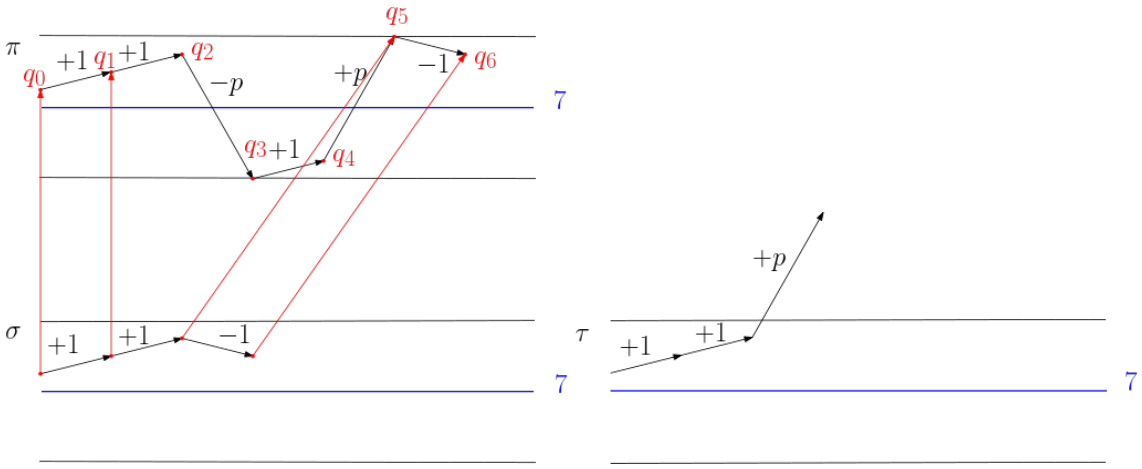

Figure 4.4: Example of a semirun σ that could possibly be an embedding of the semirun π and a semirun τ that cannot.

- $\sigma_i = \pi_{\psi(i)}$ *for all* $i \in [0, n-1]$, *and*
- $\ell \bowtie y_i$ *if, and only if,* $\ell \bowtie z_{\psi(i)}$ *for all* $\bowtie \in \{<, =, >\}$ *and all* $i \in [0, n]$.

Moreover we say σ *is*

- max-falling (w.r.t π) *if* $\max(\sigma) \leq \max(\pi)$, *and*
- min-rising (w.r.t. π) *if* $\min(\sigma) \geq \min(\pi)$.

Example 27. *Consider the semiruns* π, σ *and* τ *in Figure 4.4, where neither concrete counter values nor the states of* σ *and* τ *are mentioned. The semirun* σ *can possibly be a* 7*-embedding of* π *(if its source is* q_0 *and its target state is* q_6*). However,* τ *cannot be a* 7*-embedding of* π*. Indeed, for every possible* ψ *such that* $\tau_2 = +p = \pi_{\psi(2)}$*, the counter value of* τ *at position* 2 *is strictly larger than* 7*, whereas the counter value of* π *at position* $\psi(2)$ *is strictly below* 7*.*

The following remark is implictly being used in subsequent sections.

Remark 28. *Embeddings possess some useful properties that all follow immediately from definition.*

- Transitivity. *Let* π*,* ρ *and* σ *be semiruns such that* π *is an* ℓ*-embedding of* ρ *and* ρ *is an* ℓ*-embedding of* σ*. Then* π *is an* ℓ*-embedding of* σ*. Moreover, if* π *was max-falling (resp. min-rising) w.r.t.* ρ *and* ρ *was max-falling (resp. min-rising) w.r.t.* σ*, then* π *is max-falling (resp. min-rising) w.r.t.* σ*.*
- Closure under concatenation. *Let* π *be an* N*-semirun from* $q(x)$ *to* $r(y)$ *and let* ρ *be* N*-semirun from* $r(y)$ *to* $s(z)$*. Moreover, let* π' *be an* N'*-semirun from* $q(x')$ *to* $r(y')$ *that is an* ℓ*-embedding of* π *and let* ρ' *be an* N'*-semirun from* $r(y')$ *to* $s(z')$ *that is an* ℓ*-embedding of* ρ*. Then* $\pi'\rho'$ *is an* ℓ*-embedding of* $\pi\rho$*. If furthermore,* π' *was max-falling (resp. min-rising) w.r.t.* π *and* ρ' *was max-falling (resp. min-rising) w.r.t.* ρ*, then* $\pi'\rho'$ *is max-falling (resp. min-rising) w.r.t.* $\pi\rho$*.*
- Shifting distant embeddings. *Let* $D \in Z_C\mathbb{Z}$ *be a multiple of* Z_C*, let* π *be a semirun and let* ρ *be an* ℓ*-embedding of* π *such that for all configurations* $q(z)$ *in* ρ *we have* $|z - \ell| > |D|$*. Then both* $\rho + D$ *and* $\rho - D$ *are* ℓ*-embeddings of* π*.*

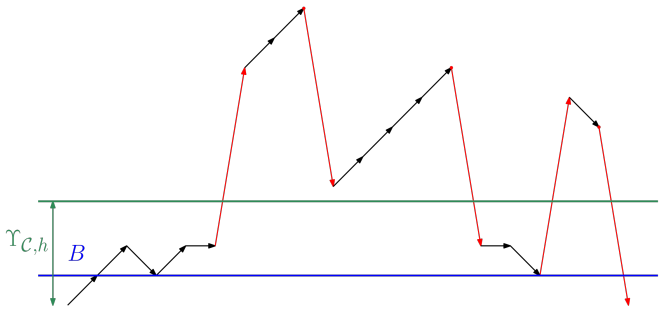

Figure 4.5: Illustration of a B-hill.

4.5 On hills and valleys

In this section we introduce the notions of hills and valleys. Hills are semiruns that start and end in configurations with low counter values but where all intermediate configurations have counter values above these source and target configurations, and where moreover $+p$-transitions (resp. $-p$-transitions) are followed (resp. preceded) by semiruns with absolute counter effect larger than $\Upsilon_{\mathcal{C},h}$ (we refer to Figure 4.5 for an illustration of the concept). We also introduce the dual notion of valleys. We then prove that an N-semirun that is either a hill or a valley can be turned into an $(N - \Gamma_{\mathcal{C},h})$-semirun with the same source and target configuration that is an embedding. This lowering process serves as a building block in the proof of the 5/6-Lemma (Lemma 41).

Definition 29 (Hills and Valleys)**.** *An N-semirun*

$$q_0(z_0) \xrightarrow{\pi_0}_N q_1(z_1) \xrightarrow{\pi_1}_N q_2(z_2) \quad \cdots \quad \xrightarrow{\pi_{n-1}}_N q_n(z_n)$$

is a

- B-hill *if*
  - $z_0, z_n < B$,
  - $z_i \geq B$ *for all* $i \in [1, n-1]$,
  - $\pi_i = -p$ *implies* $z_i > z_0 + \Upsilon_{\mathcal{C},h}$ *for all* $i \in [0, n-1]$, *and*
  - $\pi_i = +p$ *implies* $z_{i+1} > z_n + \Upsilon_{\mathcal{C},h}$ *for all* $i \in [0, n-1]$.
- B-valley *if*
  - $z_0, z_n > B$,
  - $z_i \leq B$ *for all* $i \in [1, n-1]$,
  - $\pi_i = -p$ *implies* $z_{i+1} < z_n - \Upsilon_{\mathcal{C},h}$ *for all* $i \in [0, n-1]$, *and*
  - $\pi_i = +p$ *implies* $z_i < z_0 - \Upsilon_{\mathcal{C},h}$ *for all* $i \in [0, n-1]$.

The Hill and Valley Lemma states that an N-semirun π that is either a B-hill or a B-valley can be turned into an $(N - \Gamma_{\mathcal{C},h})$-semirun with the same source and target configuration that is moreover both a min-rising and max-falling B'-embedding of π, where B' is close to B.

Lemma 30 (Hill and Valley Lemma)**.** *For all $N, B \in \mathbb{N}$, all N-semiruns π from $q_0(z_0)$ to $q_n(z_n)$ with $N > M_{\mathcal{C},h}$ and* $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$ *such that moreover π is either a B-hill or a B-valley, there exists an $(N - \Gamma_{\mathcal{C},h})$-semirun from $q_0(z_0)$ to $q_n(z_n)$ that is both a min-rising and max-falling $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embedding of π (in case π is a B-hill), or both a min-rising and max-falling $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h} + 1)$-embedding of π (in case π is a B-valley).*

We remark that the resulting $(N - \Gamma_{\mathcal{C},h})$-semirun satisfies further properties — these are being discussed in Section 4.5.

Before proving the Hill and Valley Lemma let us explain why the finding of the resulting embedding is delicate. Let us fix any N-semirun

$$\pi \quad = \quad q_0(z_0) \xrightarrow{\pi_0}_N q_1(z_1) \xrightarrow{\pi_1}_N \quad \cdots \quad \xrightarrow{\pi_{n-1}}_N q_n(z_n)$$

from $q_0(z_0)$ to $q_n(z_n)$ with $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$ and $N > M_{\mathcal{C},h}$. Let us moreover assume that π is a B-hill for some $B \in \mathbb{N}$. We need to show the existence of some $(N - \Gamma_{\mathcal{C},h})$-semirun from $q_0(z_0)$ to $q_n(z_n)$ that is moreover both a min-rising and max-falling $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embedding of π.

We are particularly interested in those transitions τ with absolute counter effect $|\Delta(\tau)| = N$, i.e. transitions with operation $+p$ or $-p$ that we will denote as *unlowered* $+p$-transitions and $-p$-transitions respectively. Note that if there is no such transition in π, then π is already an $(N - \Gamma_{\mathcal{C},h})$-semirun. Let us therefore assume there is at least one transition with absolute counter effect N in π. For obtaining only an $(N - \Gamma_{\mathcal{C},h})$-semirun it would simply suffice to lower the absolute counter effect of these transitions by $\Gamma_{\mathcal{C},h}$. Indeed, if the transition $\tau = q(z) \xrightarrow{+p}_N q'(z')$ is an N-semirun, then the *lowered transition* $\widehat{\tau} = q(z) \xrightarrow{+p}_{N-\Gamma_{\mathcal{C},h}} q'(z' - \Gamma_{\mathcal{C},h})$ is an $(N - \Gamma_{\mathcal{C},h})$-semirun. Dually, if $\tau = q(z) \xrightarrow{-p}_N q'(z')$ is an N-semirun, then $\widehat{\tau} = q(z) \xrightarrow{-p}_{N-\Gamma_{\mathcal{C},h}} q'(z' + \Gamma_{\mathcal{C},h})$ is an $(N - \Gamma_{\mathcal{C},h})$-semirun.

Thus, applying such a lowering to all transitions of π whose absolute counter effect is N yields an $(N - \Gamma_{\mathcal{C},h})$-semirun with target configuration shifted by a multiple of $\Gamma_{\mathcal{C},h}$, according to the operations seen in Subsection 4.4. However, the Hill and Valley Lemma not only requires the resulting semirun to be an $(N - \Gamma_{\mathcal{C},h})$-semirun but also to have same source and target configurations as the original semirun (and to be a min-rising and max-falling $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embedding). Hence, simply lowering all transitions with a large counter effect as described above is not enough to prove the result as the following example illustrates. Let us assume an N-semirun π containing precisely one transition τ whose absolute counter effect is N, say $\pi_j = +p$ for some position j. That is,

$$\pi \quad = \quad q_0(z_0) \xrightarrow{\pi_0}_N \quad \cdots \quad q_j(z_j) \xrightarrow{+p}_N q_{j+1}(z_{j+1}) \quad \cdots \quad \xrightarrow{\pi_{n-1}}_N q_n(z_n).$$

If we replace directly this j-th transition by a transition with $\Delta(\tau') = N - \Gamma_{\mathcal{C},h}$, and, starting with the $(j+1)$-th configuration, shift all following counter values by $-\Gamma_{\mathcal{C},h}$, we indeed obtain an $(N - \Gamma_{\mathcal{C},h})$-semirun

$$q_0(z_0) \xrightarrow{\pi_0}_{N-\Gamma_{\mathcal{C},h}} \cdots \; q_j(z_j) \xrightarrow{+p}_{N-\Gamma_{\mathcal{C},h}} q_{j+1}(z_{j+1} - \Gamma_{\mathcal{C},h}) \quad \cdots \quad \xrightarrow{\pi_{n-1}}_{N-\Gamma_{\mathcal{C},h}} q_n(z_n - \Gamma_{\mathcal{C},h}).$$

However, this $(N - \Gamma_{\mathcal{C},h})$-semirun does not have the same source and target configuration as the original semirun, as the target configuration's counter value has been shifted by $-\Gamma_{\mathcal{C},h}$. Worse yet, if our initial N-semirun π were to possess several $+p$-transitions, then the accumuluated counter value shifts could potentially yield that the resulting $(N - \Gamma_{\mathcal{C},h})$-semirun is not a $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embedding of π: indeed, such a shifted semirun could contain intermediate configurations with counter values less than $B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1$.

In order to account for those transitions whose absolute counter effect is N that have already been lowered or not we will introduce the notion of hybrid semiruns, which can be seen as sequences of N-semiruns and $(N - \Gamma_{\mathcal{C},h})$-semiruns whose source and target configurations are suitably connected.

Definition 31. *A* hybrid semirun *is a sequence* $\eta = \alpha^{(0)}\beta^{(1)}\alpha^{(1)}...\beta^{(k)}\alpha^{(k)}$*, where*

- *each* $\alpha^{(i)}$ *is an* $(N - \Gamma_{\mathcal{C},h})$*-semirun (possibly empty) of the form*

$$\alpha^{(i)} \quad = \quad p_0(y_0) \xrightarrow{\alpha_0^{(i)}}_{N-\Gamma_{\mathcal{C},h}} p_1(y_1) \quad \cdots \quad \xrightarrow{\alpha_{m_i}^{(i)}}_{N-\Gamma_{\mathcal{C},h}} p_{m_i}(y_{m_i}),$$

- *each* $\beta^{(i)}$ *is a single transition with* $|\Delta(\beta^{(i)})| = N$*,*
- *the target configuration of* $\alpha^{(i-1)}$ *is the source configuration of* $\beta^{(i)}$ *for all* $i \in [1,k]$*, and*
- *the source configuration of* $\alpha^{(i)}$ *is the target configuration of* $\beta^{(i)}$ *for all* $i \in [1,k]$*.*

We call k *the* breadth *of* η*.*

Remark 32. *In case our initial* N*-semirun* π *contains* k *transitions of absolute counter effect* N*, we observe that* π *can naturally be viewed as an initial hybrid semirun of breadth* k*.*

Several of the notions (such as counter effect, length and maximum) that we have defined for runs and semiruns can naturally be extended to hybrid semiruns. As expected, the projection $\phi(\eta)$ is defined as $\phi(\eta) = \phi(\alpha^{(0)})\phi(\beta^{(1)})\phi(\alpha^{(1)})\cdots\phi(\beta^{(k)})\phi(\alpha^{(k)})$. We moreover introduce the particular projection $\phi_\restriction$ of ϕ restricted to the $\alpha^{(i)}$, i.e. $\phi_\restriction(\eta) = \phi(\alpha^{(0)})\phi(\alpha^{(1)})\cdots\phi(\alpha^{(k)})$.

Moreover, we view the $\alpha^{(i)}$ themselves as sequences (not as atomic objects) of length m_i and the $\beta^{(i)}$ as sequences of length one. Using this convention, the notions of prefixes, infixes and suffixes are as expected. More importantly, we extend naturally the notion of (max-falling and min-rising) ℓ-embedding to hybrid semiruns as in Definition 26 when treating them as such sequences.

We prove the Hill and Valley Lemma (Lemma 30) in Section 4.5.1. We summarize important further consequences of the proof in Section 4.5.

4.5.1 Proof of the Hill and Valley Lemma

Let us fix any N-semirun

$$\pi \quad = \quad q_0(z_0) \xrightarrow{\pi_0}_{N} q_1(z_1) \xrightarrow{\pi_1}_{N} \quad \cdots \quad \xrightarrow{\pi_{n-1}}_{N} q_n(z_n)$$

from $q_0(z_0)$ to $q_n(z_n)$ with $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$ and $N > M_{\mathcal{C},h}$. Let us moreover assume that π is a B-hill for some $B \in \mathbb{N}$. The case when π is a B-valley can be proven analogously.

For reasons of simplicity we separate the proof into two cases, namely if there is a $+p$-transition or $-p$-transition whose source and target configurations have counter values that are both at most $B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h}$ or not. Section 4.5.1 deals with the latter case, Section 4.5 with the former. It is worth mentioning that Section 4.5 depends on Section 4.5.1.

π does not contain any $\pm p$-transition whose source and target configuration both have counter value at most $B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h}$

In the following let us denote by $\mathcal{L}$ the *critical level*, i.e. the constant

$$\mathcal{L} = B + \Gamma_{\mathcal{C},h}.$$

Moreover, for a hybrid semirun $\eta = \alpha^{(0)}\beta^{(1)}\alpha^{(1)}...\beta^{(k)}\alpha^{(k)}$, for every $\beta^{(j)}$ that is an unlowered $+p$-transition, we define the *critical descending infix with respect to* $\beta^{(j)}$ as the shortest prefix (when viewed as a sequence, as mentioned above) of $\alpha^{(j)}\beta^{(j+1)}\alpha^{(j+1)}...\beta^{(k)}\alpha^{(k)}$ that ends in a configuration with counter value at most $\mathcal{L}$. In particular, this critical descending infix could possibly end in a configuration inside some (strict prefix of) $\alpha^{(i)}$, where $i \in [j,k]$. Dually, for every $\beta^{(j)}$ that is an unlowered $-p$-transition, we define the *critical ascending infix with respect to* $\beta^{(j)}$ as the shortest suffix of $\alpha^{(0)}\beta^{(1)}\cdots\alpha^{(j-1)}$ that starts in a configuration with counter value at most $\mathcal{L}$. The following remark is central.

Remark 33. *For a hybrid semirun $\eta = \alpha^{(0)}\beta^{(1)}\alpha^{(1)}\dots\beta^{(k)}\alpha^{(k)}$, if some unlowered $-p$-transition (resp. $+p$-transition) $\beta^{(j)}$ appears in the critical descending infix (resp. critical ascending infix) of some unlowered $+p$-transition (resp. $-p$-transition) $\beta^{(i)}$, then so does $\beta^{(i)}$ appear in the critical ascending infix (resp. critical descending infix) of $\beta^{(j)}$.*

Viewing our initial semirun π as a hybrid semirun, we will now introduce two phases that successively lower unlowered $+p$-transitions and unlowered $-p$-transitions yielding hybrid semiruns that retain an approximation invariant (Definition 34).

In phase one, we are interested in unlowered $+p$-transitions. We want to progressively lower these, going from right to left. Moreover, we want to inspect the critical descending infix in order to obtain successive min-rising and max-falling embeddings with the same source and target configuration. In case the rightmost unlowered $+p$-transition has the property that its critical descending infix contains some unlowered $-p$-transition we lower the leftmost such directly, together with the $+p$-transition. Otherwise, we want to make use of the Bracket Lemma (Lemma 25) and the Depumping Lemma (Lemma 24) in order to retain some nice bracketing properties.

Having successively lowered all unlowered $+p$-transitions in phase one, we finally lower the remaining unlowered $-p$-transitions in phase two. For these we take their critical ascending infix and their $\varphi_\restriction$-projection into account, again yielding some carefully chosen bracketing property.

The following definition formalizes the above-mentioned bracketing property.

Definition 34. *A hybrid semirun η* approximates *π with respect to level $\ell \in \mathbb{Z}$ if*

1. *$\eta = \alpha^{(0)}\beta^{(1)}\alpha^{(1)}\dots\beta^{(k)}\alpha^{(k)}$ is a hybrid semirun of some breadth k,*
2. *π can be factorized as $\pi = \chi^{(0)}\zeta^{(1)}\chi^{(1)}\dots\zeta^{(k)}\chi^{(k)}$, where the $\zeta^{(i)}$ are transitions with operation either $+p$ or $-p$,*
3. *η is a min-rising and max-falling ℓ-embedding of π,*
4. *$\alpha^{(i)}$ is a max-falling ℓ-embedding of $\chi^{(i)}$ for all $i \in [0,k]$ with the same source and target configuration as $\chi^{(i)}$,*
5. *every prefix of $\phi_\restriction(\gamma^{(i)})$ contains at least as many occurrences of $[$ as of $]$, where $\gamma^{(i)}$ is the critical descending infix of $\beta^{(i)}$ for all $i \in [1,k]$ for which $\beta^{(i)}$ has operation $+p$, and*
6. *every suffix of $\phi_\restriction(\gamma^{(i)})$ contains at least as many occurrences of $]$ as of $[$, where $\gamma^{(i)}$ is the critical ascending infix of $\beta^{(i)}$ for all $i \in [1,k]$ for which $\beta^{(i)}$ has operation $-p$.*

By completing phase one and then phase two we will show the existence of a hybrid semirun that approximates π with respect to level B and does not contain any unlowered $+p$-transition nor any unlowered $-p$-transition (and is hence an $(N - \Gamma_{\mathcal{C},h})$-semirun). Observe first that by Point 4 any such hybrid semirun η has the same source and target configuration as π. Secondly, any such η is in particular a min-rising and max-falling $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embedding of π since π is assumed to be a B-hill. Thus, the lemma follows. We will obtain the desired $(N - \Gamma_{\mathcal{C},h})$-semirun and variants thereof by first systematically lowering $+p$-transitions from the rightmost to the leftmost in phase one and secondly systematically lowering the possibly remaining $-p$-transitions from the leftmost to the rightmost in phase two. We denote such a process — whose details are given below — by the so-called $(+p,-p)$-*lowering process*. As mentioned in Remark 37 we will also define a dual variant, namely the $(-p,+p)$-*lowering process*: here phase one will consist of systematically lowering the $-p$-transitions from the leftmost to the rightmost, whereas phase two will systematically lower the possibly remaining $+p$-transitions from the rightmost to the leftmost.

Remark 37 finally discusses a variant of a $(+p,-p)$-lowering process (resp. $(-p,+p)$-process) which ends in a hybrid semirun that contains precisely one unlowered transition.

Let us discuss the $(+p,-p)$-lowering process in detail.

Phase one of the $(+p,-p)$-lowering process: Lowering $+p$-transitions

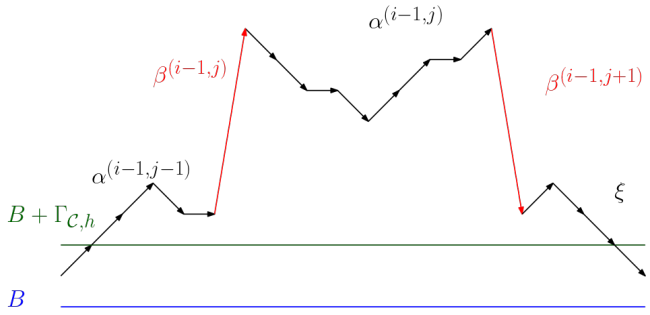

Figure 4.6: Illustration of phase one case 1, i.e. the unlowered $+p$-transition $\beta^{(i-1,j)}$ can be lowered by lowering it with the leftmost unlowered $-p$-transition on its critical descending infix, i.e. $\beta^{(i-1,j+1)}$.

We can view our initial N-semirun π as a hybrid semirun $\eta^{(0)}$ of breadth k_0, i.e.

$$\eta^{(0)} \quad = \quad \alpha^{(0,0)}\beta^{(0,1)}\alpha^{(0,1)}\ldots\beta^{(0,k_0)}\alpha^{(0,k_0)}.$$

In phase one we will inductively show the existence of a sequence of hybrid semiruns $\eta^{(0)}, \eta^{(1)}, \ldots, \eta^{(r)}$, where each $\eta^{(i)}$ has breadth k_i and approximates π with respect to level B, $\eta^{(r)}$ does not contain any unlowered $+p$-transition, and $k_{i-1} > k_i$ for all $i \in [1, r]$. Let us assume that we have inductively already defined the sequence $\eta^{(0)}, \ldots, \eta^{(i-1)}$ of hybrid semiruns for some $i \geq 1$ and where $\eta^{(i-1)}$ has breadth $k_{i-1} > 0$ and approximates π with respect to level B and contains at least one unlowered $+p$-transition. Towards extending the sequence we need to show the existence of some hybrid semirun $\eta^{(i)}$ of breadth $k_i < k_{i-1}$ that approximates π with respect to level B.

Let $\eta^{(i-1)} = \alpha^{(i-1,0)}\beta^{(i-1,1)}\alpha^{(i-1,1)}\ldots\beta^{(i-1,k_{i-1})}\alpha^{(i-1,k_{i-1})}$. Let $j \in [1, k_{i-1}]$ be maximal such that $\beta^{(i-1,j)}$ is an unlowered $+p$-transition. For defining $\eta^{(i)}$ we make the following case distinction.

1. The critical descending infix with respect to the $+p$-transition $\beta^{(i-1,j)}$ contains at least one unlowered $-p$-transition. That is, the critical descending infix is of the form

$$\alpha^{(i-1,j)}\beta^{(i-1,j+1)}\alpha^{(i-1,j+1)}\ldots\beta^{(i-1,\kappa)}\xi,$$

where ξ is a prefix (possibly empty) of $\alpha^{(i-1,\kappa)}$, $\beta^{(i-1,j+1)}$ is an unlowered $-p$-transition and where $\kappa \geq j+1$. We refer to Figure 4.6 for an illustration. Our desired hybrid semirun $\eta^{(i)}$ is obtained from $\eta^{(i-1)}$ by simply lowering both $\beta^{(i-1,j)}$ and $\beta^{(i-1,j+1)}$, i.e. replacing $\beta^{(i-1,j)}$ by $\widehat{\beta^{(i-1,j)}}$ satisfying $\Delta(\widehat{\beta^{(i-1,j)}}) = N - \Gamma_{\mathcal{C},h}$ and replacing $\beta^{(i-1,j+1)}$ by a suitable $\widehat{\beta^{(i-1,j+1)}}$ satisfying $\Delta(\widehat{\beta^{(i-1,j+1)}}) = -N + \Gamma_{\mathcal{C},h}$ and moreover suitably shifting the part after $\widehat{\beta^{(i-1,j)}}$ and until (including) $\widehat{\beta^{(i-1,j+1)}}$ by $-\Gamma_{\mathcal{C},h}$. More precisely, the part $\alpha^{(i,j-1)}$ in $\eta^{(i)}$ is chosen to be of the form

$$\alpha^{(i,j-1)} = \alpha^{(i-1,j-1)}\widehat{\beta^{(i-1,j)}}\left(\alpha^{(i-1,j)} - \Gamma_{\mathcal{C},h}\right)\left(\beta^{\widehat{(i-1,j+1)}} - \Gamma_{\mathcal{C}}\right)\alpha^{(i-1,j+1)}.$$

Moreover, observe that $\alpha^{(i,j-1)}$ and the infix $\alpha^{(i-1,j-1)}\beta^{(i-1,j)}\alpha^{(i-1,j)}\beta^{(i-1,j+1)}\alpha^{(i-1,j+1)}$ of $\eta^{(i-1)}$ connect the same source and target configurations. Thus, it easily follows that $\eta^{(i)}$ also approximates π with respect to level B. Finally, observe that the breadth of $\eta^{(i)}$ equals $k_{i-1} - 2$.

2. The critical descending infix with respect to the $+p$-transition $\beta^{(i-1,j)}$ does not contain any unlowered $-p$-transition. It follows that the critical descending infix with respect to $\beta^{(i-1,j)}$ is a non-empty prefix ξ of $\alpha^{(i-1,j)}$. We refer to Figure 4.7 for an illustration. Recall that $\eta^{(i-1)}$

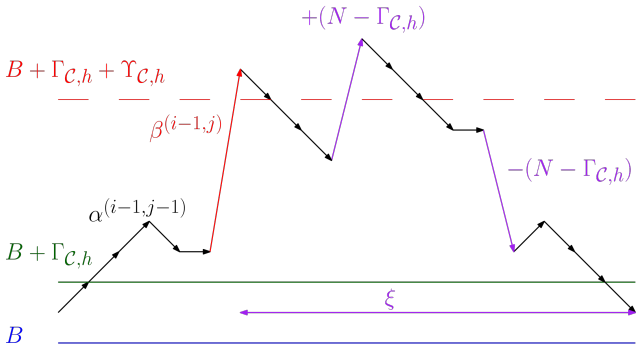

Figure 4.7: Illustration of phase one case 2, i.e. the suffix of the to be lowered $+p$ transition $\beta^{(i-1,j)}$ does not contain any unlowered $-p$-transition, i.e. any transition with counter effect $-N$, inside its critical descending infix.

approximates π with respect to level B. Firstly, since by assumption $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$, it follows from Point 3 of Definition 34 that $\textsc{Values}(\xi) \subseteq [0, h \cdot N]$. Secondly, from Point 5 of Definition 34 every prefix of $\phi(\alpha^{(i-1,j)})$ contains at least as many occurrences of $[$ as of $]$. Hence, the latter must also hold for every prefix of $\phi(\xi)$. Thirdly, since by the case of this subsection the target configuration of every transition with operation $+p$ in π has counter value strictly larger than $B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h}$, it follows from Points 2 and 4 of Definition 34 that the target configuration of $\beta^{(i-1,j)}$ ends in a configuration with counter value strictly larger than $B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h}$. Since ξ is the critical descending infix with respect to $\beta^{(i-1,j)}$ (in particular ending in a configuration with counter value at most $B + \Gamma_{\mathcal{C},h}$), it follows $\Delta(\xi) < -\Upsilon_{\mathcal{C},h}$. Hence one can apply Lemma 25 to the $(N - \Gamma_{\mathcal{C},h})$-semirun ξ yielding an infix $\xi[c,d]$ satisfying $\phi(\xi[c,d]) \in \Lambda_{2h}$ and $\Delta(\xi[c,d]) < -\Upsilon_{\mathcal{C},h}$. Applying Lemma 24 to $\xi[c,d]$ implies the existence of an $(N - \Gamma_{\mathcal{C},h})$-semirun $\xi' = \xi[c,d] - I_1 - I_2 \cdots - I_s$ satisfying $\Delta(\xi') = \Delta(\xi[c,d]) + \Gamma_{\mathcal{C},h}$ and where $I_1, \ldots, I_s$ are pairwise disjoint intervals of positions in $\xi[c,d]$ such that moreover $\phi(\xi[c,d][I_t]) \in \Lambda_{4h}$ and $\Delta(\xi[c,d][I_t]) < 0$ for all $t \in [1,s]$. Assume that $\xi = \xi[0,m]$ consisted of m transitions; thus in particular $c, d \in [0,m]$. By combining the above properties it immediately follows that

$$\xi'' = \xi[0,c]\xi'\,(\xi[d,m] + \Gamma_{\mathcal{C},h})$$

is an $(N - \Gamma_{\mathcal{C},h})$-semirun with $\Delta(\xi'') = \Delta(\xi) + \Gamma_{\mathcal{C},h}$ and that $\xi'' - \Gamma_{\mathcal{C},h}$ is a max-falling B-embedding of ξ. We define the desired $\eta^{(i)}$ to be obtained from $\eta^{(i-1)}$ by lowering $\beta^{(i-1,j)}$ to $\widehat{\beta^{(i-1,j)}}$ satisfying $\Delta(\widehat{\beta^{(i-1,j)}}) = \Delta(\beta^{(i-1,j)}) - \Gamma_{\mathcal{C},h}$ and moreover replacing ξ by $\xi'' - \Gamma_{\mathcal{C},h}$. Observe that $\eta^{(i)}$ and $\eta^{(i-1)}$ only differ in the infix $\alpha^{(i,j-1)}$ of $\eta^{(i)}$. The latter is hence of the form

$$\alpha^{(i,j-1)} = \alpha^{(i-1,j-1)}\widehat{\beta^{(i-1,j)}}\,(\xi'' - \Gamma_{\mathcal{C},h})\,\alpha^{(i-1,j)}[m, |\alpha^{(i-1,j)}|].$$

By construction $\eta^{(i-1)}$'s infix

$$\alpha^{(i-1,j-1)}\beta^{(i-1,j)}\alpha^{(i-1,j)}$$

has the same source and target configuration as the part $\alpha^{(i,j-1)}$ of $\eta^{(i)}$. Since moreover

- $\phi(\xi[c,d][I_t]) \in \Lambda_{4h}$ contains precisely as many occurrences of $[$ as of $]$ and $\Delta(\xi[c,d][I_t]) < 0$ for each $t \in [1,s]$ and
- $\Delta(\xi'') = \Delta(\xi) + \Gamma_{\mathcal{C}}$

it follows that indeed $\eta^{(i)}$ approximates π with respect to level B. Finally, observe that the breadth of $\eta^{(i)}$ is $k_{i-1} - 1$.

Recall that in phase one we have repeatedly lowered unlowered $+p$-transitions from right to left. In doing so we have hereby possibly lowered certain $-p$-transitions. The final hybrid semirun $\eta^{(r)}$ of phase one notably does not contain any unlowered $+p$-transition. However, $\eta^{(r)}$ may still contain unlowered $-p$-transitions. Lowering these will be subject of phase two. Yet, these unlowered $-p$-transitions will be lowered rather from leftmost to rightmost (instead of from rightmost to leftmost as in phase one).

Phase two of the $(+p,-p)$-lowering process: Lowering $-p$-transitions that remain after phase one

Recall that $\mathcal{L} = B + \Gamma_{\mathcal{C},h}$ denotes our critical level. Also recall that $\eta^{(r)}$ is the final hybrid semirun in the sequence $\eta^{(0)},\ldots,\eta^{(r)}$ of phase one and approximates π with respect to level B. Note that by construction $\eta^{(r)}$ does not contain any unlowered $+p$-transition. That is, all unlowered transitions of $\eta^{(r)}$ have operation $-p$ and there are as many of them as the breadth of $\eta^{(r)}$. Setting $\eta^{(0)'} = \eta^{(r)}$, phase two consists in showing the existence of a sequence of hybrid semiruns $\eta^{(1)'},\ldots,\eta^{(t)'}$ all of which do not contain any unlowered $+p$-transition and in which each $\eta^{(i)'}$ has breadth k'_i satisfying $k'_i < k'_{i-1}$, where each $\eta^{(i)'}$ approximates π with respect to level B, and finally $\eta^{(t)'}$ is of breadth 0 (and is therefore already an $(N - \Gamma_{\mathcal{C},h})$-semirun).

Let us inductively assume that we have already defined the sequence $\eta^{(0)'},\ldots,\eta^{(i-1)'}$ for some $i \geq 1$ and that the breadth k'_{i-1} of $\eta^{(i-1)'}$ satisfies $k'_{i-1} > 0$.

Let $\eta^{(i-1)'} = \alpha^{(i-1,0)'}\beta^{(i-1,1)'}\alpha^{(i-1,1)'}\ldots\beta^{(i-1,k_{i-1})'}\alpha^{(i-1,k_{i-1})'}$. There is only one possible case for this phase since the critical ascending infix with respect to the leftmost unlowered $-p$-transition $\beta^{(i-1,1)'}$ does not contain any unlowered $+p$-transition since $\eta^{(i-1)'}$ does not. The construction of $\eta^{(i)'}$, as well as the proof that $\eta^{(i)'}$ approximates π with respect to level B, is completely dual to the proof of the second case of phase one and therefore omitted.

Example 35. *Figure 4.8 illustrates an example of an application of the $(+p,-p)$-lowering process. The topmost figure on the left is the starting hybrid semirun π. We begin the process by lowering the two unlowered $+p$-transitions each by compensating them with a $-p$-transition, then we enter phase two with one unlowered $-p$-transition remaining, which we lower and compensate by shifting and cutting out portions inside the critical ascending infix by applying the Depumping Lemma.*

Remark 36. *In case π is a B-valley instead of a B-hill there is a dual variant of the $(+p,-p)$-lowering process. The critical level would be adjusted to $\mathcal{L} = B - \Gamma_{\mathcal{C},h}$, for unlowered $+p$-transitions one would define the critical descending infix to be the shortest suffix of $\alpha^{(0)}\beta^{(1)}\ldots\alpha^{(j-1)}$ that starts in a configuration with counter value at least $\mathcal{L}$, whereas for unlowered $-p$-transitions one would define the critical ascending infix to be the shortest prefix of $\alpha^{(j)}\beta^{(j+1)}\alpha^{(j+1)}\ldots\beta^{(k)}\alpha^{(k)}$ that ends in a configuration with counter value at least $\mathcal{L}$. The definition when a hybrid semirun approximates π with respect to level B would be defined analogously as in Definition 34, but where in Point 4 $\alpha^{(i)}$ is rather required to be a min-rising B-embedding of $\chi^{(i)}$ with the same source and target configuration as $\chi^{(i)}$, Point 5 (resp. Point 6) of Definition 34 would rather require that every suffix (resp. every prefix) of $\phi_{\restriction}(\gamma^{(i)})$ contains at least as many occurrences of* [*(resp.*]*) as of*] *(resp.* [*).*

Remark 37. *Consider the following variants of the $(+p,-p)$-lowering process for our B-hill π (dual variants can be formulated in the case when π is B-valley):*

1. *Consider the dual* $(-p,+p)$-lowering process*: In phase one we lower the $-p$-transitions from the leftmost to the rightmost and in phase two lower the $+p$-transitions from the rightmost to the leftmost. That is, such a $(-p,+p)$-lowering process produces a sequence of hybrid semiruns*

$$\eta^{(0)},\eta^{(1)},\ldots,\eta^{(s)},\eta^{(0)'},\eta^{(1)'},\ldots,\eta^{(t)'},$$

that all approximate π with respect to level B where $\eta^{(0)} = \pi$, $\eta^{(i)}$ is obtained from $\eta^{(i-1)}$ by lowering the leftmost unlowered $-p$-transition of $\eta^{(i)}$, $\eta^{(0)'} = \eta^{(s)}$, $\eta^{(i+1)'}$ is obtained from $\eta^{(i)'}$ by lowering the rightmost unlowered $+p$-transition, and finally $\eta^{(t)'}$ has breadth 0.

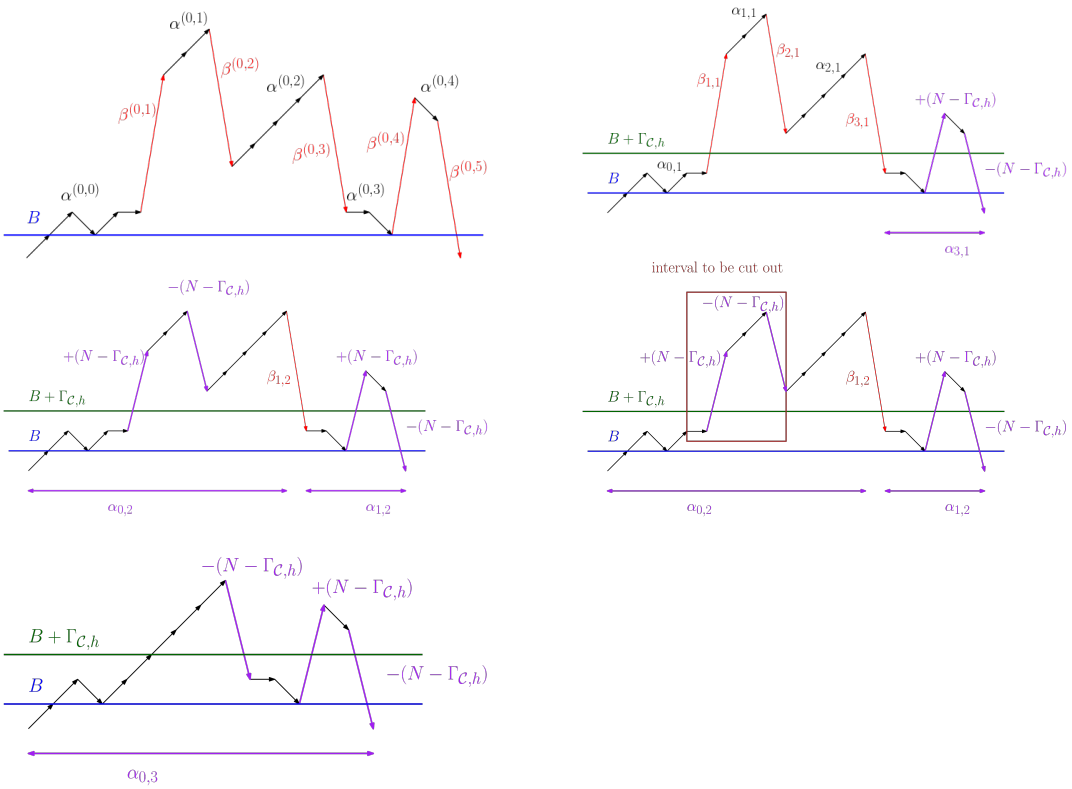

Figure 4.8: Illustration of the $(+p,-p)$-lowering process from Example 35 to be read from upper left to lower right.

2. *Consider again the sequence of hybrid semiruns*

$$\eta^{(0)}, \eta^{(1)}, \ldots, \eta^{(s)}, \eta^{(0)'}, \eta^{(1)'}, \ldots, \eta^{(t)'},$$

of the $(+p,-p)$*-process (dually* $(-p,+p)$*-process):*

(a) *If* $t > 0$*, then observe that* $\eta^{(t-1)'}$ *has breadth* 1 *and contains precisely one unlowered transition, namely an unlowered* $-p$*-transition (dually* $+p$*-transition).*

(b) *If however* $t = 0$*, then we claim that every prefix (dually suffix) of* $\phi(\eta^{(0)'}) = \phi(\eta^{(s)})$ *contains at least as many occurrences of* [*(dually occurrences of*]*) as occurrences of*] *(dually occurrences of* [*). Indeed, it follows immediately from the fact that each* $\eta^{(i)}$ *is obtained from* $\eta^{(i-1)}$ *by lowering a* $+p$*-transition (dually a* $-p$*-transition) either by shifting infixes and cutting out certain infixes* ζ' *for which* $\phi(\zeta')$ *contains as many occurrences of* [*as of*]*, or by lowering a* $+p$*-transition (dually* $-p$*-transition) together with an unlowered* $-p$*-transition (dually* $+p$*-transition) to the right (dually to the left).*

(c) *If* $\phi(\pi)$ *a priori contains strictly more occurrences of*] *than of* [*one can — by applying the* $(+p,-p)$*-lowering process — obtain a sequence of hybrid semiruns*

$$\eta^{(0)}, \eta^{(1)}, \ldots, \eta^{(s)}, \eta^{(0)'}, \eta^{(1)'}, \ldots, \eta^{(t)'},$$

where $t > 0$*, all* $\eta^{(i)}$ *and* $\eta^{(j)'}$ *approximate* π *with respect to level* B *(and are therefore, as remarked above by bearing in mind that* π *is* B*-hill, in particular both min-rising and max-falling* $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$*-embeddings of* π *with the same source and target configuration as* π*) and where the breadth of* $\eta^{(t-1)'}$ *is* 1*. Dually, if* $\phi(\pi)$ *contains strictly more occurrences of* [*than of*] *one can — by applying the* $(-p,+p)$*-lowering process — obtain a sequence of hybrid semiruns*

$$\eta^{(0)}, \eta^{(1)}, \ldots, \eta^{(s)}, \eta^{(0)'}, \eta^{(1)'}, \ldots, \eta^{(t)'},$$

where $t > 0$*, all* $\eta^{(i)}$ *and* $\eta^{(j)'}$ *approximate* π *with respect to level* B *(and are therefore both min-rising and max-falling* $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$*-embeddings of* π *with the same source and target configuration as* π*) and where the breadth of* $\eta^{(t-1)'}$ *is* 1*.*

π contains a $\pm p$-transition whose source and target configuration both have counter value at most $B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h}$

The presence of a $+p$-transition (resp. $-p$-transition) $q_i(z_i) \xrightarrow{\pi_i}_N q_{i+1}(z_{i+1})$ for which we have $\max\{z_i, z_{i+1}\} \leq B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h}$ implies $z_{i+1} - (B + \Gamma_{\mathcal{C},h}) \leq \Upsilon_{\mathcal{C},h}$ (resp. $z_i - (B + \Gamma_{\mathcal{C},h}) \leq \Upsilon_{\mathcal{C},h}$), so the core problem is that in both cases it is not possible to apply the Bracket Lemma (Lemma 25) in the critical descending (resp. ascending) infix of such an unlowered transition. We thus have to find another way to compensate for lowering such transitions.

We next claim that firstly, any $+p$-transition whose configurations both have a counter value at most $B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h}$ must be the first transition of π and secondly, any $-p$-transition with the same property must be the last transition of π. Indeed, every $+p$-transition $q_i(z_i) \xrightarrow{\pi_i}_N q_{i+1}(z_{i+1})$ that is not the first transition (i.e. $i > 0$) satisfies $z_i \geq B$ as π is a B-hill. As a consequence, we have $z_{i+1} \geq B + N > B + M_{\mathcal{C},h} > B + \Gamma_{\mathcal{C},h} + \Upsilon_{\mathcal{C},h}$, where the last inequality follows from $M_{\mathcal{C},h}$'s definition on page 33. Dually, if there exists a $-p$-transition $q_i(z_i) \xrightarrow{\pi_i}_N q_{i+1}(z_{i+1})$ with $z_i \leq B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h}$ it must be the last transition $q_{n-1}(z_{n-1}) \xrightarrow{\pi_{n-1}}_N q_n(z_n)$ of π.

To finalize the proof it thus suffices to distinguish whether both the first transition of π is a $+p$-transition with counter values at most $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$ and the last transition of π is a $-p$-transition with counter values at most $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$, or this holds for precisely one of them. We thus distinguish these two cases, however in opposite order.

Case 1. The first transition $q_0(z_0) \xrightarrow{\pi_0}_N q_1(z_1)$ *is a* $+p$*-transition with counter values at most* $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$ *and the last transition* $q_{n-1}(z_{n-1}) \xrightarrow{\pi_{n-1}}_N q_{n-1}(z_{n-1})$ *is not a* $-p$*-transition with counter values at most* $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$*, or the first transition* $q_0(z_0) \xrightarrow{\pi_0}_N q_1(z_1)$ *is not a* $+p$*-transition with counter values at most* $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$ *and the last transition* $q_{n-1}(z_{n-1}) \xrightarrow{\pi_{n-1}}_N q_{n-1}(z_{n-1})$ *is a* $-p$*-transition with counter values at most* $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$.

We only treat the case when the first transition $q_0(z_0) \xrightarrow{\pi_0}_N q_1(z_1)$ is a $+p$-transition with counter values at most $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$ and the last transition $q_{n-1}(z_{n-1}) \xrightarrow{\pi_{n-1}}_N q_{n-1}(z_{n-1})$ is not a $-p$-transition with counter values at most $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$, since the opposite case can be proven analogously.

Starting with $\eta^{(0)} = \pi$ we apply phase one of the $(+p, -p)$-lowering process to π yielding a sequence of hybrid semiruns that all approximate π with respect to level B (and thus in particular — bearing in mind that π is B-hill — approximates π with respect to level $B - \Gamma_{\mathcal{C},h} - 1$)

$$\eta^{(0)}, \eta^{(1)}, \ldots, \eta^{(s-1)}$$

in which (as above) $\eta^{(i)}$ is obtained from $\eta^{(i-1)}$ by lowering the rightmost unlowered $+p$ of $\eta^{(i-1)}$ however only until reaching the hybrid semirun $\eta^{(s-1)}$ that contains precisely one unlowered $+p$-transition, namely the first transition $q_0(z_0) \xrightarrow{\pi_0}_N q_1(z_1)$ of π, which has counter values at most $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$ by assumption. It is important but straightforward to verify that despite the case we are in, it holds that $\eta^{(i)}$ approximates π with respect to level B (and also — bearing in mind that π is a B-hill — with respect to level $B - \Gamma_{\mathcal{C},h} - 1$) for all $i \in [1, s-1]$.

Next, we will define a sequence of hybrid semiruns $\eta^{(s)} = \eta^{(0)'}, \eta^{(1)'}, \ldots, \eta^{(t)'}$ in which $\eta^{(t)'}$ will be the desired $(N - \Gamma_{\mathcal{C},h})$-semirun as required by the lemma. For first defining $\eta^{(s)} = \eta^{(0)'}$ we make a case distinction for lowering the only $+p$-transition $q_0(z_0) \xrightarrow{\pi_0}_N q_1(z_1)$ of $\eta^{(s-1)}$, which happens to have counter values at most $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$ by assumption. For this assume $\eta^{(s-1)}$ has the following form

$$\eta^{(s-1)} = \alpha^{(s-1,0)} \beta^{(s-1,1)} \alpha^{(s-1,1)} \ldots \beta^{(s-1,k_{s-1})} \alpha^{(s-1,k_{s-1})},$$

where we recall that $\beta^{(s-1,1)}$ equals $q_0(z_0) \xrightarrow{\pi_0}_N q_1(z_1)$. Observe that the critical descending infix of $\beta^{(s-1,1)}$ could possibly be empty, for instance if $z_1 \leq B + \Gamma_{\mathcal{C},h}$. We now make the following case distinction.

- In case the critical descending infix of $\beta^{(s-1,1)}$ contains an unlowered $-p$-transition we define $\eta^{(s)}$ to be obtained from $\eta^{(s-1)}$ by lowering $\beta^{(s-1,1)}$ with the leftmost unlowered $-p$-transition inside the critical descending infix as above. Thus, $\eta^{(s)}$ no longer contains any unlowered $+p$-transition. It is again straightforward to verify that $\eta^{(s)}$ approximates π with respect to level B. Setting $\eta^{(0)'} = \eta^{(s)}$ we then construct the sequence $\eta^{(s)} = \eta^{(0)'}, \eta^{(1)'}, \ldots, \eta^{(t)'}$ as usual, i.e. each $\eta^{(i)'}$ approximates π with respect to level B and is obtained from $\eta^{(i-1)'}$ by lowering the leftmost unlowered $-p$-transition and where eventually the breadth of $\eta^{(t)'}$ is 0. Thus, as desired, the final $\eta^{(t)'}$ is an $(N - \Gamma_{\mathcal{C},h})$-semirun that is a min-rising and max-falling B-embedding of π that has the same source and target configuration as π. Since $\eta^{(t)'}$ has the same source and target configuration as π it follows that $\eta^{(t)'}$ is also a min-rising and max-falling $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embedding of π as required by the lemma.

- In case the critical descending infix of $\beta^{(s-1,1)}$ does not contain any unlowered $-p$-transition, we consider the shortest prefix ζ of the remaining suffix

  $$\alpha^{(s-1,1)} \ldots \beta^{(s-1,k_{s-1})} \alpha^{(s-1,k_{s-1})}$$

  that ends in a configuration with counter value at most

  $$\mathcal{L}' = z_1 - \Upsilon_{\mathcal{C},h} - 1$$

  (where we recall that as above each $\alpha^{(i,j)}$ is viewed as a sequence of transitions). Indeed, we claim that ζ exists and moreover satisfies $\Delta(\zeta) < -\Upsilon_{\mathcal{C},h}$. Firstly, as π is a B-hill by

assumption, we have that $z_1 - z_n > \Upsilon_{\mathcal{C},h}$. Secondly, since $\eta^{(s-1)}$ approximates π with respect to level B we have that $\eta^{(s-1)}$ ends in a configuration with counter value z_n. Thus, $\Delta(\alpha^{(s-1,1)}...\beta^{(s-1,k_{s-1})}\alpha^{(s-1,k_{s-1})}) = z_n - z_1 < -\Upsilon_{\mathcal{C},h}$ which implies that the prefix ζ exists and satisfies $\Delta(\zeta) < -\Upsilon_{\mathcal{C},h}$. We make the following final case distinction.

- In case ζ contains an unlowered $-p$-transition, it must contain the leftmost unlowered $-p$-transition, namely $\beta^{(s-1,2)}$. Similar as for the critical descending infix, we define $\eta^{(s)}$ to be obtained from $\eta^{(s-1)}$ by lowering $\beta^{(s-1,1)}$ together with $\beta^{(s-1,2)}$. Here it is important to note that $\eta^{(s)}$ is not necessarily a $(B - \Gamma_{\mathcal{C},h})$-embedding of π since we cannot rule out the existence of configurations appearing in $\alpha^{(s-1,1)}$ that have counter value B. Since $\eta^{(s-1)}$ was a B-embedding of the B-hill π with the same source and target configuration it follows however that $\eta^{(s)}$ is a $(B - \Gamma_{\mathcal{C},h} - 1)$-embedding of π. Hence, $\eta^{(s)}$ approximates π with respect to level $B - \Gamma_{\mathcal{C},h} - 1$. Thus, $\eta^{(s)}$ no longer contains any unlowered $+p$-transitions, however, possibly contains unlowered $-p$-transitions. Recalling that $\eta^{(0)'} = \eta^{(s)}$ we define each of the remaining $\eta^{(i)'}$ to be obtained from $\eta^{(i-1)'}$ as usual but by retaining that each $\eta^{(i)'}$ approximates π with respect to level $B - \Gamma_{\mathcal{C},h} - 1$ (instead of level B). By construction, $\eta^{(0)'}$ has breadth 0 and thus is an $(N - \Gamma_{\mathcal{C},h})$-semirun that is a min-rising and max-falling $(B - \Gamma_{\mathcal{C},h} - 1)$-embedding and hence — bearing in mind that π is a B-hill — in particular a min-rising and max-falling $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embedding of π with the same source and target configuration as π.
- In case ζ does not contain any unlowered $-p$-transition it follows that ζ is a prefix of $\alpha^{(s-1,1)}$, thus contains neither unlowered $+p$-transitions nor unlowered $-p$-transitions but possibly lowered ones. By an analogous reasoning as Point 2 of Remark 37 every occurrence of a lowered $-p$-transition in $\alpha^{(s-1,1)}$ is preceded by a unique corresponding lowered $+p$-transition again in $\alpha^{(s-1,1)}$. Thus, every prefix of $\phi(\zeta)$ contains at least as many occurrences of [as of]. Recalling that $\Delta(\zeta) < -\Upsilon_{\mathcal{C},h}$ we can hence apply the Bracket Lemma (Lemma 25) and the Depumping Lemma (Lemma 24) to ζ as in phase one. The final $\eta^{(s)}$ is obtained from $\eta^{(s-1)}$ by suitably shifting subsemiruns and cutting out certain subsemiruns whose ϕ-projection contains the same number of occurrences of [as of]. Similar as argued in the previous point it follows that $\eta^{(s)}$ approximates π with respect to level $B - \Gamma_{\mathcal{C},h} - 1$. Setting again $\eta^{(0)'} = \eta^{(s)}$ we define the sequence of hybrid semiruns $\eta^{(0)'}, \eta^{(1)'}, \ldots, \eta^{(t)'}$ that all approximate π with respect to level $B - \Gamma_{\mathcal{C},h} - 1$ analogously as done in the previous point. Again $\eta^{(t)'}$ is an $(N - \Gamma_{\mathcal{C},h})$-semirun that is a min-rising and max-falling $(B - \Gamma_{\mathcal{C},h} - 1)$-embedding and hence in particular a min-rising and max-falling $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embedding of π with the same source and target configuration as π.

Remark 38. *Our case (where the first transition of our B-hill is a $+p$-transition with counter values at most $B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h}$ and the last transition is not a $-p$-transition with counter values at most $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$) allows the following "penultimate" process variants for our B-hill π (dual variants can be formulated in the case when π is B-valley):*

1. *The adjusted process here in Case 1 bears similar properties to those of the $(+p, -p)$-lowering process seen in Remark 37. Specifically, if $\phi(\pi)$ contains strictly more occurrences of] than of [one can obtain a sequence of hybrid semiruns*

$$\eta^{(0)}, \eta^{(1)}, \ldots, \eta^{(s)}, \eta^{(0)'}, \eta^{(1)'}, \ldots, \eta^{(t-1)'},$$

where all $\eta^{(i)}$ and $\eta^{(j)'}$ approximate π with respect to level $B - \Gamma_{\mathcal{C},h} - 1$ (and are therefore — bearing in mind that π is a B-hill — in particular min-rising and max-falling $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embeddings of π with the same source and target configuration as π) and where $\eta^{(t-1)'}$ has breadth 1 and contains precisely one unlowered $-p$-transition.

2. *Dually, the $(-p, +p)$-lowering process mentioned in Remark 37, when applied to Case 1, is such that if $\phi(\pi)$ contains strictly more occurrences of [than of] one can obtain a sequence*

of hybrid semiruns

$$\eta^{(0)}, \eta^{(1)}, \ldots, \eta^{(s)}, \eta^{(0)'}, \eta^{(1)'}, \ldots, \eta^{(t-1)'},$$

where all $\eta^{(i)}$ *and* $\eta^{(j)'}$ *approximate* π *with respect to level* $B - \Gamma_{\mathcal{C},h} - 1$ *(and are therefore — bearing in mind that* π *is a* B*-hill — in particular min-rising and max-falling* $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$*-embeddings of* π *with the same source and target configuration as* π*) and where* $\eta^{(t-1)'}$ *has breadth* 1 *and contains precisely one unlowered* $+p$*-transition.*

Case 2. The first transition $q_0(z_0) \xrightarrow{\pi_0}_N q_1(z_1)$ *is a* $+p$*-transition with counter values at most* $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$ *and the last transition* $q_{n-1}(z_{n-1}) \xrightarrow{\pi_{n-1}}_N q_{n-1}(z_{n-1})$ *is a* $-p$*-transition with counter values at most* $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$.

By our case we have that $z_0, z_n \leq B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h} - N \leq B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h} - M_{\mathcal{C},h} < B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1$, where the last inequality follows the definition of our constants on page 33.

Since π is a B-hill π is also a $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-hill. Moreover, obviously there are no $+p$-transitions nor $-p$-transitions in π whose source and target configuration both have counter value at most $B - 1$. Phrased differently, setting $B' = B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1$, we view π as a B'-hill that does not contain any $+p$-transitions nor $-p$-transitions whose source and target configuration have a counter value at most $(B' + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h})$. We can hence apply the $(+p, -p)$-lowering process to π as described in the case of in Section 4.5.1 for B' instead of B, thus yielding the sequence $\eta^{(0)}, \eta^{(1)}, \ldots, \eta^{(s)}$ and $\eta^{(0)'}, \eta^{(1)'}, \ldots \eta^{(t)'}$ of hybrid semiruns that approximate π with respect to level B' and are therefore min-rising and max-falling B'-embeddings of π: note that we use the fact that they are are indeed B'-embeddings as the construction in Section 4.5.1 guarantees rather than that they are $(B' - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embeddings. The final $\eta^{(t)'}$ is of breadth 0 and is hence a min-rising and max-falling $(N - \Gamma_{\mathcal{C},h})$-semirun that is a $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embedding of π with the same source and target configuration as π, as required by the lemma.

The Hill and Valley Lemma, dependent on the number of occurrences $+p$-transitions as of $-p$-transitions

A closer look at the proof of Lemma 30 reveals that majority of occurrences of $+p$-transitions (resp. $-p$-transitions) implies the respective majority is preserved in the resulting $(N - \Gamma_{\mathcal{C},h})$-semirun.

Remark 39. *The resulting* $\eta^{(t)'}$ *obtained from the* B*-hill (resp.* B*-valley)* π *satisfies the following.*

- *If* $\phi(\pi)$ *contains at least as many occurrences of* [*as of*]*, then so does the resulting* $(N - \Gamma_{\mathcal{C},h})$*-semirun* $\phi(\eta^{(t)'})$ *satisfying Lemma 30.*
- *If* $\phi(\pi)$ *contains at least as many occurrences of*] *as of* [*, then so does the resulting* $(N - \Gamma_{\mathcal{C},h})$*-semirun* $\phi(\eta^{(t)'})$ *satisfying Lemma 30.*

The following final remark stresses the fact that when our B-hill (resp. B-valley) π contains a number of occurrences of $+p$-transitions different from the number of $-p$-transitions, the lowering processes described in the previous section yield a penultimate hybrid semirun where all but one of whose $+p$-transitions and $-p$-transitions are lowered. It is an immediate consequence of Point 2.c in Remark 37 and Remark 38.

Remark 40. *Let* π *be an* N*-semirun that is a* B*-hill (dually a* B*-valley).*

1. *If* $\phi(\pi)$ *contains strictly more occurrences of*] *than of* [*one can obtain a sequence of hybrid semiruns*

$$\eta^{(0)}, \eta^{(1)}, \ldots, \eta^{(s)}, \eta^{(0)'}, \eta^{(1)'}, \ldots, \eta^{(t-1)'},$$

in which $\eta^{(t-1)'}$ *is a min-rising and max-falling* $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$*-embedding (dually* $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h} + 1)$*-embedding) of* π *with the same source and target configuration as* π *and where* $\eta^{(t-1)'}$ *has breadth* 1 *and contains precisely one unlowered* $-p$*-transition.*

2. *Analogously, if $\phi(\pi)$ contains strictly more occurrences of* [*than of*] *one can obtain a sequence of hybrid semiruns*

$$\eta^{(0)}, \eta^{(1)}, \ldots, \eta^{(s)}, \eta^{(0')}, \eta^{(1)'}, \ldots, \eta^{(t-1)'},$$

in which $\eta^{(t-1)'}$ is a min-rising and max-falling $(B - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embedding (dually $(B + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h} + 1)$-embedding) of π with the same source and target configuration as π and where $\eta^{(t-1)'}$ has breadth 1 and contains precisely one unlowered +p-transition.

4.6 The 5/6-Lemma

In this Section, we introduce the 5/6-Lemma (Lemma 41), stating that any N-semirun with counter effect smaller than $5/6 \cdot N$ can be turned into an $(N - \Gamma_{\mathcal{C},h})$-semirun that is moreover an ℓ-embedding for all ℓ that are in distance at most $5/6 \cdot N$ from the counter values of the source and target configuration. It will be the main technical ingredient in the proof of the Small Parameter Theorem (Theorem 21). This section is devoted to proving the lemma, hereby making extensive use of the Hill and Valley Lemma (Lemma 30), the Depumping Lemma (Lemma 24), and the Bracket Lemma (Lemma 25) introduced in previous sections.

Recall that we have fixed a PTOCA $\mathcal{C} = (Q, P, R, q_{init}, F)$ with $P = \{p\}$, $h > 0$, along with the constants $Z_{\mathcal{C}}, \Gamma_{\mathcal{C},h}, \Upsilon_{\mathcal{C},h}, M_{\mathcal{C},h}$ on page 33.

Let us first introduce the 5/6-Lemma.

Lemma 41 (5/6-Lemma)**.** *For all $N > M_{\mathcal{C},h}$ and all $\ell \in \mathbb{Z}$ and all N-semiruns π from $q_0(z_0)$ to $q_n(z_n)$ with $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$ satisfying $\max(z_0, z_n, \ell) - \min(z_0, z_n, \ell) \leq 5/6 \cdot N$ there exists an $(N - \Gamma_{\mathcal{C},h})$-semirun π' from $q_0(z_0)$ to $q_n(z_n)$ that is an ℓ-embedding of π such that $\textsc{Values}(\pi') \subseteq [\min(\pi) - \Gamma_{\mathcal{C},h}, \max(\pi) + \Gamma_{\mathcal{C},h}]$.*

Towards proving Lemma 41 let us fix

- some $N > M_{\mathcal{C},h}$,
- some $\ell \in \mathbb{Z}$,
- some N-semirun $\pi = q_0(z_0) \xrightarrow{\pi_0}_N q_1(z_1) \cdots \xrightarrow{\pi_{n-1}}_N q_n(z_n)$ from $q_0(z_0)$ to $q_n(z_n)$ satisfying $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$ and $\max(z_0, z_n, \ell) - \min(z_0, z_n, \ell) \leq 5/6 \cdot N$.

In order to prove the 5/6-Lemma we need to show the existence of some $(N - \Gamma_{\mathcal{C},h})$-semirun π' from $q_0(z_0)$ to $q_n(z_n)$ that is both an ℓ-embedding of π with $\textsc{Values}(\pi') \subseteq [\min(\pi) - \Gamma_{\mathcal{C},h}, \max(\pi) + \Gamma_{\mathcal{C},h}]$.

For this, let us define following two constants

$$B_{\min} = \min(z_0, z_n, \ell) - \Upsilon_{\mathcal{C},h} - 2\Gamma_{\mathcal{C},h} - 1 \quad \text{and} \quad B_{\max} = \max(z_0, z_n, \ell) + \Upsilon_{\mathcal{C},h} + 2\Gamma_{\mathcal{C},h} + 1$$

and observe that

$$\begin{aligned} B_{\max} - B_{\min} &= \max(z_0, z_n, \ell) - \min(z_0, z_n, \ell) + 2\Upsilon_{\mathcal{C},h} + 4\Gamma_{\mathcal{C},h} + 2 \\ &\leq 5/6 \cdot N + 2\Upsilon_{\mathcal{C},h} + 4\Gamma_{\mathcal{C},h} + 2 \\ &\leq 5/6 \cdot N + M_{\mathcal{C},h}/6 \qquad (4.4) \\ &< N, \qquad (4.5) \end{aligned}$$

where the penultimate inequality follows from the definitions of our constants on page 33.

We are particularly interested in subsemiruns of π that start and end in configurations with counter

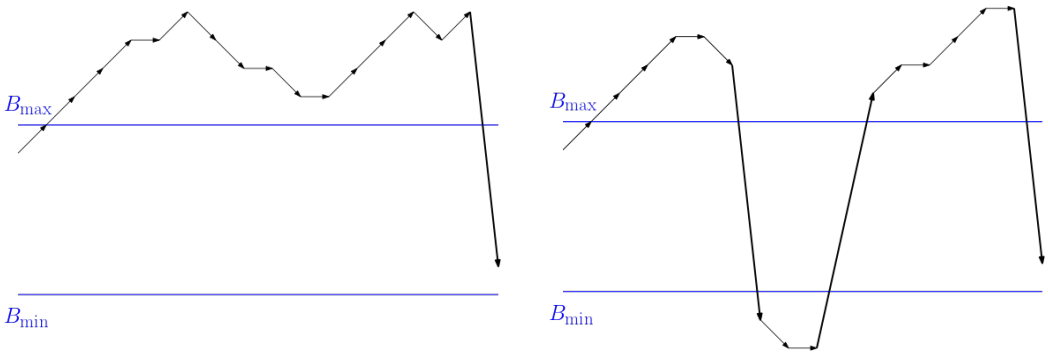

Figure 4.9: On the left, an example of a Type II subsemirun. On the right, an example of a Type III subsemirun. Bold transitions are crossing, and the first two bold transitions of the figure on the right are moreover doubly crossing.

values in $[B_{\min}+1, B_{\max}-1]$. To categorize such subsemiruns into different types, we introduce the notion of crossing and doubly-crossing transitions.

Definition 42. *A transition* $q_i(z_i) \xrightarrow{\pi_i} q_{i+1}(z_{i+1})$ *is called* crossing *if either*

- $\pi_i = +p$ *and we have* $z_i < B_{\max} \leq z_{i+1}$ *or* $z_i \leq B_{\min} < z_{i+1}$, *or*
- $\pi_i = -p$ *and we have* $z_i > B_{\min} \geq z_{i+1}$ *or* $z_i \geq B_{\max} > z_{i+1}$.

If even moreover $z_i \leq B_{\min} \leq B_{\max} \leq z_{i+1}$ *or* $z_i \geq B_{\max} \geq B_{\min} \geq z_{i+1}$ *we call* π_i doubly-crossing.

We already refer to Figure 4.9, where subsemiruns of a certain type (to be defined below) are depicted, some of whose transitions crossing transitions, some of whose are even doubly-crossing transitions.

Next, we introduce three particular types of subsemiruns of π starting and ending in configurations with counter values in $[B_{\min}+1, B_{\max}-1]$.

Definition 43 (Type I, II and III subsemiruns of π)**.** *A subsemirun* $\pi[a,b]$ *of* π *with source and target configuration in* $Q \times [B_{\min}+1, B_{\max}-1]$ *is*

- *of* Type I *if* $\textsc{Values}(\pi[a,b]) \subseteq [B_{\min}+1, B_{\max}-1]$,
- *of* Type II *if*
  - $\textsc{Values}(\pi[a+1,b-1]) \cap [B_{\min}+1, B_{\max}-1] = \varnothing$, *and*
  - $\pi[a,b]$ *does not contain any doubly-crossing transitions,*
- *of* Type III *if*
  - $\textsc{Values}(\pi[a+1,b-1]) \cap [B_{\min}+1, B_{\max}-1] = \varnothing$, *and*
  - $\pi[a,b]$ *contains at least one doubly-crossing transition.*

Remark 44. *All crossing transitions in a Type III semirun, except possibly the first or the last transition, are doubly-crossing.*

Figure 4.9 shows an example of a Type II and of a Type III subsemirun.

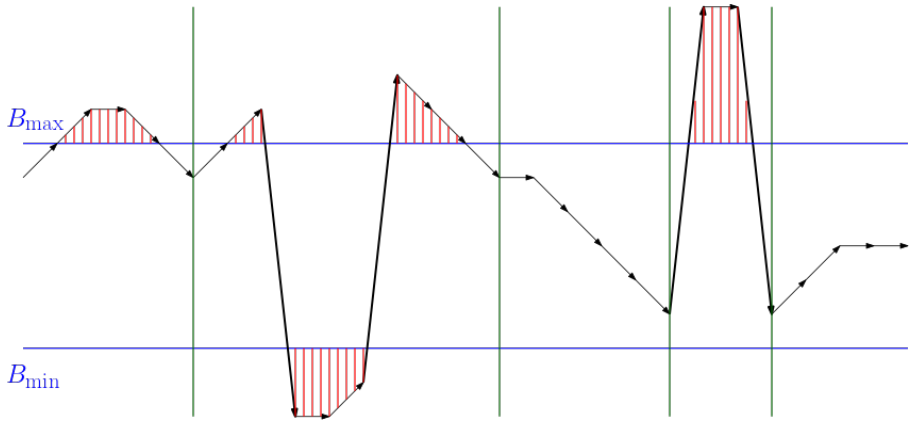

Figure 4.10: In this figure, we provide an example factorization of a semirun π. A semirun π is divided into five subsemiruns, separated by vertical lines. The third and fifth subsemiruns are of Type I, the first and fourth subsemiruns are of Type II, and the second one is of Type III.

The following lemma factorizes π into Type I, Type II and Type III subsemiruns, bearing in mind that both the source and target configuration of π have a counter value in $[B_{\min}+1, B_{\max}-1]$.

Lemma 45. *The N-semirun π can be factorized into Type I, Type II, and Type III subsemiruns.*

Proof. Let us first factorize π as

$$\pi \quad = \quad \pi[c_1,d_1]\pi[d_1,c_2]\pi[c_2,d_2]\pi[d_2,c_3] \quad \cdots \quad \pi[c_t,d_t], \tag{4.6}$$

where

- $\pi[c_i,d_i]$ are Type I and *maximal* (possibly empty), i.e. $\pi[c_i,d_i]$ is of Type I but neither $\pi[c_i-1,d_i]$ nor $\pi[c_i,d_i+1]$ is of Type I for all $i \in [1,t]$, and
- $\textsc{Values}(\pi[d_i+1,c_{i+1}-1]) \cap [B_{\min}+1, B_{\max}-1] = \varnothing$ for all $i \in [1,t-1]$.

Now it suffices to show that each subsemirun $\pi[d_i,c_{i+1}]$ is either of Type II or of Type III. For this, let us make a case distincton on whether $\pi[d_i,c_{i+1}]$ contains a doubly-crossing transition or not.

For the first case, namely that $\pi[d_i,c_{i+1}]$ does contain a doubly-crossing transition, since $\textsc{Values}(\pi[d_i+1,c_{i+1}-1]) \cap [B_{\min}+1, B_{\max}-1] = \varnothing$, we have that $\pi[d_i,c_{i+1}]$ is of Type III by definition.

For the second case, namely that $\pi[d_i,c_{i+1}]$ does not contain any doubly-crossing transition, since $\textsc{Values}(\pi[d_i+1,c_{i+1}-1]) \cap [B_{\min}+1, B_{\max}-1] = \varnothing$ we have that $\pi[d_i,c_{i+1}]$ is of Type II by definition. □

By Remark 28 in order to prove the existence of the desired $(N-\Gamma_{\mathcal{C},h})$-semirun it suffices to show it for Type I, Type II and Type III subsemiruns of π.

Since Type I subsemiruns neither contain any $+p$-transition nor any $-p$-transition by (4.5), they are already $(N-\Gamma_{\mathcal{C},h})$-semiruns.

Let us now discuss the situation for Type II subsemiruns of π. If a Type II subsemirun is already an $(N-\Gamma_{\mathcal{C},h})$-subsemirun we are done as above. In case a Type II subsemirun ρ is not of Type I we first claim that ρ is either a $B_{\min}$-valley or a $B_{\max}$-hill. Indeed, if ρ is of Type II but not of Type I one can factorize ρ as

$$\rho \quad = \quad p_0(x_0) \quad \overset{\rho_0}{\dashrightarrow}_N \quad p_1(x_1) \quad \cdots \quad \overset{\rho_{m-1}}{\dashrightarrow}_N \quad p_m(x_m),$$

where

1. $m \geq 2$,
2. $x_0, x_m \in [B_{\min}+1, B_{\max}-1]$, and
3. either $x_i \in [0, B_{\min}]$ for all $i \in [1, m-1]$ or $x_i \in [B_{\max}, h \cdot N]$ for all $i \in [1, m-1]$,

where Point 3 follows from the absence of doubly-crossing transitions.

First assume that $x_i \in [B_{\max}, h \cdot N]$ for all $i \in [1, m-1]$. In this case any $+p$-transition (resp. $-p$-transition) ends (resp. starts) in a configuration with counter value strictly larger than $B_{\min}+N$. Due to the definition of our constants on page 33 we have

$$\begin{aligned} x_0 + \Upsilon_{\mathcal{C},h},\ x_n + \Upsilon_{\mathcal{C},h} &< B_{\max} + \Upsilon_{\mathcal{C},h} \\ &= B_{\min} + (B_{\max} - B_{\min}) + \Upsilon_{\mathcal{C},h} \\ &\leq B_{\min} + 5/6 \cdot N + 3\Upsilon_{\mathcal{C},h} + 4\Gamma_{\mathcal{C},h} + 2 \\ &< B_{\min} + 5/6 \cdot N + M_{\mathcal{C},h}/6 \\ &< B_{\min} + N, \end{aligned}$$

hence ρ is a $B_{\max}$-hill.

Secondly, in case $x_i \in [0, B_{\min}]$ for all $i \in [1, m-1]$ it can analogously be shown that ρ is $B_{\min}$-valley.

The existence of the desired $(N - \Gamma_{\mathcal{C},h})$-semirun ρ' that is an ℓ-embedding of the Type II semirun ρ with the same source and target configuration as ρ follows immediately from the following claim, which itself (with a short justificaton below) is a consequence of the Hill and Valley Lemma (Lemma 30); thanks to the fact that the Hill and Valley Lemma guarantees the resulting $(N - \Gamma_{\mathcal{C},h})$-semiruns to be min-rising and max-falling, we can even guarantee $\textsc{Values}(\rho') \subseteq [\min(\rho), \max(\rho)]$.

Claim 2. *For every N-semirun ρ that is either a B-hill with $B \geq \ell + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h} + 1$ or a B-valley with $B \leq \ell - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1$, there exists an $(N - \Gamma_{\mathcal{C},h})$-semirun that is both a min-rising and max-falling ℓ-embedding of ρ with same source and target configuration as ρ.*

That the Hill and Valley Lemma produces an ℓ-embedding that has the same source and target configuration is important here. Indeed, generally speaking if ρ is any B-hill and ρ' is any k-embedding of ρ with the same source and target configuration as ρ and where $k < B$, then ρ' is also a k'-embedding of ρ for all $k' < k$. Dually, if ρ is any B-valley and ρ' is a k-embedding of ρ with with the same source and target configuration as ρ and where $k > B$, then ρ' is also an k'-embedding of ρ for all $k' > k$.

For the rest of this section it now suffices to prove that for every Type III subsemirun ρ of π there exists an $(N - \Gamma_{\mathcal{C},h})$-semirun ρ' that is an ℓ-embedding of ρ with the same source and target configuration as ρ and that moreover satisfies $\textsc{Values}(\rho') \subseteq [\min(\rho) - \Gamma_{\mathcal{C},h}, \max(\rho) + \Gamma_{\mathcal{C},h}]$.

4.6.1 Lowering Type III subsemiruns

For the rest of the Section let us fix a Type III subsemirun ρ of π. Let us factorize ρ by its crossing semitransitions, i.e. as

$$\rho \quad = \quad \alpha^{(0)}\beta^{(1)}\alpha^{(1)} \quad \cdots \quad \beta^{(n)}\alpha^{(n)},$$

where $\beta^{(1)}, \ldots, \beta^{(n)}$ is an enumeration of the crossing semitransitions of ρ and each $\alpha^{(i)}$ is a (possibly empty) N-subsemirun of ρ. It is worth mentioning that, indeed abusing notation, for the rest of this section we refer to n as the number of crossing semitransitions of ρ, rather than the number of transitions of our original N-run π.

An example factorization is shown in Figure 4.9, where the crossing transitions are depicted in bold. We remark that the only crossing transitions of ρ that are not doubly-crossing can possibly only be the first or the last one (or both).

We intend to now factorize ρ, if possible, into hills and valleys. In order to do this let us first introduce the notions of B-hill candidate and B-valley candidate.

Definition 46. *Let*

$$\chi \quad = \quad p_0(x_0) \xrightarrow{\chi_0}_N p_1(x_1) \quad \cdots \quad \xrightarrow{\chi_{m-1}}_N p_m(x_m)$$

be an N-semirun. We say χ is a B-hill candidate *if $x_0, x_m < B$ and $x_i \geq B$ for all $i \in [1, m-1]$, respectively a* B-valley candidate *if $x_0, x_m > B$ and $x_i \leq B$ for all $i \in [1, m-1]$.*

Note that every B-hill is a B-hill candidate but not vice versa, since being a B-hill moreover requires $+p$-transitions to end at a configuration with counter value strictly larger than $x_n + \Upsilon_{\mathcal{C},h}$ and $-p$-transitions to start at a configuration with counter value strictly larger than $x_0 + \Upsilon_{\mathcal{C},h}$. A similar remark applies to B-valleys and B-valley candidates.

For the rest of this section we assume *without loss of generality* that the crossing transition β_1 is a $+p$-transition. The case when β_1 is $-p$-transition can be proven analogously.

It follows that if the number n of crossing transitions is even, then there is a unique factorization

$$\rho \quad = \quad \alpha^{(0)}\sigma^{(1)}\alpha^{(2)}\sigma^{(2)}\alpha^{(4)}\sigma^{(3)}\alpha^{(6)} \quad \cdots \quad \sigma^{(n/2)}\alpha^{(n)}, \tag{4.7}$$

where $\sigma^{(i)} = \beta^{(2i-1)}\alpha^{(2i-1)}\beta^{(2i)}$ is a $B_{\max}$-hill candidate, $\beta^{(2i-1)}$ is a $+p$-transition and $\beta^{(2i)}$ is a $-p$-transition for all $i \in [1, n/2]$. Indeed, this immediately follows from the definition of crossing transitions and the fact that $\alpha^{(2i-1)}$ does not contain any configuration with counter value strictly less than $B_{\max}$.

Therefore our proof makes first a case distinction on the parity of the number n of crossing transitions.

Case A: The number of crossing transitions n is even.

Our proof next makes a case distinction on the number of $B_{\max}$-hill candidates $\sigma^{(i)}$ in the factorization (4.7) that are in fact $B_{\max}$-hills.

Case A.1: All of the $B_{\max}$-hill candidates $\sigma^{(i)}$ in (4.7) are in fact $B_{\max}$-hills.

Since each $\sigma^{(i)} = \beta^{(2i-1)}\alpha^{(2i-1)}\beta^{(2i)}$ from (4.7) is a $B_{\max}$-hill, to each $\sigma^{(i)}$ we can apply Claim 2 and obtain an $(N - \Gamma_{\mathcal{C},h})$-semirun $\widehat{\sigma^{(i)}}$ that is both a min-rising and max-falling ℓ-embedding of $\sigma^{(i)}$ with the same source and target configuration as $\sigma^{(i)}$. Thus, it remains to show the same for $\alpha^{(2i)}$ for each $i \in [0, n/2]$. We do this separately for $\alpha^{(0)}, \alpha^{(n)}$ and finally for those $\alpha^{(2i)}$, where $i \in [1, n/2 - 1]$.

Let us first show it for $\alpha^{(0)}$. The proof for $\alpha^{(n)}$ is completely analogous. If $\alpha^{(0)}$ is empty (which would imply that $\beta^{(1)}$ is crossing but not doubly-crossing), there is nothing to show. Let us therefore assume that $\alpha^{(0)}$ is not empty. It follows that $\beta^{(1)}$ must be a doubly-crossing $+p$-transition by Remark 44. Since $\beta^{(1)}$ is the first crossing transition (even doubly-crossing) and a $+p$-transition and moreover ρ is of Type III one can factorize $\alpha^{(0)}$ as

$$\alpha^{(0)} \quad = \quad \alpha^{(0,0)}\sigma^{(0,1)}\alpha^{(0,1)} \quad \cdots \quad \sigma^{(0,k)},$$

where $\alpha^{(0,j)}$ satisfies $\textsc{Values}(\alpha^{(0,j)}) \subseteq [B_{\max} - N, B_{\min} + 1]$ for all $j \in [0, k]$ and $\sigma^{(0,j)}$ is a $(B_{\max} - N - 1)$-valley candidate for all $j \in [1, k]$. It immediately follows that each $\alpha^{(0,j)}$ does not contain any $+p$-transition nor any $-p$-transition, and is hence already an $(N - \Gamma_{\mathcal{C},h})$-semirun. Finally we claim that each $\sigma^{(0,j)}$ is in fact a $(B_{\max} - N - 1)$-valley. Indeed, firstly the target configuration of each $+p$-transition in $\sigma^{(0,j)}$ has a counter value at most $B_{\min}$ and hence a source configuration with counter value at most $B_{\min} - N < B_{\max} - N - 1 - \Upsilon_{\mathcal{C},h}$, where the inequality follows from definition of $B_{\min}$ and $B_{\max}$ from page 55. Secondly and analogously, the source configuration of each $-p$-transition in $\sigma^{(0,j)}$ has a counter value of at most $B_{\min}$ and hence a target configuration with

counter value at most $B_{\min} - N < B_{\max} - N - 1 - \Upsilon_{\mathcal{C},h}$. Thus, to each $\sigma^{(0,j)}$ we can apply Claim 2 to obtain an $(N - \Gamma_{\mathcal{C},h})$-semirun $\widetilde{\sigma^{(0,j)}}$ that is a min-rising and max-falling ℓ-embedding of $\sigma^{(0,j)}$ with the same source and target configuration as $\sigma^{(0,j)}$. Hence, by appropriately concatenationg the $\alpha^{(0,j)}$ with the $\widetilde{\sigma^{(0,j)}}$ we obtain the desired $(N - \Gamma_{\mathcal{C},h})$-semirun that is an ℓ-embedding of $\alpha^{(0)}$.

It now only remains and suffices to show that for each $\alpha^{(2i)}$ with $i \in [1, n/2 - 1]$, that there exists an $(N - \Gamma_{\mathcal{C},h})$-semirun that is a min-rising and max-falling ℓ-embedding of $\alpha^{(2i)}$ with same source and target configuration as $\alpha^{(2i)}$. Again by Remark 44 any such $\alpha^{(2i)}$ succeeds $\sigma^{(i)} = \beta^{(2i-1)}\alpha^{(2i-1)}\beta^{(2i)}$ and thus succeeds the doubly-crossing transition $\beta^{(2i)}$ and analogously preceeds $\sigma^{(i+1)} = \beta^{(2i+1)}\alpha^{(2i+1)}\beta^{(2i+2)}$ and thus precedes the doubly-crossing $\beta^{(2i+1)}$. Therefore, analogously as done for $\alpha^{(0)}$, one can factorize $\alpha^{(2i)}$ as

$$\alpha^{(2i)} \quad = \quad \alpha^{(2i,0)}\sigma^{(2i,1)}\alpha^{(2i,1)} \quad \dots \quad \sigma^{(2i,k)}\alpha^{(2i,k)}$$

for some $k \geq 0$, where $\alpha^{(2i,j)}$ satisfies $\textsc{Values}(\alpha^{(2i,j)}) \subseteq [B_{\max} - N, B_{\min}]$ (and is thus already an $(N - \Gamma_{\mathcal{C},h})$-semirun) for all $j \in [0, k]$ and $\sigma^{(2i,j)}$ is a $(B_{\max} - N - 1)$-valley candidate that is in fact a $(B_{\max} - N - 1)$-valley for all $j \in [1, k]$.

Case A.2: All but one of the $B_{\max}$-hill candidates $\sigma^{(i)}$ in (4.7) are in fact $B_{\max}$-hills.

Recall the factorization

$$\rho \quad = \quad \alpha^{(0)}\sigma^{(1)}\alpha^{(2)}\sigma^{(2)}\alpha^{(4)}\sigma^{(3)}\alpha^{(6)} \quad \dots \quad \sigma^{(n/2)}\alpha^{(n)}$$

from (4.7) where each $\sigma^{(i)} = \beta^{(2i-1)}\alpha^{(2i-1)}\beta^{(2i)}$ is a $B_{\max}$-hill candidate for all $i \in [1, n/2]$.

First let us show that, in case one such $B_{\max}$-hill candidate is not a $B_{\max}$-hill, then it must be either $\sigma^{(1)}$ or $\sigma^{(n/2)}$. For every of the remaining $j \in [2, n/2-1]$ we have that $\sigma^{(j)} = \beta^{(2j-1)}\alpha^{(2j-1)}\beta^{(2j)}$ is such that $\beta^{(2j-1)}$ and $\beta^{(2j)}$ are both doubly-crossing, implying that $\sigma^{(j)}$ is a $B_{\max}$-hill: indeed, both the source and target configuration of $\sigma^{(j)}$ have a counter value at most $B_{\min} < B_{\max} - \Upsilon_{\mathcal{C},h}$, which is sufficient since every $+p$-transition (resp. $-p$-transition) of $\sigma^{(j)}$ ends (resp. starts) in a configuration with counter value at least $B_{\max}$.

Let us assume without loss of generality that $\sigma^{(1)} = \beta^{(1)}\alpha^{(1)}\beta^{(2)}$ is the only $B_{\max}$-hill candidate that is not a $B_{\max}$-hill, the case when $\sigma^{(n/2)}$ is not a $B_{\max}$-hill can be treated analogously.

Clearly, by the above reasoning, either $\beta^{(1)}$ or $\beta^{(2)}$ must not be doubly-crossing. Without loss of generality let us assume that the first crossing transition $\beta^{(1)}$ is not doubly-crossing (for $\beta^{(2)}$ not doubly-crossing implies $n = 2$ by definition of Type III and Remark 44; this case is thus included in the dual case when $\sigma^{(n/2)}$ is not a $B_{\max}$-hill and the last crossing transition $\beta^{(n)}$ is not doubly-crossing).

Remarking that $\alpha^{(0)}$ must be empty by our case, one can now factorize our N-semirun ρ as

$$\rho \quad = \quad \left(\beta^{(1)}\alpha^{(1)}\right)\beta^{(2)}\left(\alpha^{(2)}\sigma^{(2)}\alpha^{(4)} \quad \dots \quad \sigma^{(n/2)}\alpha^{(n)}\right),$$

where $\beta^{(2)}$ is a $-p$-transition. For finishing this case we will proceed as follows.

1. Firstly we show the existence of an $(N-\Gamma_{\mathcal{C},h})$-semirun that is both a min-rising and max-falling ℓ-embedding of $\beta^{(1)}\alpha^{(1)}$ with the same source and target configuration as $\beta^{(1)}\alpha^{(1)}$.

2. Secondly, let us assume that $\beta^{(2)}$ is an N-semirun from $q(x)$ to $q'(y)$, say, and moreover that $\alpha^{(2)}\sigma^{(2)}\alpha^{(4)}\dots\,\sigma^{(n/2)}\alpha^{(n)}$ is an N-semirun from $q'(y)$ to $q''(z)$, say. Noting that $\beta^{(2)} = q(x) \xrightarrow{-p}_N q'(y)$, we explicitly lower $\beta^{(2)}$ into the $(N - \Gamma_{\mathcal{C},h})$-semirun $q(x) \xrightarrow{-p}_{N-\Gamma_{\mathcal{C},h}} q'(y + \Gamma_{\mathcal{C},h})$, which is — since $\beta^{(2)}$ is doubly-crossing — obviously both a min-rising and max-falling ℓ-embedding of $\beta^{(2)}$ from $q(x)$ to $q'(y + \Gamma_{\mathcal{C},h})$. Finally, we show the existence of an $(N - \Gamma_{\mathcal{C},h})$-semirun that is an ℓ-embedding of $\alpha^{(2)}\sigma^{(2)}\alpha^{(4)}\dots\,\sigma^{(n/2)}\alpha^{(n)}$ from $q'(y + \Gamma_{\mathcal{C},h})$ to $q''(z)$ all of whose counter values lie in $[\min(\rho) - \Gamma_{\mathcal{C},h}, \max(\rho) + \Gamma_{\mathcal{C},h}]$.

Let us first show Point 1. Since $\beta^{(1)}$ is not doubly-crossing and $\sigma^{(1)} = \beta^{(1)}\alpha^{(1)}\beta^{(2)}$ is a $B_{\max}$-hill candidate that is not a $B_{\max}$-hill, the only reason for the latter is the existence of a $-p$-transition τ such that the counter values of the source configuration of τ and $\sigma^{(1)}$ have an absolute difference at

most $\Upsilon_{\mathcal{C},h}$. Such a transition τ must have a source configuration with counter value in the interval $[B_{\max}, B_{\max} + \Upsilon_{\mathcal{C},h} - 1]$ and $\sigma^{(1)}$ must then necessarily have a source configuration with a counter value in the interval $[B_{\max} - \Upsilon_{\mathcal{C},h}, B_{\max} - 1]$. Such a violation can only happen for $\tau = \beta^{(2)}$. As a consequence, the target configuration $q(x)$ of $\alpha^{(1)}$ has a counter value inside $[B_{\max}, B_{\max} + \Upsilon_{\mathcal{C},h} - 1]$. Recalling that $\beta^{(1)}$ is not doubly-crossing one can (analogously as has been done in Case A.1) factorize $\beta^{(1)}\alpha^{(1)}$ as

$$\beta^{(1)}\alpha^{(1)} \quad = \quad \chi^{(1)}\xi^{(1)} \cdots \chi^{(k)}\xi^{(k)},$$

for some $k \geq 0$, where each $\chi^{(i)}$ is a $(B_{\min} + N + 1)$-hill and each $\xi^{(i)}$ satisfies $\textsc{Values}(\xi^{(i)}) \subseteq [B_{\max}, B_{\min} + N]$. Again, analogously as has been done in Case A.1, to each of the $\chi^{(i)}$ we can apply Claim 2 to turn them into a suitable min-rising and max-falling $(N - \Gamma_{\mathcal{C},h})$-semirun that is an ℓ-embedding of $\chi^{(i)}$ with the same source and target configuration as $\chi^{(i)}$, whereas each of the $\xi^{(i)}$ are already $(N - \Gamma_{\mathcal{C},h})$-semiruns since they do not contain any $+p$-transitions nor $-p$-transitions.

Let us finally show Point 2. Consider the remaining factorization

$$\gamma \quad = \quad \alpha^{(2)}\sigma^{(2)}\alpha^{(4)} \quad \cdots \quad \sigma^{(n/2)}\alpha^{(n)} \tag{4.8}$$

from $q'(y)$ to $q''(z)$. We need prove the existence of an $(N - \Gamma_{\mathcal{C},h})$-semirun from $q'(y + \Gamma_{\mathcal{C},h})$ to $q''(z)$ that is an ℓ-embedding of γ and whose counter values lie in $[\min(\gamma) - \Gamma_{\mathcal{C},h}, \max(\gamma) + \Gamma_{\mathcal{C},h}]$.

We first claim that $\Delta(\gamma) > \Upsilon_{\mathcal{C},h}$. Since $x \in [B_{\max}, B_{\max} + \Upsilon_{\mathcal{C},h} - 1]$ it follows that the counter value y of γ's source configuration $q'(y)$ satisfies $y \in [B_{\max} - N, B_{\max} + \Upsilon_{\mathcal{C},h} - 1 - N]$. Moreover, the target configuration $q''(z)$ of γ is the target configuration of our Type III N-semirun ρ, thus $z \in [B_{\min} + 1, B_{\max} - 1]$. Hence by the definition of our constants on page 33 we have

$$\begin{aligned}
\Delta(\gamma) &\geq B_{\min} + 1 - (B_{\max} + \Upsilon_{\mathcal{C},h} - 1 - N) \\
&> N - (B_{\max} - B_{\min}) - \Upsilon_{\mathcal{C},h} \\
&\overset{(4.4)}{\geq} N - (5/6 \cdot N + 2\Upsilon_{\mathcal{C},h} + 4\Gamma_{\mathcal{C},h} + 2) - \Upsilon_{\mathcal{C},h} \\
&= N/6 - 2\Upsilon_{\mathcal{C},h} - 4\Gamma_{\mathcal{C},h} - 2 - \Upsilon_{\mathcal{C},h} \\
&> M_{\mathcal{C},h}/6 - 2\Upsilon_{\mathcal{C},h} - 4\Gamma_{\mathcal{C},h} - 2 - \Upsilon_{\mathcal{C},h} \\
&= M_{\mathcal{C},h}/6 - 4\Upsilon_{\mathcal{C},h} - 4\Gamma_{\mathcal{C},h} - 2 + \Upsilon_{\mathcal{C},h} \\
&> (M_{\mathcal{C},h}/6 - 4(\Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h} + 1)) + \Upsilon_{\mathcal{C},h} \\
&> \Upsilon_{\mathcal{C},h}.
\end{aligned}$$

Recall that each $\sigma^{(i)}$ is a $B_{\max}$-hill for all $i \in [2, n/2]$ by our case. Analogously, as has been done in Case A.1, for each $i \in [1, n/2]$ one can factorize $\alpha^{(2i)}$ as

$$\alpha^{(2i)} \quad = \quad \alpha^{(2i,0)}\sigma^{(2i,1)} \quad \cdots \quad \sigma^{(2i,k_i)}\alpha^{(2i,k_i)},$$

where $\alpha^{(2i,j)}$ satisfies $\textsc{Values}(\alpha^{(2i,j)}) \subseteq [B_{\max} - N, B_{\min} + 1]$ for each $j \in [0, k_i]$ and $\sigma^{(2i,j)}$ is a $(B_{\max} - N - 1)$-valley for each $j \in [1, k_i]$: more precisely for the final $\alpha^{(n)}$ we have $\textsc{Values}(\alpha^{(n)}) \subseteq [B_{\max} - N, B_{\min} + 1]$, however for all $i \in [1, n/2 - 1]$ we have $\alpha^{(2i)} \subseteq [B_{\max} - N, B_{\min}]$.

It is important to remark that each $\alpha^{(2i,j)}$ is already an $(N - \Gamma_{\mathcal{C},h})$-semirun since it does not contain any $+p$-transition nor any $-p$-transition. The following remark summarizes the factorization of γ.

Remark 47. *Our N-semirun γ from $q'(y)$ to $q''(z)$ can be written as*

$$\gamma \quad = \quad \alpha^{(2)}\sigma^{(2)}\alpha^{(4)} \quad \cdots \quad \sigma^{(n/2)}\alpha^{(n)} \quad = \quad \alpha^{(2)}\left(\prod_{i=2}^{n/2} \sigma^{(i)}\ \alpha^{(2i,0)}\left(\prod_{j=1}^{k_i} \sigma^{(2i,j)}\alpha^{(2i,j)}\right)\right), \tag{4.9}$$

where

1. *each $\sigma^{(i)}$ is a $B_{\max}$-hill,*

2. *each* $\sigma^{(2i,j)}$ *is a* $(B_{\max} - N - 1)$*-valley,*
3. *each* $\alpha^{(2i,j)}$ *is already an* $(N - \Gamma_{\mathcal{C},h})$*-semirun,*
4. $\Delta(\gamma) > \Upsilon_{\mathcal{C},h}$*, and*
5. *every configuration in* γ *(except for possibly the target configuration* $q''(z)$*) has a counter value in* $[0, B_{\min}] \cup [B_{\max}, h \cdot N]$ *whose absolute difference with* ℓ *is thus strictly larger than* $\Gamma_{\mathcal{C},h}$ *(recall the definition of* $B_{\min}$ *and* $B_{\max}$ *of page 55).*

We can now obtain suitable $(N - \Gamma_{\mathcal{C},h})$-semiruns with the same source and target configuration for any of the above hills and valleys by applying the Hill and Valley Lemma.

Remark 48. *By applying the Hill and Valley Lemma (Lemma 30) we obtain the following.*

1. *For each of the* $B_{\max}$*-hills* $\sigma^{(i)}$ *there exists an* $(N - \Gamma_{\mathcal{C},h})$*-semirun* $\widehat{\sigma^{(i)}}$ *that is both a min-rising and max-falling* $(B_{\max} - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$*-embedding of* $\sigma^{(i)}$ *from the same source and target configuration as* $\sigma^{(i)}$*. In particular, since* $\sigma^{(i)}$ *is a* $B_{\max}$*-hill and* $B_{\max} - \Upsilon_{\mathcal{C}} - \Gamma_{\mathcal{C},h} - 1 > \ell + \Gamma_{\mathcal{C},h}$ *it follows that* $\widehat{\sigma^{(i)}}$ *is in fact an* ℓ*-embedding of* $\sigma^{(i)}$ *all of whose configurations have a counter value whose absolute difference with* ℓ *is strictly larger than* $\Gamma_{\mathcal{C},h}$ *(except for the exotic case when* $i = n/2$ *and* γ *in fact ends with* $\sigma^{(i)}$*, and hence the last configuration of* $\sigma^{(i)}$ *happens to be the last configuration* $q''(z)$ *of* γ*; recalling that* $z \in [B_{\min} + 1, B_{\max} - 1]$*).*
2. *For each of the* $(B_{\max} - N - 1)$*-valleys* $\sigma^{(2i,j)}$ *there exists an* $(N - \Gamma_{\mathcal{C},h})$*-semirun* $\widehat{\sigma^{(2i,j)}}$ *that is both a min-rising and max-falling* $(B_{\max} - N - 1 + \Upsilon_{\mathcal{C},h} + \Gamma_{\mathcal{C},h} + 1)$*-embedding of* $\sigma^{(2i,j)}$ *with the same source and target configuration as* $\sigma^{(2i,j)}$*. In particular, since* $\sigma^{(2i,j)}$ *is a* $(B_{\max} - N - 1)$*-valley and*

$$B_{\max} - N + \Upsilon_C + \Gamma_{\mathcal{C},h} \overset{(4.5)}{<} B_{\min} + \Upsilon_C + \Gamma_{\mathcal{C},h} \leq \ell - \Gamma_{\mathcal{C},h},$$

where the last inequality follows from the definition of $B_{\min}$ *on page 55), it follows that* $\widehat{\sigma^{(2i,j)}}$ *is in fact an* ℓ*-embedding of* $\sigma^{(2i,j)}$ *all of whose configurations have a counter value whose absolute difference with* ℓ *is is strictly larger than* $\Gamma_{\mathcal{C},h}$ *(except, similar as above, for the exotic case when* $i = n/2$*,* $j = k_i$ *and* γ *in fact ends with* $\sigma^{(2i,j)}$*, and hence the last configuration* $\sigma^{(2i,j)}$ *happens to be the last configuration* $q''(z)$ *of* γ*.*

It is worth pointing out that applying the remark immediately would only yield the existence of an $(N - \Gamma_{\mathcal{C},h})$-semirun γ' that is both a min-rising and max-falling ℓ-embedding of γ with the same source configuration $q'(y)$ and the same target configuration $q''(z)$ as γ such that $\textsc{Values}(\gamma') \subseteq [\min(\gamma) - \Gamma_{\mathcal{C},h}, \max(\gamma) + \Gamma_{\mathcal{C},h}]$. However we need to show the existence of such an ℓ-embedding rather from $q'(y + \Upsilon_C)$ to $q''(z)$. For this, we make a final case distinction on whether among the $B_{\max}$-hills $\sigma^{(i)}$ and the $(B_{\max} - N - 1)$-valleys $\sigma^{(2i,j)}$ there exists one whose ϕ-projection contains strictly more occurrences of [as occurrences of].

- *Case A.2.i: Among the* $B_{\max}$*-hills* $\sigma^{(i)}$ *and the* $(B_{\max} - N - 1)$*-valleys* $\sigma^{(2i,j)}$ *there exists one whose* ϕ*-projection contains strictly more occurrences of* [*as occurrences of*]*.*

  Without loss of generality let us assume that there exists some $s \in [2, n/2]$ such that $\sigma^{(s)} = \beta^{(2s-1)}\alpha^{(2s-1)}\beta^{(2s)}$ is a $B_{\max}$-hill for which $\phi(\sigma^{(s)})$ contains strictly more occurrences of [as occurrences of] — the case when there is a $(B_{\max} - N - 1)$-valley $\sigma^{(2s,j)}$ for which $\phi(\sigma^{(2s,j)})$ has the above property can be proven analogously.

  Assume $\sigma^{(s)} = \beta^{(2s-1)}\alpha^{(2s-1)}\beta^{(2s)}$ has source configuration $r_1(x_1)$ and target configuration $r_2(x_2)$, say. Since $\beta^{(2s-1)}$ was surely neither the first nor the last crossing transition of ρ (recall that $s \geq 2$), it follows that $\beta^{(2s-1)}$ is doubly-crossing by Remark 44, and therefore $x_1 \leq B_{\min}$. Recalling the notion of hybrid semirun (Definition 31) we now apply Point 2 of Remark 40 to our $B_{\max}$-hill $\sigma^{(s)}$ and obtain a hybrid semirun η

- whose source configuration is $r_1(x_1)$ and whose target configuration is $r_2(x_2)$,
- that is both a min-rising and max-falling $(B_{\max} - \Upsilon_{\mathcal{C},h} - \Gamma_{\mathcal{C},h} - 1)$-embedding of $\sigma^{(s)}$, and
- that has breadth 1 and contains precisely one unlowered $+p$-transition.

From the above and the fact that $\sigma^{(s)}$ is a $B_{\max}$-hill the following remark follows.

Remark 49. *All configurations of η (except possibly the target configuration $r_2(x_2)$ in the exotic case when γ ends with $\sigma^{(s)}$) have a counter value whose absolute difference with ℓ is strictly larger than $\Gamma_{\mathcal{C},h}$. Moreover on can write η as $\eta = \alpha\beta\alpha'$, where for some intermediate configurations $r'_1(x'_1)$ and $r'_2(x'_2)$ we have that*

- *α is an $(N - \Gamma_{\mathcal{C},h})$-semirun from $r_1(x_1)$ to $r'_1(x'_1)$,*
- *β is an N-semirun $r'_1(x'_1) \xrightarrow{+p}_N r'_2(x'_2)$ that is a $+p$-transition, i.e. $x'_2 = x'_1 + N$, and*
- *α' is an $(N - \Gamma_{\mathcal{C},h})$-semirun from $r'_2(x'_2)$ to $r_2(x_2)$.*

Let $\widehat{\beta}$ denote the lowering of β, i.e. $\widehat{\beta}$ is the $(N-\Gamma_{\mathcal{C},h})$-semirun $r'_1(x'_1) \xrightarrow{+p}_{N-\Gamma_{\mathcal{C},h}} r'_2(x'_2-\Gamma_{\mathcal{C},h})$. By Remark 49 it follows that the $(N - \Gamma_{\mathcal{C},h})$-semirun

$$\theta \quad = \quad \left(\left(\alpha\widehat{\beta}\right) + \Gamma_{\mathcal{C},h}\right)\alpha'$$

from $r'_1(x_1 + \Gamma_{\mathcal{C},h})$ to $r_2(x_2)$ is an ℓ-embedding of η. Bearing in mind our factorization of γ from Remark 4.9 and taking into account Remark 48 we obtain that

$$\widetilde{\gamma^{(1)}} \quad = \quad \left(\left(\widehat{\alpha^{(2)}}\left(\prod_{i=2}^{s-1}\widehat{\sigma^{(i)}}\ \alpha^{(2i,0)}\left(\prod_{j=1}^{k_i}\widehat{\sigma^{(2i,j)}}\ \alpha^{(2i,j)}\right)\right)\right) + \Gamma_{\mathcal{C},h}\right)\theta$$

is an $(N - \Gamma_{\mathcal{C},h})$-semirun from $q'(y + \Gamma_{\mathcal{C},h})$ to $r'_2(x'_2)$ that is an ℓ-embedding of γ's prefix N-semirun

$$\gamma^{(1)} \quad = \quad \alpha^{(2)}\left(\prod_{i=2}^{s-1}\sigma^{(i)}\ \alpha^{(2i,0)}\left(\prod_{j=1}^{k_i}\sigma^{(2i,j)}\ \alpha^{(2i,j)}\right)\right)\sigma^{(s)}$$

from $q'(y)$ to $r'_2(x'_2)$ satisfying $\textsc{Values}(\widetilde{\gamma^{(1)}}) \subseteq [\min(\gamma^{(1)}), \max(\gamma^{(1)}) + \Gamma_{\mathcal{C},h}]$. Moreover we have by Remark 48 that

$$\widehat{\gamma^{(2)}} \quad = \quad \alpha^{(2s,0)}\left(\prod_{j=1}^{k_s}\widehat{\sigma^{(2i,j)}}\alpha^{(2i,j)}\right)\left(\prod_{i=s+1}^{n/2}\widehat{\sigma^{(i)}}\ \alpha^{(2i,0)}\left(\prod_{j=1}^{k_i}\widehat{\sigma^{(2i,j)}}\ \alpha^{(2i,j)}\right)\right)$$

is an $(N - \Gamma_{\mathcal{C},h})$-semirun from $r'_2(x'_2)$ to $q''(z)$ that is both a min-rising and max-falling ℓ-embedding of γ's remaining suffix N-semirun

$$\gamma^{(2)} \quad = \quad \alpha^{(2s,0)}\left(\prod_{j=1}^{k_s}\sigma^{(2i,j)}\alpha^{(2i,j)}\right)\left(\prod_{i=s+1}^{n/2}\sigma^{(i)}\ \alpha^{(2i,0)}\left(\prod_{j=1}^{k_i}\sigma^{(2i,j)}\ \alpha^{(2i,j)}\right)\right)$$

from $r'_2(x'_2)$ to $q''(z)$. Altogether $\gamma' = \widetilde{\gamma^{(1)}}\widehat{\gamma^{(2)}}$ is the desired $(N - \Gamma_{\mathcal{C},h})$-semirun from $q'(y + \Gamma_{\mathcal{C},h})$ to $q''(z)$ that is an ℓ-embedding of $\gamma = \gamma^{(1)}\gamma^{(2)}$ with $\textsc{Values}(\gamma') \subseteq [\min(\gamma) - \Gamma_{\mathcal{C},h}, \max(\gamma) + \Gamma_{\mathcal{C},h}]$.

- *Case A.2.ii: Among the $B_{\max}$-hills $\sigma^{(i)}$ and the $(B_{\max} - N - 1)$-valleys $\sigma^{(2i,j)}$ all have a ϕ-projection that contains at least as many occurrences of*] *as occurrences of* [.

Observe that by Remark 48 the $(N - \Gamma_{\mathcal{C},h})$-semirun

$$\widehat{\gamma} \quad = \quad \alpha^{(2)} \left(\prod_{i=2}^{n/2} \widehat{\sigma^{(i)}} \; \alpha^{(2i,0)} \left(\prod_{j=1}^{k_i} \widehat{\sigma^{(2i,j)}} \; \alpha^{(2i,j)} \right) \right)$$

from $q'(y)$ to $q''(z)$ is both a min-rising and max-falling ℓ-embedding of γ all of whose configurations (except for possibly the target configuration $q''(z)$) have a counter value whose absolute difference with ℓ is strictly larger than $\Gamma_{\mathcal{C},h}$. Yet we need to show the existence of some $(N - \Gamma_{\mathcal{C},h})$-semirun γ' that is an ℓ-embedding of γ from $q'(y + \Gamma_{\mathcal{C},h})$ to $q''(z)$ that satisfies $\textsc{Values}(\gamma') \subseteq [\min(\gamma) - \Gamma_{\mathcal{C},h}, \max(\gamma) + \Gamma_{\mathcal{C},h}]$.

By Remark 39 all of the lowered $(N - \Gamma_{\mathcal{C},h})$-semiruns $\widehat{\sigma^{(i)}}$ and $\widehat{\sigma^{(2i,j)}}$ mentioned in Remark 48 contain at least as many occurrences of $[$ as of $]$ or, vice versa, at least as many occurrences of $]$ as of $[$, if $\sigma^{(i)}$ does, respectively if $\sigma^{(2i,j)}$ does.

Thus, by our case we obtain that every $\phi(\widehat{\sigma^{(i)}})$ and $\phi(\widehat{\sigma^{(2i,j)}})$ contains at least as many occurrences of $]$ as occurrences of $[$.

Recalling that neither $\alpha^{(2)}$ nor any of the $\alpha^{(2i,j)}$ contain any $+p$-transitions nor $-p$-transitions (and thus have all a ϕ-projection ε) it follows that $\phi(\widehat{\gamma})$ contains at least as many occurrences of $]$ as occurrences of $[$.

Since $\Delta(\gamma) > \Upsilon_{\mathcal{C},h}$ by Point 4 of Remark 4.9, and thus $\Delta(\widehat{\gamma}) > \Upsilon_{\mathcal{C},h}$, there exists a subsemirun $\widehat{\gamma}[c,d]$ satisfying $\Delta(\widehat{\gamma}[c,d]) > \Upsilon_{\mathcal{C},h}$ and $\phi(\widehat{\gamma}[c,d]) \in \Lambda_{2h}$ by Lemma 25. By now applying Lemma 24 there exists an $(N - \Gamma_{\mathcal{C},h})$-semirun χ satisfying

- $\Delta(\chi) = \Delta(\widehat{\gamma}[c,d]) - \Gamma_{\mathcal{C},h}$ and
- $\chi = \widehat{\gamma}[c,d] - I_1 - I_2 \cdots - I_h$ for pairwise disjoint intervals $I_1, \ldots, I_h \subseteq [c,d]$ such that $\phi(\widehat{\gamma}[I_i]) \in \Lambda_{4h}$ and $\Delta(\widehat{\gamma}[I_i]) > 0$ for all $i \in [1,h]$.

Note that from the definition of χ and the fact that all intermediate configurations of $\widehat{\gamma}$ have counter values whose absolute difference with ℓ is strictly larger than $\Gamma_{\mathcal{C},h}$ it follows that $\chi + \Gamma_{\mathcal{C},h}$ is an ℓ-embedding of $\widehat{\gamma}[c,d]$ that has the same target configuration as $\widehat{\gamma}[c,d]$ and that satisfies $\textsc{Values}(\chi + \Gamma_{\mathcal{C},h}) \subseteq [\min(\widehat{\gamma}[c,d]) - \Gamma_{\mathcal{C},h}, \max(\widehat{\gamma}[c,d]) + \Gamma_{\mathcal{C},h}]$. Analogously, it follows that $\delta = (\widehat{\gamma}[0,c] + \Gamma_{\mathcal{C},h})\ (\chi + \Gamma_{\mathcal{C},h})$ is an $(N - \Gamma_{\mathcal{C},h})$-semirun from $q'(y + \Gamma_{\mathcal{C},h})$ to the same target configuration as $\widehat{\gamma}[0,d]$ that is an ℓ-embedding of $\widehat{\gamma}[0,d]$ and that satisfies $\textsc{Values}(\delta) \subseteq [\min(\gamma[0,d]) - \Gamma_{\mathcal{C},h}, \max(\gamma[0,d]) + \Gamma_{\mathcal{C},h}]$.

Finally it follows that

$$\gamma' \quad = \quad (\widehat{\gamma}[0,c] + \Gamma_{\mathcal{C},h})\ (\chi + \Gamma_{\mathcal{C},h})\ \widehat{\gamma}[d, |\widehat{\gamma}|]$$

is the desired $(N - \Gamma_{\mathcal{C},h})$-semirun from $q'(y + \Gamma_{\mathcal{C},h})$ to $q''(z)$ that is an ℓ-embedding of $\widehat{\gamma}$ and hence of γ that satisfies $\textsc{Values}(\gamma') \subseteq [\min(\gamma) - \Gamma_{\mathcal{C},h}, \max(\gamma) + \Gamma_{\mathcal{C},h}]$.

Case A.3: All but at least two of the $B_{\max}$-hill candidates $\sigma^{(i)}$ in (4.7) are in fact $B_{\max}$-hills.
Recall the factorization

$$\rho \quad = \quad \alpha^{(0)} \sigma^{(1)} \alpha^{(2)} \sigma^{(2)} \alpha^{(4)} \sigma^{(3)} \alpha^{(6)} \quad \ldots \quad \sigma^{(n/2)} \alpha^{(n)}$$

from (4.7) where each $\sigma^{(i)} = \beta^{(2i-1)} \alpha^{(2i-1)} \beta^{(2i)}$ is a $B_{\max}$-hill candidate for all $i \in [1, n/2]$. By our case we must have $n \geq 4$.

By a similar reasoning as in Case A.2 one can show that the $B_{\max}$-hill candidate that are not $B_{\max}$-hills must be precisely the two subsemiruns $\sigma^{(1)}$ and $\sigma^{(n/2)}$. In particular there cannot be strictly more than two $B_{\max}$-hill candidates in (4.7) that are not in fact $B_{\max}$-hills. Moreover, as already reasoned in Case A.2, neither $\beta^{(1)}$ nor $\beta^{(n)}$ is doubly-crossing. Thus $\alpha^{(0)}$ and $\alpha^{(n)}$ are

empty. Hence, one can now factorize our N-semirun ρ as

$$\rho \quad = \quad \left(\beta^{(1)}\alpha^{(1)}\right)\beta^{(2)}\alpha^{(2)}\beta^{(3)}\alpha^{(3)} \quad \dots \quad \beta^{(n-1)}\left(\alpha^{(n-1)}\beta^{(n)}\right).$$

For finishing this case we will show the existence

1. of an $(N-\Gamma_{\mathcal{C},h})$-semirun that is both a min-rising and max-falling ℓ-embedding of the semirun $\beta^{(1)}\alpha^{(1)}$ with the same source and target configuration as $\beta^{(1)}\alpha^{(1)}$,
2. of an $(N-\Gamma_{\mathcal{C},h})$-semirun that is both a min-rising and max-falling ℓ-embedding of the semirun $\alpha^{(n-1)}\beta^{(n)}$ with the same source and target configuration as $\alpha^{(n-1)}\beta^{(n)}$, and
3. of an $(N-\Gamma_{\mathcal{C},h})$-semirun that is both a min-rising and max-falling ℓ-embedding of the semirun $\beta^{(2)}\alpha^{(2)}\beta^{(3)}\alpha^{(3)}...\beta^{(n-1)}$ with same source and target configuration.

Points 1 and 2 are proven analogously as Point 1 from Case A.2. For proving Point 3, we consider a different factorization

$$\beta^{(2)}\alpha^{(2)}\beta^{(3)}\alpha^{(3)} \quad \dots \quad \beta^{(n-1)} \quad = \quad \tau^{(1)}\alpha^{(3)}\tau^{(2)}\ \alpha^{(5)} \quad \dots \quad \tau^{((n-2)/2)},$$

where $\tau^{(i)} = \beta^{(2i)}\alpha^{(2i)}\beta^{(2i+1)}$ is a $B_{\min}$-valley candidate for all $i \in [1,(n-2)/2]$. Since $\beta^{(2i)}$ and $\beta^{(2i+1)}$ have to be doubly crossing for all $i \in [1,(n-2)/2]$, $\tau^{(i)}$ is in fact a $B_{\min}$-valley for all $i \in [1,(n-2)/2]$ by a similar reasoning as used in Case A.2 to show that the $B_{\max}$-hill candidate that is not a $B_{\max}$-hill must be $\sigma^{(1)}$ or $\sigma^{(n/2)}$.

We can apply Claim 2 to each $\tau^{(i)}$ and obtain an $(N-\Gamma_{\mathcal{C},h})$-semirun $\widehat{\tau^{(i)}}$ that is both a min-rising and max-falling ℓ-embedding of $\tau^{(i)}$ with the same source and target configuration. Thus, it only remains to show the same for $\alpha^{(2i+1)}$ for each $i \in [1,(n-2)/2]$. This is done analogously as in Case A.1 when proving the same for each $\alpha^{(2i)}$ for each $i \in [0,n/2]$.

Case B: The number of crossing transitions n is odd.

Recall that we had assumed without loss of generality that $\beta^{(1)}$ is a $+p$-transition. Since n is odd one can consider the following first factorization

$$\rho \quad = \quad \alpha^{(0)}\sigma^{(1)}\alpha^{(2)}\sigma^{(2)}\alpha^{(4)}\sigma^{(3)}\alpha^{(6)} \quad \dots \quad \sigma^{(\lfloor n/2\rfloor)}\alpha^{(n-1)}\beta^{(n)}\alpha^{(n)}, \tag{4.10}$$

where $\sigma^{(i)} = \beta^{(2i-1)}\alpha^{(2i-1)}\beta^{(2i)}$ is a $B_{\max}$-hill candidate, $\beta^{(2i-1)}$ is a $+p$-transition, $\beta^{(2i)}$ is a $-p$-transition for all $i \in [1,\lfloor n/2\rfloor]$, and $\beta^{(n)}$ is a $+p$-transition; as well as the following second factorization

$$\rho \quad = \quad \alpha^{(0)}\beta^{(1)}\alpha^{(1)}\tau^{(1)}\alpha^{(3)}\tau^{(2)}\alpha^{(5)}\tau^{(3)} \quad \dots \quad \tau^{(\lfloor n/2\rfloor)}\alpha^{(n)}, \tag{4.11}$$

where β_1 is a $+p$-transition, $\tau^{(i)} = \beta^{(2i)}\alpha^{(2i)}\beta^{(2i+1)}$ is a $B_{\min}$-valley candidate, $\beta^{(2i)}$ is a $-p$-transition and $\beta^{(2i+1)}$ is a $+p$-transition for all $i \in [1,\lfloor n/2\rfloor]$. Indeed, this— as for the Case A factorization (4.7)— immediately follows from the definition of crossing transitions and the fact that neither $\alpha^{(2i-1)}$ (resp. $\alpha^{(2i)}$) contains any configuration with counter value strictly less than $B_{\max}$ (resp. strictly larger than $B_{\min}$) for all $i \in [1,\lfloor n/2\rfloor]$.

Our proof next will make a case distinction on the number of $B_{\max}$-hill candidates $\sigma^{(i)}$ in the factorization (4.10) that are in fact $B_{\max}$-hills and on the number of $B_{\min}$-valley candidates $\tau^{(i)}$ in the factorization (4.11) that are in fact $B_{\min}$-valleys.

Case B.1: All of the $B_{\max}$-hill candidates $\sigma^{(i)}$ in (4.10) are in fact $B_{\max}$-hills or all of the $B_{\min}$-valley candidates $\tau^{(i)}$ in (4.11) are in fact $B_{\min}$-valleys.

Let us assume without loss of generality that all of the $B_{\max}$-hill candidates in (4.10) are in fact $B_{\max}$-hills. The case when all $B_{\min}$-valley candidates in (4.11) are in fact $B_{\min}$-valleys can be proven analogously. Each N-semirun $\sigma^{(i)}$ can hence be turned into an $(N-\Gamma_{\mathcal{C},h})$-semirun $\widehat{\sigma^{(i)}}$ that is both a min-rising and max-falling ℓ-embedding of $\sigma^{(i)}$ with same source and target configuration as $\sigma^{(i)}$ according to Claim 2. Moreover, the same holds for $\alpha^{(2i)}$ for all $i \in [0,\lfloor n/2\rfloor - 1]$, as seen in Case A.1. Thus it remains to deal with the subsemiruns $\alpha^{(n-1)}$, $\beta^{(n)}$ and $\alpha^{(n)}$.

We make a final case distinction on the target configuration of the dangling $+p$-transition $\beta^{(n)}$.

- *Case B.1.i: $\beta^{(n)}$ has a target configuration with a counter value strictly larger than $B_{\max}+\Upsilon_{\mathcal{C},h}$.*
  Then clearly $\beta^{(n)}\alpha^{(n)}$ is a $B_{\max}$-hill as well since $\alpha^{(n)}$ contains no configurations with counter value strictly less than $B_{\max}$ besides its last one. The N-semirun $\beta^{(n)}\alpha^{(n)}$ can hence be turned into an $(N-\Gamma_{\mathcal{C},h})$-semirun that is both a min-rising and max-falling ℓ-embedding with the same source and target configuration according to Claim 2. Moreover, the same holds for $\alpha^{(n-1)}$, as analogously proven for $\alpha^{(2i)}$ for all $i \in [0, \lfloor n/2 \rfloor]$ in Case A.1. The concatenation of these two ℓ-embeddings yields an $(N-\Gamma_{\mathcal{C},h})$-semirun that is a min-rising and max-falling ℓ-embedding of $\alpha^{(n-1)}\beta^{(n)}\alpha^{(n)}$ with the same source and target configuration.

- *Case B.1.ii: $\beta^{(n)}$ has a target configuration with a counter value strictly less than $B_{\max}$.*
  It immediately follows that $\beta^{(n)}$ is crossing but not doubly-crossing, thus $\alpha^{(n)}$ is empty. The remaining $\alpha^{(n-1)}\beta^{(n)}$ can thus be factorized as

$$\alpha^{(n-1)}\beta^{(n)} \quad = \quad \xi^{(1)}\chi^{(1)}...\xi^{(k)}\chi^{(k)},$$

  where each $\chi^{(i)}$ is a $(B_{\max}-N-1)$-valley and each $\xi^{(i)}$ satisfies $\textsc{Values}(\xi^{(i)}) \subseteq [B_{\max}-N, B_{\min}+1]$, using a similar factorization as for proving Point 1 in Case A.2. The N-semirun $\alpha^{(n-1)}\beta^{(n)}$ can hence be turned into an $(N-\Gamma_{\mathcal{C},h})$-semirun that is both a min-rising and max-falling ℓ-embedding with same source and target configuration. Recalling that $\alpha^{(n)}$ is empty, the above embedding is an $(N-\Gamma_{\mathcal{C},h})$-semirun that is both a min-rising and max-falling ℓ-embedding of $\alpha^{(n-1)}\beta^{(n)}\alpha^{(n)}$ with same source and target configuration.

- *Case B.1.iii: $\beta^{(n)}$ has a target configuration with counter value in $[B_{\max}, B_{\max}+\Upsilon_{\mathcal{C},h}]$.*
  Thus, the source configuration of $\beta^{(n)}$ has a counter value in $[B_{\max}-N, B_{\max}+\Upsilon_{\mathcal{C},h}-N]$. One finishes this case analogously as Points 1 and 2 in Case A.2:

  1. Firstly, one shows the existence of an $(N-\Gamma_{\mathcal{C},h})$-semirun that is both a min-rising and max-falling ℓ-embedding of $\alpha^{(n)}$ with the same source and target configuration as $\alpha^{(n)}$ as follows: one factorizes $\alpha^{(n)}$ into $(N-\Gamma_{\mathcal{C},h})$-semiruns that have all counter values in $[B_{\max}-1, B_{\min}+N]$ and into $(B_{\min}+N+1)$-hills.
  2. Secondly, let us assume that $\beta^{(n)}$ is an N-semirun from $q'(y)$ to $q''(z)$ and that moreover $\alpha^{(0)}\sigma^{(1)}\alpha^{(2)}...\sigma^{(\lfloor n/2 \rfloor)}\alpha^{(n-1)}$ is an N-semirun from $q(x)$ to $q'(y)$. Stipulating that $\beta^{(n)} = q'(y) \xrightarrow{+p}_N q''(z)$, we explicitly lower $\beta^{(n)}$ into the $(N-\Gamma_{\mathcal{C},h})$-semirun $q'(y+\Gamma_{\mathcal{C},h}) \xrightarrow{+p}_{N-\Gamma_{\mathcal{C},h}} q''(z)$, which is — since $\beta^{(n)}$ is doubly-crossing — obviously both a min-rising and max-falling ℓ-embedding of $\beta^{(n)}$ from $q'(y+\Gamma_{\mathcal{C},h})$ to $q''(z)$. Then one shows the existence of an $(N-\Gamma_{\mathcal{C},h})$-semirun that is an ℓ-embedding of $\alpha^{(0)}\sigma^{(1)}\alpha^{(2)}...\sigma^{(\lfloor n/2 \rfloor)}\alpha^{(n)}$ from $q(x)$ to $q'(y+\Gamma_{\mathcal{C},h})$ with configurations all of whose counter values lie in the interval $[\min(\rho)-\Gamma_{\mathcal{C},h}, \max(\rho)+\Gamma_{\mathcal{C},h}]$ as follows: one subfactorizes each of the $\alpha^{(2i)}$ into $(N-\Gamma_{\mathcal{C},h})$-semiruns that have a counter values in $[B_{\max}-N, B_{\min}]$ and into $(B_{\max}-N-1)$-valleys and by recalling that $\sigma^{(i)}$ is a $B_{\max}$-hill for all $i \in [1, \lfloor n/2 \rfloor]$.

Case B.2: Not all of the $B_{\max}$-hill candidates $\sigma^{(i)}$ in (4.10) are in fact $B_{\max}$-hills and not all of the $B_{\min}$-valley candidates $\tau^{(i)}$ in (4.11) are in fact $B_{\min}$-valleys.

Since they all start and end with a doubly-crossing transition we remark that $\sigma^{(i)}$ is in fact a $B_{\max}$-hill for all $i \in [2, \lfloor n/2 \rfloor]$ and $\tau^{(i)}$ is in fact a $B_{\min}$-valley for all $i \in [1, \lfloor n/2 \rfloor - 1]$. Hence our case implies that $\sigma^{(1)}$ is in fact not a $B_{\max}$-hill and that $\tau^{(\lfloor n/2 \rfloor)}$ is in fact not a $B_{\min}$-valley. As in Case A.2, $\beta^{(1)}$ and $\beta^{(n)}$ are hence not doubly-crossing, and hence $\alpha^{(0)}$ and $\alpha^{(n)}$ are empty.

By definition of a Type III semirun, ρ contains at least one doubly-crossing transition and thus $n \geq 3$. Since the $+p$-transition $\beta^{(1)}$ is not doubly-crossing (and therefore ends at a counter value strictly larger than $B_{\min}+N$) but $\beta^{(2)}$ is, it follows that the only reason for $\sigma^{(1)} = \beta^{(1)}\alpha^{(1)}\beta^{(2)}$ not to be a $B_{\max}$-hill is that the $-p$-transition $\beta^{(2)}$ has a source configuration with a counter value in $[B_{\max}, B_{\max}+\Upsilon_{\mathcal{C},h}]$ and hence a target configuration with counter value in $[B_{\max}-N, B_{\max}+\Upsilon_{\mathcal{C},h}-N]$, similarly as seen in Case A.2. Analogously, the only reason for $\tau^{(\lfloor n/2 \rfloor)} = \beta^{(n-1)}\alpha^{(n-1)}\beta^{(n)}$ not

to be a $B_{\min}$-valley is that the doubly-crossing $-p$-transition $\beta^{(n-1)}$ has a target configuration with counter value in $[B_{\min} - \Upsilon_{\mathcal{C},h}, B_{\min}]$.

Recalling that $\sigma^{(\lfloor n/2 \rfloor)} = \beta^{(n-2)}\alpha^{(n-2)}\beta^{(n-1)}$, for finishing this case we will apply an analogous reasoning as in Case A.2:

1. Firstly, one shows the existence of an $(N - \Gamma_{\mathcal{C},h})$-semirun that is both a min-rising and max-falling ℓ-embedding of $\beta^{(1)}\alpha^{(1)}$ with the same source and target configuration as $\beta^{(1)}\alpha^{(1)}$.

2. Secondly, let us assume that $\beta^{(2)}$ is an N-semirun from $q(x)$ to $q'(y)$ and that moreover $\alpha^{(2)}\sigma^{(2)}\alpha^{(4)}...\sigma^{(\lfloor n/2 \rfloor)}$ is an N-semirun from $q'(y)$ to $q''(z)$, with $y \in [B_{\max} - N, B_{\max} + \Upsilon_{\mathcal{C},h} - N]$ and $z \in [B_{\min} - \Upsilon_{\mathcal{C},h}, B_{\min}]$. Noting that $\beta^{(2)} = q(x) \xrightarrow{-p}_N q'(y)$, we explicitly lower $\beta^{(2)}$ into the $(N - \Gamma_{\mathcal{C},h})$-semirun $q(x) \xrightarrow{-p}_{N-\Gamma_{\mathcal{C},h}} q'(y + \Gamma_{\mathcal{C},h})$, which is — since $\beta^{(2)}$ is doubly-crossing — obviously both a min-rising and max-falling ℓ-embedding of $\beta^{(2)}$ from $q(x)$ to $q'(y + \Gamma_{\mathcal{C},h})$. Then one shows, as done in Case 2.A, the existence of an $(N - \Gamma_{\mathcal{C},h})$-semirun that is an ℓ-embedding of $\alpha^{(2)}\sigma^{(2)}\alpha^{(4)}...\sigma^{(\lfloor n/2 \rfloor)}$ all of whose counter values lie in $[\min(\rho) - \Gamma_{\mathcal{C},h}, \max(\rho) + \Gamma_{\mathcal{C},h}]$ from $q'(y + \Gamma_{\mathcal{C},h})$ to $q''(z)$ by subfactorizing each of the $\alpha^{(2i)}$ into $(N - \Gamma_{\mathcal{C},h})$-semiruns that have counter values in $[B_{\max} - N, B_{\min}]$ and into $(B_{\max} - N - 1)$-valleys, by recalling that $\sigma^{(i)}$ is a $B_{\max}$-hill for all $i \in [2, \lfloor n/2 \rfloor]$, and that

$$\begin{array}{rcl} z - y & \geq & N - (B_{\max} - B_{\min}) - 2\Upsilon_{\mathcal{C},h} \\ & \overset{(4.4)}{>} & N - (5/6 \cdot N + 2\Upsilon_{\mathcal{C}} + 4\Gamma_{\mathcal{C}} + 2) - 2\Upsilon_{\mathcal{C},h} \\ & = & N/6 - (5\Upsilon_{\mathcal{C},h} - 4\Gamma_{\mathcal{C},h} - 2) + \Upsilon_{\mathcal{C},h} \\ & > & M_{\mathcal{C},h}/6 - (5\Upsilon_{\mathcal{C},h} - 4\Gamma_{\mathcal{C},h} - 2) + \Upsilon_{\mathcal{C},h} \\ & > & \Upsilon_{\mathcal{C},h}. \end{array}$$

3. Finally (analogously as Point 1) one shows the existence of an $(N - \Gamma_{\mathcal{C},h})$-semirun that is both a min-rising and max-falling ℓ-embedding of $\alpha^{(n-1)}\beta^{(n)}$ with the same source and target configuration as $\alpha^{(n-1)}\beta^{(n)}$.

4.7 Proof of the Small Parameter Theorem

This section is devoted to proving the Small Parameter Theorem (Theorem 21).

For proving this let us fix some $N > M_{\mathcal{C}}$ and some accepting N-run π in $\mathcal{C}$ with $\textsc{Values}(\pi) \subseteq [0, h{\cdot}N]$ of the form

$$\pi \quad = \quad r_0(x_0) \quad \xrightarrow{\pi_0}_N \quad r_1(x_1) \quad \cdots \quad \xrightarrow{\pi_{n-1}}_N \quad r_n(x_n)$$

with $r_n \in F$. We will assume that accepting runs in $\mathcal{C}$ end with counter value 0 and hence, that $x_n = x_0 = 0$. We do not lose generality by making this assumption. Indeed, from every PTOCA $\mathcal{C}$, one can build a PTOCA $\mathcal{C}'$ with all its accepting runs ending in configuration with counter value 0 such that, for all $N \in \mathbb{N}$, there exists an accepting N-run in $\mathcal{C}$ with values in $[0, h \cdot N]$ if, and only if, there exists an accepting N-run in $\mathcal{C}'$ with values in $[0, h \cdot N]$. This is clear when one considers the construction $\mathcal{C}'$ obtained from $\mathcal{C}$ by adding two states r_- and r_f such that every final state of $\mathcal{C}$ has a ≥ 0 rule leading to r_-, r_- has a -1 rule that is a loop, and finally a $= 0$ rule to r_f, the only final state of $\mathcal{C}'$.

Starting from the accepting N-run π, we need to prove the existence of an accepting $(N - \Gamma_{\mathcal{C},h})$-run in $\mathcal{C}$. For every $a, b \in \mathbb{Q}$ with $a < b$ we define $[a, b[= \{c \in \mathbb{Q} \mid a \leq c < b\}$ and $]a, b] = \{c \in \mathbb{Q} \mid a < c \leq b\}$.

Since $\frac{N}{3} < N - \Gamma_{\mathcal{C},h}$, as $\Gamma_{\mathcal{C},h} < \frac{2M_{\mathcal{C},h}}{3} < \frac{2N}{3}$ by definition of the constants on page 33, the following claim is clear.

Claim 3. *Every subrun ρ of π with $\textsc{Values}(\rho) \subseteq [0, \frac{N}{3}[$ is already an $(N - \Gamma_{\mathcal{C},h})$-run.*

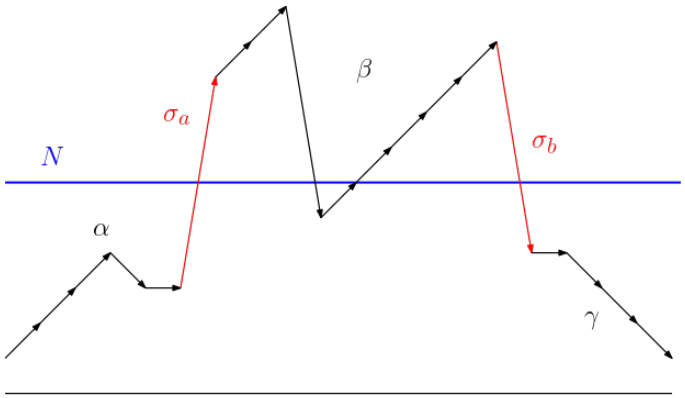

Figure 4.11: Illustration of the factorization (4.13).

We can therefore uniquely factorize π as

$$\pi \quad = \quad \rho^{(0)}\sigma^{(1)}\rho^{(1)} \quad \cdots \quad \sigma^{(m)}\rho^{(m)}, \tag{4.12}$$

where each $\rho^{(j)}$ satisfies $\textsc{Values}(\rho^{(j)}) \subseteq [0, \frac{N}{3}[$ and each $\sigma^{(j)}$ is some subrun $\pi[c,d]$ with $x_c < \frac{N}{3}$, $x_d < \frac{N}{3}$ and $x_k \geq \frac{N}{3}$ for all $k \in [c+1, d-1]$, where $[c+1, d-1] \neq \varnothing$.

To finish the proof of the Small Parameter Theorem (Theorem 21), by Claim 3 it thus suffices to prove the following statement for the rest of this section.

For every $N > M_{\mathcal{C},h}$ and every N-run

$$\sigma \quad = \quad q_0(z_0) \quad \xrightarrow{\sigma_0}_N \quad q_1(z_1) \quad \cdots \quad \xrightarrow{\sigma_{m-1}}_N \quad q_m(z_m)$$

satisfying $\textsc{Values}(\sigma) \subseteq [0, h \cdot N]$, $z_0, z_m < \frac{N}{3}$ and $z_i \geq \frac{N}{3}$ for all $i \in [1, m-1]$, there exists an $(N - \Gamma_{\mathcal{C},h})$-run from $q_0(z_0)$ to $q_m(z_m)$.

Let σ be such an N-run. Let us first assume that $z_i \geq N$ for some $i \in [1, m-1]$; the case $z_i < N$ for all $i \in [1, m-1]$ will be treated later. By this asumption, one can uniquely factorize σ — as seen in Figure 4.11 — as

$$\sigma \quad = \quad \alpha\ \sigma[a, a+1]\ \beta\ \sigma[b, b+1]\ \gamma, \tag{4.13}$$

where, for some $a, b \in [0, m-1]$,

- $\alpha = \sigma[0, a]$ is the maximal prefix of σ satisfying $\textsc{Values}(\alpha) \subseteq [0, N[$, in particular the transition $q_a(z_a) \xrightarrow{\sigma_a}_N q_{a+1}(z_{a+1})$ satisfies $z_a \in [0, N[$ and $z_{a+1} \in [N, h \cdot N]$,
- $\gamma = \sigma[b+1, m]$ is the maximal suffix of σ satisfying $\textsc{Values}(\gamma) \subseteq [0, N[$, i.e. the transition $q_b(z_b) \xrightarrow{\sigma_b}_N q_{b+1}(z_{b+1})$ satisfies $z_b \in [N, h\ \cdot N]$ and $z_{b+1} \in [0, N[$, and
- $\beta = \sigma[a+1, b]$ is the remaining infix of σ (note that $a+1 = b$ is possible).

We will apply the 5/6-Lemma (Lemma 41) to one of the subruns

$$\beta, \quad \sigma[a, a+1]\ \beta, \quad \beta\ \sigma[b, b+1], \text{ or } \quad \sigma[a, a+1]\ \beta\ \sigma[b, b+1],$$

hereby showing the existence of a suitable $(N - \Gamma_{\mathcal{C},h})$-semirun with same source and target configuration, respectively. We then shift this $(N - \Gamma_{\mathcal{C},h})$-semirun by $-\Gamma_{\mathcal{C},h}$ to obtain a suitable $(N - \Gamma_{\mathcal{C},h})$-run. To which of the subruns we will choose to apply the 5/6-Lemma will depend on

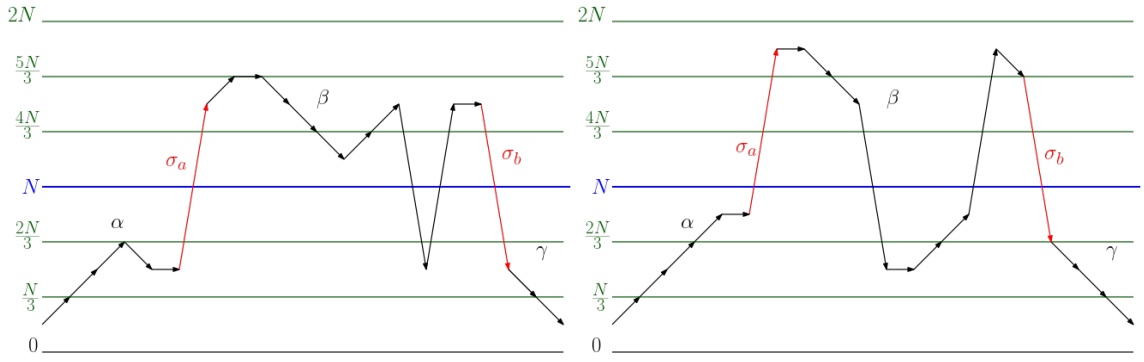

Figure 4.12: Illustration of Case 1, i.e. $z_{a+1}, z_b \in [N, \frac{5N}{3}[$, on the left, and Case 2, i.e. $z_{a+1}, z_b \in [\frac{5N}{3}, 2N[$, on the right.

the counter values z_a, z_{a+1}, z_b and z_{b+1}. For deciding this, we make a case distinction on which of the five intervals $\left\{\left[\frac{iN}{3}, \frac{(i+1)N}{3}\right[\ : i \in [1,5]\right\}$ they lie in, respectively.

Before the above-mentioned distinction on z_a, z_{a+1}, z_b and z_{b+1} we first claim that one can turn the possible resulting prefixes α and $\alpha\sigma[a, a+1]$ and possible suffixes $\sigma[b, b+1]\gamma$ and γ into $(N - \Gamma_{\mathcal{C},h})$-runs separately. The following claim tells us when these latter prefixes (resp. suffixes) can be turned into $(N - \Gamma_{\mathcal{C},h})$-runs whose target (resp. source) configuration has been shifted down by $\Gamma_{\mathcal{C},h}$.

Claim 4 (Possible lowering of the prefixes and suffixes)**.**

1. *If* $z_{a+1} \in [N, \frac{5N}{3}[$, *then there exists an* $(N - \Gamma_{\mathcal{C},h})$*-run from* $q_0(z_0)$ *to* $q_{a+1}(z_{a+1} - \Gamma_{\mathcal{C},h})$.
2. *If* $z_a \in [\frac{N}{3} + \Upsilon_{\mathcal{C},h}, N[$, *then there exists an* $(N - \Gamma_{\mathcal{C},h})$*-run from* $q_0(z_0)$ *to* $q_a(z_a - \Gamma_{\mathcal{C},h})$.
3. *If* $z_b \in [N, \frac{5N}{3}[$, *then there exists an* $(N - \Gamma_{\mathcal{C},h})$*-run from* $q_b(z_b - \Gamma_{\mathcal{C},h})$ *to* $q_m(z_m)$.
4. *If* $z_{b+1} \in [\frac{N}{3} + \Upsilon_{\mathcal{C},h}, N[$, *then there exists an* $(N - \Gamma_{\mathcal{C},h})$*-run from* $q_{b+1}(z_{b+1} - \Gamma_{\mathcal{C},h})$ *to* $q_m(z_m)$.

We postpone the proof of Claim 4 to the end of this section but refer to Figure 4.15 for an illustration of Points 1 and 2.

We can use Point (1) or Point (2) of the claim to turn the possible resulting prefixes $\alpha\sigma[a, a+1]$ or α respectively into $(N - \Gamma_{\mathcal{C},h})$-runs with target configuration shifted down by $\Gamma_{\mathcal{C},h}$. Symmetrically, we can use Point (3) or Point (4) of the claim to turn the possible resulting suffixes $\sigma[b, b+1]\gamma$ or γ respectively into $(N - \Gamma_{\mathcal{C},h})$-runs with source configuration shifted down by $\Gamma_{\mathcal{C},h}$. The claim will rely on the Depumping Lemma (Lemma 24) and on the fact that a transition with operation $+p$ or $-p$ has an absolute counter effect of N in an N-run but $N - \Gamma_{\mathcal{C},h}$ in an $(N - \Gamma_{\mathcal{C},h})$-run.

Let us for the moment assume $z_i \geq N$ for some $i \in [1, m-1]$ along with the factorization (4.13) of σ and Claim 4.

Assuming Claim 4 we conclude the proof by treating the following exhaustive cases on the positions of z_{a+1} and z_b separately.

Case 1. $z_{a+1}, z_b \in [N, \frac{5N}{3}[$, *cf. Figure 4.12.*

Recall that $\beta = \sigma[a+1, b]$ as defined in (4.13) is an N-run from $q_{a+1}(z_{a+1})$ to $q_b(z_b)$ satisfying $\textsc{Values}(\beta) \subseteq [\frac{N}{3}, h \cdot N]$. We view β as an N-semirun. We consider $\ell = N$ and observe that

$$\max(z_{a+1}, z_b, \ell) - \min(z_{a+1}, z_b, \ell) < \frac{5N}{3} - N = \frac{2N}{3} \leq \frac{5N}{6}.$$

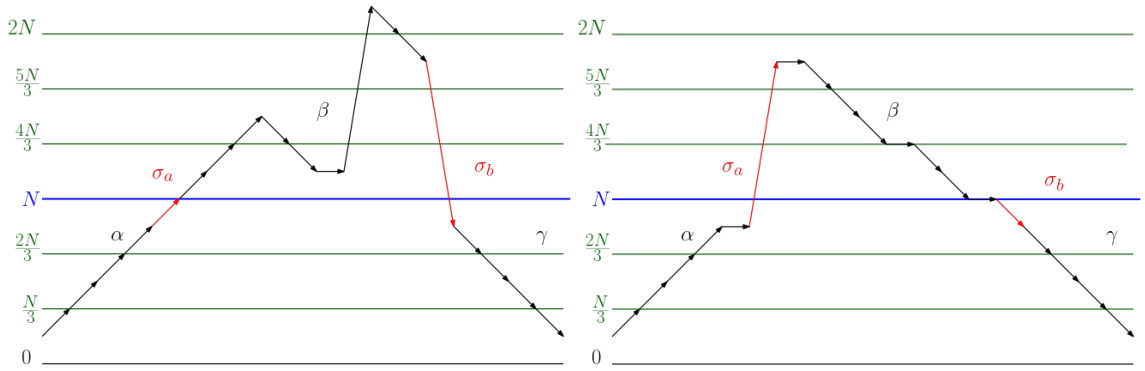

Figure 4.13: Illustration of Case 3, i.e. $z_{a+1} \in [N, \frac{4N}{3}[$ and $z_b \in [\frac{5N}{3}, 2N[$, on the left, and Case 4, i.e. $z_{a+1} \in [\frac{5N}{3}, 2N[$ and $z_b \in [N, \frac{4N}{3}[$, on the right.

Hence we can apply the 5/6-Lemma (Lemma 41) to β: there exists an $(N - \Gamma_{\mathcal{C},h})$-semirun $\widehat{\beta}$ from $q_{a+1}(z_{a+1})$ to $q_b(z_b)$ that is an N-embedding of β with $\textsc{Values}(\widehat{\beta}) \subseteq [\min(\beta) - \Gamma_{\mathcal{C},h}, \max(\beta) + \Gamma_{\mathcal{C},h}]$. Since moreover $\frac{N}{3} - 2\Gamma_{\mathcal{C},h} > \max(\mathsf{Consts}(\mathcal{C}))$, from $M_{\mathcal{C},h}$'s definition on page 33, and because $\min(\beta) \geq N/3$, it follows that $\widehat{\beta} - \Gamma_{\mathcal{C},h}$, the shifting of $\widehat{\beta}$ by $-\Gamma_{\mathcal{C},h}$, is in fact an $(N - \Gamma_{\mathcal{C},h})$-run from $p(z_{a+1} - \Gamma_{\mathcal{C},h})$ to $q_b(z_b - \Gamma_{\mathcal{C},h})$. It thus remains to show the existence of an $(N - \Gamma_{\mathcal{C},h})$-run from $q_0(z_0)$ to $q_{a+1}(z_{a+1} - \Gamma_{\mathcal{C},h})$ and one from $q_b(z_b - \Gamma_{\mathcal{C},h})$ to $q_m(z_m)$: the former follows from Point (1) of Claim 4, and the latter follows from Point (3) of Claim 4.

Case 2. $z_{a+1}, z_b \in [\frac{5N}{3}, 2N[$*, cf. Figure 4.12.*

It follows that $z_a, z_{b+1} \in [\frac{2N}{3}, N[$, and that σ_a and σ_b must be a $+p$ and $-p$ respectively. We apply the 5/6-Lemma (Lemma 41) to

$$\sigma[a, a+1] \ \beta \ \sigma[b, b+1]$$

with $\ell = N$, then shift the output by $-\Gamma_{\mathcal{C},h}$. Then we apply Point (2) of Claim 4 and Point (4) of Claim 4.

Case 3. $z_{a+1} \in [N, \frac{4N}{3}[$ *and* $z_b \in [\frac{5N}{3}, 2N[$*, cf. Figure 4.13.*

It follows that $z_{b+1} \in [\frac{2N}{3}, N[$. We apply the 5/6-Lemma (Lemma 41) to

$$\beta \ \sigma[b, b+1]$$

with $\ell = N$, then shift the output by $-\Gamma_{\mathcal{C},h}$. Then we apply Point (1) of Claim 4 and Point (4) of Claim 4.

Case 4. $z_{a+1} \in [\frac{5N}{3}, 2N[$ *and* $z_b \in [N, \frac{4N}{3}[$*, cf. Figure 4.13.*

It follows that $z_a \in [\frac{2N}{3}, N[$. We apply the 5/6-Lemma (Lemma 41) to

$$\sigma[a, a+1] \ \beta$$

with $\ell = N$, then shift the output by $-\Gamma_{\mathcal{C},h}$. Then we apply Point (2) of Claim 4 and Point (3) of Claim 4.

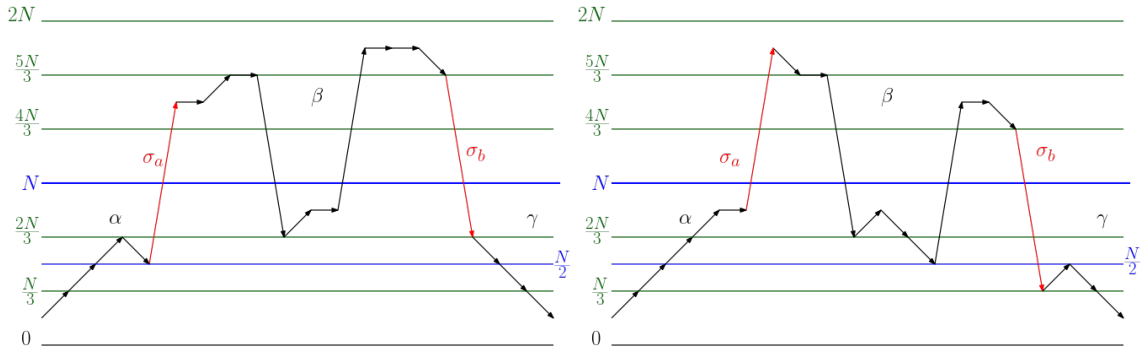

Figure 4.14: Illustration of Case 5, i.e. $z_{a+1} \in [\frac{4N}{3}, \frac{5N}{3}[$ and $z_b \in [\frac{5N}{3}, 2N[$, on the left (with moreover $z_a \notin [\frac{N}{3}, \frac{N}{2}[$), and Case 6, i.e. $z_{a+1} \in [\frac{5N}{3}, 2N[$ and $z_b \in [\frac{4N}{3}, \frac{5N}{3}[$, on the right (with moreover $z_{b+1} \in [\frac{N}{3}, \frac{N}{2}[$).

Case 5. $z_{a+1} \in [\frac{4N}{3}, \frac{5N}{3}[$ *and* $z_b \in [\frac{5N}{3}, 2N[$, *cf. Figure 4.14.*

It follows $z_{b+1} \in [\frac{2N}{3}, N[$, and that σ_a and σ_b must be a $+p$ and $-p$ respectively. We distinguish whether $z_a \in [\frac{N}{3}, \frac{N}{2}[$ or not.

Case 5.A. $z_a \notin [\frac{N}{3}, \frac{N}{2}[$.

It follows $z_a \in [\frac{N}{2}, N[$. We apply the 5/6-Lemma (Lemma 41) to

$$\sigma[a, a+1] \ \beta \ \sigma[b, b+1]$$

with $\ell = N$, then shift the output by $-\Gamma_{\mathcal{C},h}$. Then we apply Point (2) of Claim 4 and Point (4) of Claim 4.

Case 5.B. $z_a \in [\frac{N}{3}, \frac{N}{2}[$.

It follows $z_{a+1} \in [\frac{4N}{3}, \frac{3N}{2}[$. We apply the 5/6-Lemma (Lemma 41) to

$$\beta \ \sigma[b, b+1]$$

with $\ell = N$, then shift the output by $-\Gamma_{\mathcal{C},h}$. Then we apply Point (1) of Claim 4 and Point (4) of Claim 4.

Case 6. $z_{a+1} \in [\frac{5N}{3}, 2N[$ *and* $z_b \in [\frac{4N}{3}, \frac{5N}{3}[$, *cf. Figure 4.14.*

It follows $z_a \in [\frac{2N}{3}, N[$, and that σ_a and σ_b must be a $+p$ and $-p$ respectively. We distinguish whether $z_{b+1} \in [\frac{N}{3}, \frac{N}{2}[$ or not.

Case 6.A. $z_{b+1} \notin [\frac{N}{3}, \frac{N}{2}[$.

It follows $z_{b+1} \in [\frac{N}{2}, \frac{2N}{3}[$. We apply the 5/6-Lemma (Lemma 41) to

$$\sigma[a, a+1] \ \beta \ \sigma[b, b+1]$$

with $\ell = N$, then shift the output by $-\Gamma_{\mathcal{C},h}$. Then we apply Point (2) of Claim 4 and Point (4) of Claim 4.

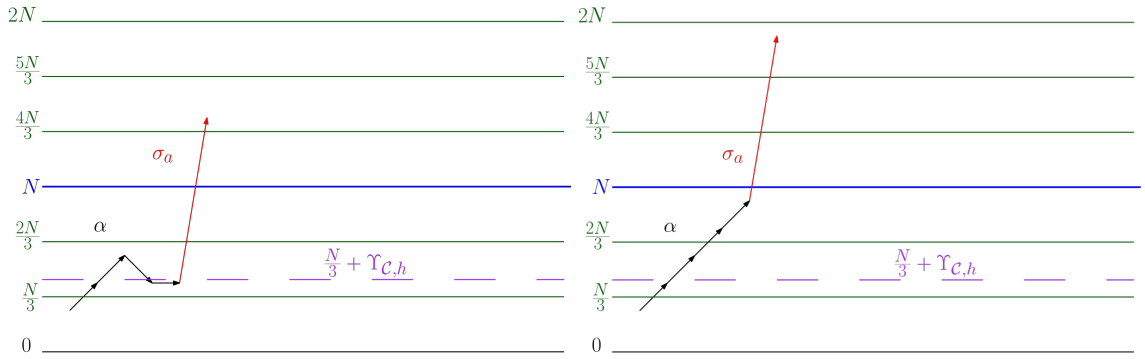

Figure 4.15: Claim 4: Examples for Point 1 (left) and Point 2 (right).

Case 6.B. $z_{b+1} \in [\frac{N}{3}, \frac{N}{2}[$.

It follows $z_b \in [\frac{4N}{3}, \frac{3N}{2}[$. We apply the 5/6-Lemma (Lemma 41) to

$$\sigma[a, a+1]\ \beta$$

with $\ell = N$, then shift the output by $-\Gamma_{\mathcal{C},h}$. Then we apply Point (2) of Claim 4 and Point (3) of Claim 4.

It remains to provide the proof of the Claim 4 before discussing the remaining case when $z_i < N$ for all $i \in [1, m-1]$

Proof of Claim 4. Let us only prove Points (1) and (2). Points (3) and (4) can be proven in a symmetrical manner as Points (1) and (2). Let us first prove Point (1), so let us assume that $z_{a+1} \in [N, \frac{5N}{3}[$. We refer to Figure 4.15 for an example of such a situation. Recall that $\alpha = \sigma[0, a]$, $z_0 < \frac{N}{3}$ and $z_i \in [\frac{N}{3}, N[$ for all $i \in [1, a]$.
We first factorize α, as seen in Figure 4.16, as

$$\alpha = \left(\prod_{i=1}^{t} \chi_i\right) \zeta,$$

where

- each χ_i is a subrun of α that either
  a) starts in a configuration with counter value strictly less than $N - \Gamma_{\mathcal{C},h} - \Upsilon_{\mathcal{C},h}$ and ends in the first next configuration with counter value at least $N - \Gamma_{\mathcal{C},h}$, or conversely
  b) starts in a configuration with counter value at least $N - \Gamma_{\mathcal{C},h}$ and ends in the first next configuration with counter value strictly less than $N - \Gamma_{\mathcal{C},h} - \Upsilon_{\mathcal{C},h}$, and
- the (possibly empty) suffix ζ's prefixes are neither of form a) nor b), i.e. $\textsc{Values}(\zeta) \subseteq [0, N - \Gamma_{\mathcal{C},h}[$ or $\textsc{Values}(\zeta) \subseteq [N - \Gamma_{\mathcal{C},h} - \Upsilon_{\mathcal{C},h}, N[$.

First observe that α and hence in particular $\chi_1, \ldots, \chi_t, \zeta$ all do not contain any $+p$-transition nor any $-p$-transition, and that $|\Delta(\chi_i)| > \Upsilon_h$ for all $i \in [1, t]$ by definition.

Next observe that $t = 0$ is possible; in this case we have $\alpha = \zeta$ and $\textsc{Values}(\alpha) \subseteq [0, N - \Gamma_{\mathcal{C},h}[$.

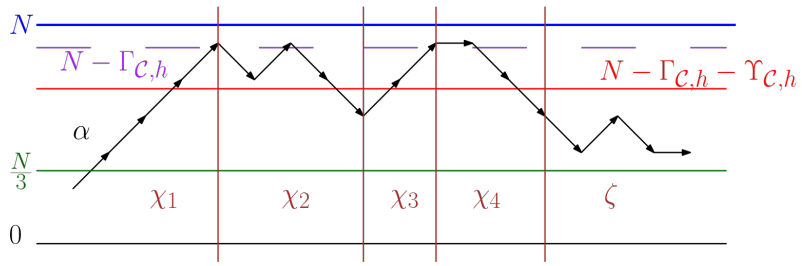

Figure 4.16: Illustration of the factors χ_1, χ_2, χ_3, χ_4 and ζ of α.

We will however first treat the case $t > 0$, the case $t = 0$ will be treated later. It follows from $z_0 < \frac{N}{3}$ that χ_1 must be of type a); more generally, χ_i is of type a) for all odd $i \in [1,t]$ and of type b) for all even $i \in [1,t]$. Since $z_0 < \frac{N}{3}$, observe that if for α's last counter value z_a we have $z_a \in [N - \Gamma_{\mathcal{C},h}, N[$, then t must be odd, and, similarly (but not entirely dually), if for α's last counter value z_a we have $z_a \in [\frac{N}{3}, N - \Gamma_{\mathcal{C},h} - \Upsilon_{\mathcal{C},h}[$, then t must be even.
In the following we prefer to write α as $\alpha = \alpha[0,a]$ rather than $\sigma[0,a]$. It is important to recall that $z_0 < \frac{N}{3}$ and $z_s \in [\frac{N}{3}, N[$ for all $s \in [1,a]$.

Let

$$q_0(z_0) = q_{j_1}(z_{j_1}) \xrightarrow{\chi_1} q_{j_2}(z_{j_2}) \xrightarrow{\chi_2} q_{j_3}(z_{j_3}) \quad \cdots \quad \xrightarrow{\chi_{t-1}} q_{j_t}(z_{j_t}) \xrightarrow{\chi_t} q_{j_{t+1}}(z_{j_{t+1}}) \xrightarrow{\zeta} q_{j_{t+2}}(z_{j_{t+2}}).$$

Note that $j_{t+1} = j_{t+2}$ is possible if ζ is empty. In the following, we will first show how to turn any χ_i of type a) (resp. b)) into an $(N - \Gamma_{\mathcal{C},h})$-run with target (resp. source) configuration shifted down by $\Gamma_{\mathcal{C},h}$, and then we make a case distinction on how to end the proof based on the parity of t.

Subclaim 1. *Let $i \in [1,t]$ be odd. Then there exists an $(N - \Gamma_{\mathcal{C},h})$-run $\widehat{\chi_i}$ from $q_{j_i}(z_{j_i})$ to $q_{j_{i+1}}(z_{j_{i+1}} - \Gamma_{\mathcal{C},h})$.*

Proof of Subclaim 1. Indeed, $\phi(\chi_i) = \varepsilon \in \Lambda_{2h}$, as α contains neither $+p$-transitions nor $-p$-transitions. Since moreover $\Delta(\chi_i) > \Upsilon_{\mathcal{C},h}$ we can now apply Lemma 24 to χ_i (viewed as an N-semirun) and obtain an N-semirun $\widehat{\chi_i}$ with $\Delta(\widehat{\chi_i}) = \Delta(\chi_i) - \Gamma_{\mathcal{C},h}$ that is such that $\widehat{\chi_i} = \alpha[j_i, j_{i+1}] - I_1 - I_2 \cdots - I_k$ for pairwise disjoint intervals $I_1, \ldots, I_k \subseteq [j_i, j_{i+1}]$ such that

- $\phi(\alpha[I_h]) \in \Lambda_{4h}$,
- $\Delta(\alpha[I_h]) \in Z_{\mathcal{C}}\mathbb{Z}$ and $\Delta(\alpha[I_h]) > 0$ for all $h \in [1,k]$.

Recall $\textsc{Values}(\alpha[1,a]) \subseteq [\frac{N}{3}, N[$ and $\frac{N}{3} - \Gamma_{\mathcal{C},h} > \frac{M_{\mathcal{C},h}}{3} - \Gamma_{\mathcal{C},h} > \Gamma_{\mathcal{C},h} > \max(\mathsf{Consts}(\mathcal{C}))$, where the inequalities follows from $M_{\mathcal{C},h}$'s and $\Gamma_{\mathcal{C},h}$'s definition on page 33. It follows that $\widehat{\chi_i}$ has all its counter values (except for the first one) in $[\frac{N}{3} - \Gamma_{\mathcal{C},h}, N - \Gamma_{\mathcal{C},h}[$. Moreover, the first transition's operation must be a +1 update and therefore cannot be a test, and hence $\widehat{\chi_i}$ is an $(N - \Gamma_{\mathcal{C},h})$-run from $q_{j_i}(z_{j_i})$ to $q_{j_{i+1}}(z_{j_{i+1}} - \Gamma_{\mathcal{C},h})$.

□

Subclaim 2. *Let $i \in [1,t]$ be even. Then there exists an $(N - \Gamma_{\mathcal{C},h})$-run $\widehat{\chi_i}$ from $q_{j_i}(z_{j_i} - \Gamma_{\mathcal{C},h})$ to $q_{j_{i+1}}(z_{j_{i+1}})$.*

Proof of Subclaim 2. Analogously, by use of Lemma 24, for $i \in [1,t]$ even, there exists an $(N - \Gamma_{\mathcal{C},h})$-semirun χ' from $q_{j_i}(z_{j_i})$ to $q_{j_{i+1}}(z_{j_{i+1}} + \Gamma_{\mathcal{C},h})$, from which we obtain an $(N - \Gamma_{\mathcal{C},h})$-semirun $\widehat{\chi_i} = \chi' - \Gamma_{\mathcal{C},h}$ from $q_{j_i}(z_{j_i} - \Gamma_{\mathcal{C},h})$ to $q_{j_{i+1}}(z_{j_{i+1}})$ by shifting χ' by $-\Gamma_{\mathcal{C},h}$. Moreover, as $\textsc{Values}(\alpha[1,a]) \subseteq [\frac{N}{3}, N[$ and $\frac{N}{3} - \Gamma_{\mathcal{C},h} > \Gamma_{\mathcal{C},h} > \max(\mathsf{Consts}(\mathcal{C}))$ as seen in the proof of Subclaim 1, $\widehat{\chi_i}$ is an $(N - \Gamma_{\mathcal{C},h})$-run as required.

□

To finish the proof of the existence of an $(N-\Gamma_{\mathcal{C},h})$-run from $q_0(z_0)$ to $q_{a+1}(z_{a+1}-\Gamma_{\mathcal{C},h})$ we make a case distinction on the parity of t.

Assume first that the parity of t is odd. By applying Subclaims 1 and 2 to the runs $\chi_1,\dots,\chi_t$ appropriately we obtain the $(N-\Gamma_{\mathcal{C},h})$-run

$$\widehat{\chi_1}\cdots\widehat{\chi_t}$$

from $q_{j_1}(z_{j_1})$ to $q_{j_{t+1}}(z_{j_{t+1}}-\Gamma_{\mathcal{C},h})$. Since t is odd we have that χ_t is of type a), $z_{j_{t+1}}\in[N-\Gamma_{\mathcal{C},h},N[$ and $\textsc{Values}(\zeta)\subseteq[N-\Gamma_{\mathcal{C},h}-\Upsilon_{\mathcal{C},h},N[$. As $(N-\Gamma_{\mathcal{C},h}-\Upsilon_{\mathcal{C},h})-\Gamma_{\mathcal{C},h}>(M_{\mathcal{C},h}-\Gamma_{\mathcal{C},h}-\Upsilon_{\mathcal{C},h})-\Gamma_{\mathcal{C},h}>\max(\mathsf{Consts}(\mathcal{C}))$, following from $M_{\mathcal{C},h}$'s, $\Gamma_{\mathcal{C},h}$'s and $\Upsilon_{\mathcal{C},h}$'s definition on page 33, and $\phi(\alpha)=\varepsilon$, it follows that

$$\widehat{\chi_1}\cdots\widehat{\chi_t}\,(\zeta-\Gamma_{\mathcal{C},h})$$

is an $(N-\Gamma_{\mathcal{C},h})$-run from $q_{j_1}(z_{j_1})$ to $q_{j_{t+2}}(z_{j_{t+2}}-\Gamma_{\mathcal{C},h})$. Recall that $z_{a+1}<\frac{5N}{3}$ by case assumption and also recall that $N>M_{\mathcal{C},h}$. Since $\textsc{Values}(\zeta)\subseteq[N-\Gamma_{\mathcal{C},h}-\Upsilon_{\mathcal{C},h},N[$ we have $z_a=z_{j_{t+2}}\geq N-\Gamma_{\mathcal{C},h}-\Upsilon_{\mathcal{C},h}$ and hence $0<\Delta(\sigma,a)<\frac{5N}{3}-(N-\Gamma_{\mathcal{C},h}-\Upsilon_{\mathcal{C},h})\leq\frac{2N}{3}+\Gamma_{\mathcal{C},h}+\Upsilon_{\mathcal{C},h}<\frac{2N}{3}+\frac{M_{\mathcal{C},h}}{3}<N$, where the penultimate inequality follows from the definition of $M_{\mathcal{C},h}$ on page 33. Hence, as σ_a is not a test nor a $+p$-transition we have that

$$\widehat{\chi_1}\cdots\widehat{\chi_t}\,(\zeta-\Gamma_{\mathcal{C},h})\,(\sigma[a,a+1]-\Gamma_{\mathcal{C},h})$$

is an $(N-\Gamma_{\mathcal{C},h})$-run from $q_{j_1}(z_{j_1})=q_0(z_0)$ to $q_{j_{t+2}}(z_{j_{t+2}}-\Gamma_{\mathcal{C},h})=q_{a+1}(z_{a+1}-\Gamma_{\mathcal{C},h})$ as required.

Let us now treat the case when t is even. It follows $\textsc{Values}(\zeta)\subseteq[0,N-\Gamma_{\mathcal{C},h}[$, in particular $z_a\in[0,N-\Gamma_{\mathcal{C},h}[$. Again,

$$\widehat{\chi_1}\cdots\widehat{\chi_t}$$

is an $(N-\Gamma_{\mathcal{C},h})$-run from $q_{j_1}(z_{j_1})=q_0(z_0)$ to $q_{j_{t+1}}(z_{j_{t+1}})$. Since $z_{a+1}\geq N$ and $z_a<N-\Gamma_{\mathcal{C},h}$ it follows that σ_a is a $+p$-transition, in particular $\Delta(\sigma,a)>\Gamma_{\mathcal{C},h}$. Thus,

$$\widehat{\chi_1}\cdots\widehat{\chi_t}\zeta\tau,$$

where $\tau=q_a(z_a)\xrightarrow{\sigma_a}_{N-\Gamma_{\mathcal{C},h}}q_{a+1}(z_{a+1}-\Gamma_{\mathcal{C},h})$ with $\Delta(\tau)=\Delta(\sigma,a)-\Gamma_{\mathcal{C},h}=N-\Gamma_{\mathcal{C},h}$, is an $(N-\Gamma_{\mathcal{C},h})$-run from $q_{j_1}(z_{j_1})=q_0(z_0)$ to $q_{j_{t+2}}(z_{j_{t+2}}-\Gamma_{\mathcal{C},h})=q_{a+1}(z_{a+1}-\Gamma_{\mathcal{C},h})$, as required.

It remains to discuss the case when $t=0$. This case can be proven analogously. Indeed, from $t=0$ it follows immediately that $\alpha=\zeta$ and $\textsc{Values}(\zeta)\subseteq[0,N-\Gamma_{\mathcal{C},h}[$ and the proof is analogous as the case when $t>0$ and when t is even.

Let us now sketch the proof of Point (2) of Claim 4. Let us assume $z_a\in[\frac{N}{3}+\Upsilon_{\mathcal{C},h},N[$. Similarly as in Point (1) we can factorize α as $\alpha=\left(\prod_{i=1}^{t}\chi_i\right)\zeta$ and Subclaims 1 and 2 hold again.

If t is odd, then by Subclaims 1 and 2 we have that the run $\left(\prod_{i=1}^{t}\widehat{\chi_i}\right)(\zeta-\Gamma_{\mathcal{C},h})$, stipulating that $\widehat{\chi_i}$ is the of Subclaims 1 and 2 respectively (depending on the parity of i), is the desired $(N-\Gamma_{\mathcal{C},h})$-run from $q_0(z_0)$ to $q_a(z_a-\Gamma_{\mathcal{C},h})$.

If t is even, then again by Subclaims 1 and 2 we have that $\xi=\left(\prod_{i=1}^{t}\widehat{\chi_i}\right)\zeta$ is an $(N-\Gamma_{\mathcal{C},h})$-run from $q_0(z_0)$ to $q_a(z_a)$, where again $\widehat{\chi_i}$ is defined as above. By definition the run ξ does not contain any $+p$-transitions nor $-p$-transitions, thus $\phi(\xi)=\varepsilon\in\Lambda_{2h}$. By construction also the run ξ has all counter values, besides the first, above $\frac{N}{3}-\Gamma_{\mathcal{C},h}>\Gamma_{\mathcal{C},h}+\max(\mathsf{Consts}(\mathcal{C}))$. Moreover, as $z_a\geq\frac{N}{3}+\Upsilon_{\mathcal{C},h}$ and $z_0<\frac{N}{3}$, we have $\Delta(\xi)>\Upsilon_{\mathcal{C},h}$. We can thus apply Lemma 24 to ξ, obtaining an $(N-\Gamma_{\mathcal{C},h})$-run ξ' from $q_0(z_0)$ to $q_a(z_a-\Gamma_{\mathcal{C},h})$, as required.

□

We now conclude the proof of our statement by treating the only remaining case, the case when σ is such that $z_i<N$ for all $i\in[1,m-1]$. In this case we can factorize σ as $\sigma=\prod_{i=1}^{t}\chi_i\zeta$ similarly as done in the proof of Point (1) of Claim 4, where t is even, and analogously prove that $\prod_{i=1}^{t}\widehat{\chi_i}\zeta$ is an $(N-\Gamma_{\mathcal{C},h})$-run from $q_0(z_0)$ to $q_m(z_m)$, where $\widehat{\chi_i}$ is the output of Subclaim 1 and 2 respectively (depending on the parity of i).

4.8 Discussion and open problems

In this section we have shown that the reachability problem for a subset of PTOCA lie in PSPACE. We considered h-bounded 1-PTOCA, that is, 1-PTOCA such that for all $N \in \mathbb{N}$, all accepting N-runs π satisfy $\textsc{Values}(\pi) \subseteq [0, h \cdot N]$. For any such PTOCA, a repeated application of our Small Parameter Theorem (Theorem 21) allows to conclude that the PTOCA has an accepting N-run if, and only if, there exists an accepting N'-run for some N' that is at most exponential in the sum of h and the size of the PTOCA. This exponential upper bound on the parameter value led to a PSPACE upper bound for h-BOUNDED 1-PTOCA REACHABILITY.

For proving the Small Parameter Theorem, we introduced the notion of semiruns and gave several techniques for manipulating them. The Depumping Lemma (Lemma 24) allowed us to construct from semiruns with large absolute counter effect new semiruns with a smaller absolute counter effect. The Bracket Lemma (Lemma 25) allowed us to find in semiruns having a sufficiently large absolute counter effect and satisfying some majority condition on the number of occurrences of $+p$-transitions and $-p$-transitions some subsemirun that has again a large absolute counter effect and moreover some bracketing properties. Our Hill and Valley Lemma (Lemma 30) allowed to turn, for sufficiently large N, any N-semirun that is either a hill or a valley into an N'-semirun for some $N' < N$. Our 5/6-Lemma (Lemma 41) allowed to turn for sufficiently large N any N-semirun with an absolute counter effect of at most $5/6 \cdot N$ into an N'-semirun for some $N' < N$.

We hope that our techniques can be extended for analysing 1-PTOCA REACHABILITY. Of note is that the decidablity proof from [17] starts with only considering N-runs π with $\textsc{Values}(\pi) \subseteq [0, 2{\cdot}N]$ before relaxing the restriction. Bundala and Ouaknine proved by a careful factorization (Theorem 10.13 in [17]) that for a PTOCA $\mathcal{C}$ without modulo test, there exists bounds L and h depending only on $\mathcal{C}$ such that for N sufficiently large there exists an accepting N-run only if there exists one which have all counter values bellow $h \cdot N$ exept for at most L maximal subruns — all of which can be expressed as N-runs in a particular POCA. Since their reduction is towards existential Presburger arithmetic with divisibility, Bundala and Ouaknine then proceed with proving that there exists ∃PAD formula for defining these finitely many "out of bounds" subruns, then extend the techniques to support having operations of the form "$= 0 \bmod c$". How to deal more efficiently with such subruns remains an open problem, but we are convinced that the machinery and techniques developed therein will prove useful for exploring the complexity of the problem.

We also hope that extensions of our techniques provide a line of attack for analysing PTOCA REACHABILITY in general. When analysing runs in PTOCA that involves an arbitrary number of parameters, it will become necessary to "de-scale" semiruns in the following sense. Already in the presence of two parameters one can see that it becomes necessary to decrease the value of both parameters simultaneously proportionally: for instance one can build a 2-PTOCA for which there exists an accepting μ-run only if the first parameter is assigned by μ a multiple of the value assigned to the second parameter. How our techniques can be extended to handle such obstacles remains yet to be explored.

It is worth noting that reachability is not the only problem one can consider with regards to PTOCA, as one can also explore the complexity of PTOCA reachability games. As OCA can be seen as special cases of PDA, upper bounds generally follow from known results for PDA. This however is not the case in the parametric extension considered in this thesis, as parametric updates in PTOCA are not trivially handled by our PPDA model.

Perhaps closer to the study of the problem of solving parity games for PTOCA, [47] studies the computational complexity of model checking CTL and LTL on SOCA and POCA with respect to data and combined complexity. As mentioned in Section 2 and Section 6, model checking logics is tied to the study of games, for instance the μ-calculus model checking problem is polynomially equivalent to the solution of a parity game in the case of PDA. Since the data and combined complexity of the μ-calculus on POCA is Π_1^0-complete, where Π_1^0 denotes the class of all languages whose complements are recursively enumerable, it is conceivable that the Π_1^0-completeness result could be extended to PTOCA parity games.

Chapter 5

Parametric timed automata

This chapter studies the computational complexity of reachability in classes of parametric timed automata. Parametric timed automata (PTA) have been introduced by Alur, Henzinger, and Vardi as an extension of timed automata in which clocks can be compared against parameters. They serve as a more general means to specify the behavior of under-specified systems, compared to timed automata, by allowing clocks to be compared against parameters that can take unspecified non-negative integer values. The *reachability problem for PTA* in turn asks for the existence of an assignment of the parameters to the non-negative integers such that there exists an execution that ends in a state belonging to a set of final states in the resulting timed automaton.

We call a clock of a PTA *parametric* when there is at least one rule of the automaton where it is compared against a parameter. It has been shown in [5] that already for PTA that contain *three parametric clocks* reachability is undecidable — even in the presence of a single parameter [10]. To the contrary, Alur, Henzinger and Vardi have shown in [5] that reachability is decidable for PTA that contain only *one parametric clock*, yet by an algorithm whose running time is nonelementary.

Recently, there has been some advances in the study of the decidability and complexity status of the reachability problem for PTA with one or two parametric clocks only. In the case of one parametric clock, Bundala and Ouaknine have shown a first elementary complexity upper bound for the reachability problem, providing a NEXP lower bound and a 2NEXP upper bound [17]. The gap was closed by Beneš et al. in [10] who have shown a matching NEXP upper bound. (also in the continuous time setting), we refer to [12] for an alternative proof by Bollig, Quaas and Sangnier using alternating two-way automata.

In the case of two parametric clocks, Bundala and Ouaknine [17] have shown that in the presence of one parameter the reachability problem is decidable and hard for the complexity class $\mathsf{PSPACE}^{\mathsf{NEXP}}$. For showing the above-mentioned decidability result [17] provides a reduction from PTA over two parametric clocks to a suitable formalism of parametric one-counter automata. Such an approach via parametric one-counter automata has already successfully been applied to model checking freeze-LTL as shown by Demri and Sangnier [33] and Lechner et al. [74], yet notably over a weaker model of parametric one-counter automata than the one introduced in [17].

After providing more formal definitions of parametric timed automata and the reachability problem, we give an overview of the existing literature and of our contribution. We prove our result, namely that reachability for parametric timed automata with one parameter and two parametric clocks is EXPSPACE-complete, and then close the section with a discussion about the methods used along with some directions for future work.

5.1 Definitions

A *parametric guard* over a finite set of clocks Ω and a finite set of parameters P is a comparison of the form $g = \omega \bowtie p$, where $\omega \in \Omega$, $p \in P$, and $\bowtie \in \{<, \leq, =, \geq, >\}$. We will now call elements g in $\mathcal{G}(\Omega)$ *non-parametric guards*. We denote by $\mathcal{G}(\Omega, P)$ the *set of parametric and non-parametric*

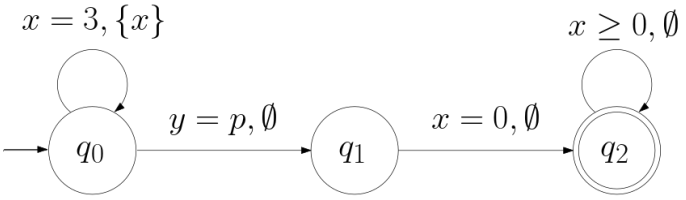

Figure 5.1: An example of a PTA. The automaton consists of three states, the set of clocks is $\{x, y\}$, the set of parameters is $\{p\}$. The edges are represented by arrows labeled with the corresponding guard and the set of clocks U to be reset. A parameter valuation μ witnesses that there exists an accepting μ-run for this PTA if, and only if, $\mu(p) \in 3\mathbb{Z}$.

guards over the set of clocks Ω and the set of parameters P.

The *size* $|g|$ of a guard $g = \omega \bowtie e$ is defined as

$$|g| = \begin{cases} \log(e) & \text{if } e \in \mathbb{N}, \\ 1 & \text{otherwise.} \end{cases}$$

A *clock valuation* is a function from Ω to $\mathbb{N}$; we write $\vec{0}$ to denote the clock valuation $\omega \mapsto 0$. For each clock valuation v and each $t \in \mathbb{N}$ we denote by $v + t$ the clock valuation $\omega \mapsto v(\omega) + t$. For every guard $g = \omega \bowtie p$ with $p \in P$ (resp. $g = \omega \bowtie k$ with $k \in \mathbb{N}$) we write $v \vDash_\mu g$ if $v(\omega) \bowtie \mu(p)$ (resp. $v(\omega) \bowtie k$; in this case we may also simply write $v \vDash g$). We define an *empty guard* g_ϵ over a non-empty finite set of clocks Ω and a finite set of parameters P to be of the form $\omega \geq 0$ for some $\omega \in \Omega$. In particular, we define g_ϵ such that for all $v \in \mathbb{N}^\Omega$ and all $\mu \in \mathbb{N}^P$ we have $v \vDash_\mu g_\epsilon$, hence g_ϵ can be used as a guard that is always true.

A parametric timed automaton as introduced in [5] is a finite automaton extended with a finite set of parameters P and a finite set of clocks Ω that all progress at the same rate and that can be individually reset to zero. Moreover, every transition is labeled by a guard over Ω and P and by a set of clocks to be reset.

Formally, a *parametric timed automaton* (*PTA* for short) is a tuple $\mathcal{A} = (Q, \Omega, P, R, q_{init}, F)$, where

- Q is a non-empty finite *set of states*,
- Ω is a non-empty finite *set of clocks*,
- P is a finite *set of parameters*,
- $R \subseteq Q \times \mathcal{G}(\Omega, P) \times \mathscr{P}(\Omega) \times Q$ is a finite *set of rules*,
- $q_{init} \in Q$ is an *initial state*, and
- $F \subseteq Q$ is a *set of final states*.

A clock $\omega \in \Omega$ is called *parametric* if there exists some $(q, g, U, q') \in R$ such that the guard g is of the form $\omega \bowtie p$, with $\bowtie \in \{<, \leq, =, \geq, >\}$ and $p \in P$. We also refer to $\mathcal{A}$ as an (m, n)-PTA if $m = |\{\omega \in \Omega \mid \omega \text{ is parametric}\}|$ is the number of parametric clocks and $n = |P|$ is the number of parameters of $\mathcal{A}$ — sometimes we also just write $(m, *)$-PTA (resp. $(*, n)$-PTA) when n (resp. m) is a priori not fixed.

The *size* of $\mathcal{A}$ is defined as

$$|\mathcal{A}| \;=\; |Q| + |\Omega| + |P| + |R| + \sum_{(q,g,U,q') \in R} |g|.$$

Let $\mathsf{Consts}(\mathcal{A}) = \{c \in \mathbb{N} \mid \exists (q, g, U, q') \in R,\ \exists \omega \in \Omega : g = \omega \bowtie k\}$ denote the set of constants that appear in the guards of the rules of $\mathcal{A}$.

By $\mathsf{Conf}(\mathcal{A}) = Q \times \mathbb{N}^\Omega$ we denote the set of *configurations* of $\mathcal{A}$. We prefer however to denote a configuration by $q(v)$ instead of (q, v).

A parametric timed automaton $\mathcal{A} = (Q, \Omega, P, R, q_{init}, F)$ and a parameter valuation $\mu : P \to \mathbb{N}$ induce the labeled transition system $T^\mu_\mathcal{A} = (\mathsf{Conf}(\mathcal{A}), \lambda_\mathcal{A}, \to_{\mathcal{A},\mu})$ where $\lambda_\mathcal{A} = R \times \mathbb{N}$ and where $\to_{\mathcal{A},\mu}$ is defined such that, for all $q(z), q'(z') \in \mathsf{Conf}(\mathcal{A})$, for all $(\delta, t) \in R \times \mathbb{N}$ with $\delta = (q, g, U, q') \in R$, $q(v) \xrightarrow{\delta,t}_{\mathcal{A},\mu} q'(v')$ if $v + t \vDash_\mu g$, $v'(u) = 0$ for all $u \in U$ and $v'(\omega) = v(\omega) + t$ for all $\omega \in \Omega \setminus U$.

Let $\mu : P \to \mathbb{N}$ be a parameter valuation. A *μ-run* from $q_0(v_0)$ to $q_n(v_n)$ in $\mathcal{A}$ is a path in $T^\mu_\mathcal{A}$, that is, a sequence

$$q_0(v_0) \xrightarrow{\delta_1,t_1}_{\mathcal{A},\mu} q_1(v_1) \quad \cdots \quad \xrightarrow{\delta_n,t_n}_{\mathcal{A},\mu} q_n(v_n).$$

It is called *reset-free* if the set appearing in the third component is empty for all δ_i. We say π is *accepting* if $q_0(v_0) = q_{init}(\vec{0})$ and $q_n \in F$. We refer to Figure 5.1 for an instance of a PTA for which there exists an accepting μ-run for some $\mu \in \mathbb{N}^P$.

As before, in the particular case where $P = \{p\}$ is a singleton for some parameter p and $\mu(p) = N$ we prefer to write $q(v) \to_{\mathcal{A},N} q'(v')$ to denote $q(v) \to_{\mathcal{A},\mu} q'(v')$ and will call the μ-run an *N-run*. We also prefer to write $\vDash_N$ to denote $\vDash_\mu$. In case the automaton $\mathcal{A}$ is obvious from context, we write $\to_\mu$ (resp. $\to_N$) instead of $\to_{\mathcal{A},\mu}$ (resp. $\to_{\mathcal{A},N}$).

We are interested in the following decision problem.

(m,n)-PTA REACHABILITY

INPUT: An (m,n)-PTA $\mathcal{A}$.
QUESTION: Does there exists $\mu \in \mathbb{N}^P$ such that there exists an accepting μ-run in $\mathcal{A}$?

We just write PTA REACHABILITY when neither $m = |\{\omega \in \Omega \mid \omega \text{ is parametric}\}|$ nor $n = |P|$ is a priori fixed for all input PTA. Alur et al. have already shown in their seminal paper that PTA REACHABILITY is in general undecidable, already in the presence of only three parametric clocks [5]. Beneš et al. strengthened this when only one parameter is present [10].

Theorem 50. *[10]* $(3,1)$-PTA REACHABILITY *is undecidable.*

On the positive side, $(1,*)$-PTA REACHABILITY has recently been shown to be complete for NEXP, where a nonelementary upper bound was initially given by Alur et al. [5].

Theorem 51. *[17, 10, 12]* $(1,*)$-PTA REACHABILITY *is* NEXP*-complete.*

On the other end, decidability of $(2,*)$-PTA REACHABILITY is still considered to be a challenging open problem to the best of our knowledge. In the presence of one parameter the following has recently been proven.

Theorem 52. *[17]* $(2,1)$-PTA REACHABILITY *is decidable and* $\mathsf{PSPACE}^{\mathsf{NEXP}}$*-hard.*

5.1.1 Contribution

The following theorem states our main result concerning PTA reachability.

Theorem 53. $(2,1)$-PTA REACHABILITY *is* EXPSPACE*-complete.*

Our contribution is two-fold.

Inspired by [47, 49], for the EXPSPACE lower bound we make use of deep results from complexity theory, namely a serializability characterization of EXPSPACE (in turn originally based on Barrington's Theorem [9]) and a logspace translation of numbers in Chinese Remainder Representation to binary representation due to Chiu, Davida, and Litow [24]. We provide a programming language that we show can simulate serializability computations. It is then shown that with small PTA over two parametric clocks and one parameter one can simulate the programming language.

For the EXPSPACE upper bound, we first give a careful exponential time reduction from PTA over two parametric clocks and one parameter to a (slight subclass of) parametric one-counter automata over one parameter based on a minor adjustment of a construction due to Bundala and Ouaknine [17]. In solving the reachability problem for parametric one-counter automata with one parameter, we refer to the results from Chapter 4, that allows us to prove that it is sufficient to consider a parameter value of exponential magnitude. This allows us to show a doubly-exponential upper bound on the value of the only parameter of PTA with two parametric clocks and one parameter.

Like the results in [5], our results hold for PTA over discrete time, where it is worth mentioning that in [5] parameters can both be integer-valued and rational-valued. For PTA with closed (i.e., non-strict) clock constraints and parameters ranging over integers, techniques [58, 83] exist that allow to reduce the reachability problem over continuous time to discrete time. There is a plethora of variants of PTA that have recently been studied, we refer to [6] for an extensive overview by André.

5.1.2 Overview

In Section 5.2 we introduce some programming language that can perform serializability computations. We then introduce some auxiliary gadgets that we build upon to show that small PTA over two parametric clocks and one parameter can simulate the computations from the programming language. Section 5.3 is devoted to the proof of the reduction from PTA over two parametric clocks and one parameter to parametric one-counter automata over one parameter, and how this reduction, combined with Theorem 21, leads to an EXPSPACE upper bound for $(2,1)$-PTA REACHABILITY.

5.2 An EXPSPACE lower bound via serializability

In this section, we show an EXPSPACE lower bound for $(2,1)$-PTA REACHABILITY using a serializability characterization of EXPSPACE. The proof first appeared in the paper [48], taking inspiration from [47, 49]. We remark here that the technique used is slightly more powerful than required. Indeed it was shown that with small PTA over two parametric clocks and one parameter one can simulate several types of computations and combinations thereof, essentially forming a programming language. Then, the serializability characterization of EXPSPACE was rephrased in terms of an execution of a program. Finally this program has been shown to belong to the programming language.

In the following section, after some necessary precision on the definition of space bounded Turing machines, we are thus going to introduce our programming language whose interface was essentially introduced "by hand" in [48]. We show that it can serve as an interface for proving EXPSPACE lower bounds via serializability. Finally we show how, from a program and an input word for the program, one can build a $(2,1)$-PTA that simulates the behavior of the program on the input word.

5.2.1 Space bounded deterministic Turing machines

In the following, we introduce $f(n)$ space-bounded deterministic Turing machines. We consider a model of Turing machines that contains precisely one input tape and one working tape. In our setting, the working alphabet is $\{0, 1, \triangleright, \triangleleft\}$ where $\triangleright$ is the left marker and $\triangleleft$ is the right marker.

For an input alphabet Σ, the working tape of the initial configuration of such a deterministic Turing Machine on an input $w \in \Sigma$ of size n is assumed to be $\triangleright\, 0^{f(n)} \triangleleft$ whereas its input tape is $\triangleright\, w \triangleleft$.

Formally, an *$f(n)$-space bounded deterministic Turing Machine* (DTM for short) is a tuple $\mathsf{M} = (Q, \Sigma, q_0, F, R)$ where

- Q is a *finite set of states*,

- Σ is a *finite input alphabet* with $\triangleright, \triangleleft \notin \Sigma$,
- $R : Q \times (\Sigma \cup \{\triangleright, \triangleleft\}) \times \{0, 1, \triangleright, \triangleleft\} \to Q \times \{-1, 0, +1\}^2 \times \{0, 1, \triangleright, \triangleleft\}$ is a set of *rules*,
- $q_0 \in Q$ is an initial state, and
- $F \subseteq Q$ is the set of *final states*.

A *configuration* of a DTM M is a tuple consisting of a state q, a word $u_0 = \triangleright\, w_0 \,\triangleleft$ for some $w_0 \in \Sigma^n$, a word $u = \triangleright\, w \,\triangleleft$ for some $w \in \{0,1\}^{f(n)}$, a reading position $i \in \{0, 1, \ldots, n+1\}$ and a writing position $j \in \{0, 1, \ldots, f(n)+1\}$. Let $\mathsf{Conf}(M)$ denote the set of configurations of the DTM M.

The DTM M induces a transition system where there is a transition from a configuration (q, u_0, u, i, j) to a configuration (q', u_0, u', i', j') iff $R(q, u_0[i], u[j]) = (q', i' - i, j' - j, u'[j])$ and $u'[k] = u[k]$ for $k \neq j$.

We say a function $g : \Sigma^* \to \{0,1\}^*$ can be computed by an $f(n)$-space bounded DTM M if for all words $w \in \Sigma^*$, of size n, there exists a run in the transition system induced by M from $(q, \triangleright w \triangleleft, \triangleright 0^{f(n)} \triangleleft, 0, 0)$ to $(q', \triangleright w \triangleleft, \triangleright g(w) \triangleleft, i, j)$ for some $q \in Q$, $q' \in F$, and some $i, j \in \mathbb{N}$.

Recall that L, PSPACE, and $\mathsf{EXPSPACE}$ are used to denote the class of all problems that can be decided by a DTM that is logarithmically, polynomially, exponentially space bounded, respectively. We abuse notations and say a language L is in L, PSPACE, or $\mathsf{EXPSPACE}$ if the problem of whether or not a word w belongs to L is in the class $\mathsf{LOGSPACE}$, PSPACE, or $\mathsf{EXPSPACE}$ respectively.

5.2.2 The programming language

In this Section, we introduce a programming language tailored towards a possible reduction to obtain an $\mathsf{EXPSPACE}$ lower bound using the complexity class' serializability characterization.

Serializability is used to show $\mathsf{EXPSPACE}$-hardness in [47] and [48] for the data complexity of CTL on SOCA and the reachability problem for PTA respectively, each time building essentially "by hand" a programming langage to simulate serializability computations. The goal here is to make explicit the programming language used in both articles in order to serve as unifying framework for proving $\mathsf{EXPSPACE}$ lower bounds. The main requirement for the programming language is to be able to simulate serializability computations. We adopt the leaf language view of $\mathsf{EXPSPACE}$ from [47] which relies on the ability to simulate both operations of a deterministic automaton and L computations on a series of exponentially large bit strings.

The manipulation of an exponentially large bit string is the more complex element of the programming language. Programs will manipulate a number of bit strings of specified size — and whose size counts in the size of a program in which they appear — and a specific variable of unspecified size representing an arbitrarily large bit string. The programming language will essentially allow for resetting the specific variable to 0, incrementing it, performing L computations using it as an input, and checking its residual modulo some other variable. It will also allow PSPACE computations on the other variables. The semantic of a program will then be parameterized by a the size to be assigned to the special variable.

We now introduce the syntax and semantic of our programming language.

Syntax We assume a countably infinite set of variables $\mathcal{X}$ each of which has a specific associated size. Variables will be typically denoted by $x, y, z, \ldots$, with subscripts and superscripts. The size of a variable x will be denoted by $|x|$. Additionally we assume a special variable B without stipulated

size. We inductively define programs of our programming langage as follows:

- "$B := 0$" is a basic program,
- "$B := B + 1$" is a basic program,
- if x, y are variables of size n and m respectively, $g : \{0,1\}^n \to \{0,1\}^m$ is a PSPACE computable function, then "$y := g(x)$" is a basic program,
- if x, y are variables then "$y := B \bmod x$" is a basic program,
- if x, y are variables with $|y| = 1$ and U is a language in L, then "$y := \chi_U(x \cdot B)$" is a basic program, where '$\cdot$' stands for the concatenation of bit strings,
- if ϖ is a program and x is a variable with $|x| = 1$ then "while x do ϖ" is a program,
- if ϖ, ϖ' are programs and x is a variable with $|x| = 1$ then "if x do ϖ else do ϖ'" is a program, and
- if ϖ, ϖ' are programs then "ϖ; ϖ'" is a program.

The size of a program is defined inductively as

- $|B := 0| = 1$
- $|B := B + 1| = 1$
- $|y := g(x)| = |x| + |y| + |g|$
- $|y := B \bmod x| = |y| + |x|$
- $|y := \chi_U(x \cdot B)| = |y| + |x| + |U|$
- $|\text{while } x \text{ do } \varpi| = |x| + |\varpi|$
- $|\text{if } x \text{ do } \varpi \text{ else do } \varpi'| = |x| + |\varpi| + |\varpi'|$
- $|\varpi; \varpi'| = |\varpi| + |\varpi'|$

where the size of some PSPACE computable function $g : \{0,1\}^n \to \{0,1\}^m$ is the size of a PSPACE-bounded Turing Machine computing g, and the size $|U|$ of a language U in L is the size of a L-bounded Turing Machine deciding whether or not an input word belongs to U.

The sets of variables of a program is defined as follows:

- the set of variables of "$B := 0$" is $\{B\}$,
- the set of variables of "$B := B + 1$" is $\{B\}$,
- the set of variables of "$y := g(x)$" is $\{y, x\}$,
- the set of variables of "$y := B \bmod x$" is $\{y, x, B\}$,
- the set of variables of "$y := \chi_U(x \cdot B)$" is $\{y, x, B\}$,
- the set of variables of "while x do ϖ" is the union of $\{x\}$ with the set of variables of ϖ,
- the set of variables of "if x do ϖ else do ϖ'" is the union of $\{x\}$ with the sets of variables of ϖ and ϖ', and
- the set of variables of "ϖ; ϖ'" is the union of the sets of variables of ϖ and ϖ'.

Semantic For each $i, n \in \mathbb{N}$ let $\textsc{Bit}_i(n)$ denote the i-th least significant bit of the binary presentation of n, where the least significant bit is on the left, i.e. $n = \sum_{i\in\mathbb{N}} 2^i \cdot \textsc{Bit}_i(n)$. For each $m \geq 1$, by $\textsc{Bin}_m(n) = \textsc{Bit}_0(n)\cdots\textsc{Bit}_{m-1}(n)$ we denote the sequence of the first m least significant bits of the binary representation of n. Conversely, given a binary string $w = w_0\cdots w_{m-1} \in \{0,1\}^m$ of length m we denote by $\textsc{Val}(w) = \sum_{i=0}^{m-1} 2^i \cdot w_i \in [0, 2^m - 1]$ the value of w interpreted as a non-negative integer. By $\preceq_n$ we denote the lexicographic order on n-bit strings, thus $w \preceq_n v$ if $\textsc{Val}(w) \leq \textsc{Val}(v)$, e.g. $0101 \preceq_4 0011$.

For a finite set of variables $\mathcal{V} \subseteq \mathcal{X}$ and a size specifier n, we define a $(\mathcal{V}, n)$-variable assignment η as a function from the set $\mathcal{V} \cup \{B\}$ to $\{0,1\}^*$ such that for every variable $x \in \mathcal{V}$, $\eta(x) \in \{0,1\}^{|x|}$ and $\eta(B) \in \{0,1\}^{2^n}$. For a variable $x \in \mathcal{X}$, an element $w \in \{0,1\}^{|x|}$, and a $(\mathcal{V}, n)$ variable assignment η where $x \in \mathcal{V}$, $\eta[x \mapsto w]$ denotes the $(\mathcal{V}, n)$ variable assignment η' that maps x to w and, otherwise, coincides with η. For $w \in \{0,1\}^*$, $\eta[B \mapsto w]$ is defined similarly.

The semantic of a program ϖ is a partial function from any set of $(\mathcal{V}, n)$-assignments towards the same set of $(\mathcal{V}, n)$-assignments, denoted $[\![\varpi]\!]_n$, where $\mathcal{V}$ contains the set of variables of ϖ.

- $[\![B := 0]\!]_n(\eta) = \eta[B \mapsto 0^{2^n}]$
- $[\![B := B + 1]\!]_n(\eta) = \eta[B \mapsto \textsc{Bin}_{2^n}(\textsc{Val}(\eta(B)) + 1)]$
- $[\![y := g(x)]\!]_n(\eta) = \eta[y \mapsto g(\eta(x))]$
- $[\![y := B \bmod x]\!]_n(\eta) = \eta[y \mapsto \textsc{Bin}_{|y|}(\textsc{Val}(\eta(B)) \bmod \textsc{Val}(\eta(x)))]$
- $[\![y := \chi_U(x \cdot B)]\!]_n(\eta) = \eta[y \mapsto \chi_U(\eta(x) \cdot \eta(B))]$, where $\cdot$ represents string concatenation,
- $[\![\text{while } y \text{ do } \varpi]\!]_n(\eta) = \rho$ iff there exists a sequence of assignments $\lambda_1, \lambda_2, \ldots \lambda_t$ such that $\eta = \lambda_1$, $\lambda_t = \rho$, with $\rho(y) = 0$ (i.e. False) and for every $0 < j < t$, $\lambda_j(y) = 1$ (i.e. True) and $[\![\varpi]\!]_n(\lambda_j) = \lambda_{j+1}$,
- $[\![\text{if } y \text{ then } \varpi_1 \text{ else } \varpi_2]\!]_n(\eta) = \begin{cases} [\![\varpi_1]\!]_n(\eta) & \text{if } \eta(y) = 1, \\ [\![\varpi_2]\!]_n(\eta) & \text{otherwise,} \end{cases}$
- $[\![\varpi_1 \,;\, \varpi_2]\!]_n(\eta) = [\![\varpi_2]\!]_n([\![\varpi_1]\!]_n(\eta))$,

for all variables x, y and all PSPACE computable functions $g : \{0,1\}^{|x|} \to \{0,1\}^{|y|}$ and languages U in L. We say a program ϖ *returns* k for size specifier n and input η if $\textsc{Val}([\![\varpi]\!]_n(\eta)(B)) = k$.

For a program ϖ with variables $\mathcal{V}$ and a size specifier $n \in \mathbb{N}$, $\eta_{\varpi,n}$ is the $(\mathcal{V}, n)$-assignment such that $\eta_{\varpi,n}(x) = 0^{|x|}$ for all $x \in \mathcal{V}$ and $\eta_{\varpi,n}(B) = 0^{2^n}$.

5.2.3 Serializability

Our $\mathsf{EXPSPACE}$ lower bound proof makes use of the following leaf language view of $\mathsf{EXPSPACE}$ from [47], which is a padded adjustment of the leaf-language characterization of PSPACE from [59], which in turn has its roots in Barrington's Theorem [9].

Theorem 54 (Theorem 9 in [47])**.** *For every language $L \subseteq \{0,1\}^*$ in $\mathsf{EXPSPACE}$ there exists a polynomial $s : \mathbb{N} \to \mathbb{N}$, a regular language $R \subseteq \{0,1\}^*$, and a language $U \in \mathsf{L}$ such that for all $w \in \{0,1\}^n$ we have*

$$w \in L \iff \prod_{m=0}^{2^{2^{s(n)}-1}} \chi_U(w \cdot \textsc{Bin}_{2^{s(n)}}(m)) \in R, \tag{5.1}$$

where $\cdot$ and $\prod$ denote string concatenation.

Let us fix any language L in $\mathsf{EXPSPACE}$ and assume $L \subseteq \{0,1\}^*$ without loss of generality. Applying Theorem 54, let us fix the regular language $R \subseteq \{0,1\}^*$ along with some fixed deterministic

```
(1)   var q ∈ Q_D, x ∈ {0,1}^|w|
(2)   var b ∈ {0,1}, y ∈ {0,1}
(3)   var B ∈ ℕ
(4)   q := q_0;
(5)   x := w;
(6)   y := 1;
(7)   B := 0;
(8)   while y loop
(9)       b := χ_U(x · BIN_{2^s(n)}(B));
(10)      q := R_D(q, b);
(11)      B := B + 1;
(12)      y := B < 2^{2^n};
(13)  end loop
(14)  return q ∈ F_D
```

Figure 5.2: A program returning 1 if, and only if, $w \in L$ (using the characterization in Theorem 54), where $D = (Q_D, \{0,1\}, q_0, R_D, F_D)$ is some deterministic finite automaton such that $L(D) = R$.

finite automaton $D = (Q_D, \{0,1\}, q_0, R_D, F_D)$ with $L(D) = R$, the fixed polynomial s and the fixed language $U \in \mathsf{L}$. Let us moreover fix an input $w \in \{0,1\}^n$ of length n for L. Figure 5.2 rephrases characterization (5.1) in Theorem 54 in terms of an execution of a program that returns 1 if, and only if, $w \in L$.

This program can be realised by a program in the introduced programming language. Line (5), (6),(7), (9) and (11) are directly basic programs. Lines (4), (10) and (14) can be done by representing the states of the automaton D with variables. Line (14) especially can be done by differentiating whether $q \in F_D$ evaluates to 1 or 0 and setting B to the corresponding value. Line (8) follows the structure of the while loop from the programming language. Finally, line (12) will be done by checking that the binary representation of B is not $0^{2^{s(n)}}$ which can be done analogously as line (9).

Hence the following result:

Theorem 55. *Given a language L in* EXPSPACE*, a natural $n \in \mathbb{N}$, the following is computable in polynomial time in n:*
Input: a word $w \in \{0,1\}^n$
Output: a program ϖ of size polynomial in n such that ϖ returns 1 for size specifier $s(n)$ and input $\eta_{\varpi,s(n)}$ if, and only if, $w \in L$.

5.2.4 Simulation of the programming language

Let $\mathcal{A}$ be a parametric timed automaton over a set of clocks Ω with two parametric clocks x and y. We say a valuation $v : \Omega \to \mathbb{N}$ is *bit-compatible* if $v(\zeta) \in \{0,1\}$ for all non-parametric clocks $\zeta \in \Omega$ of $\mathcal{A}$. Assume moreover that Ω contains *non-parametric clocks* $\Theta_+ \cup \Theta_-$, where Θ is some set and $\Theta_+ = \{\vartheta^+ \mid \vartheta \in \Theta\}$ and $\Theta_- = \{\vartheta^- \mid \vartheta \in \Theta\}$ are two disjoint corresponding copies of Θ; in this case, for any valuation $v : \Omega \to \mathbb{N}$ we define the mapping $\widehat{v} : \Theta \to \{0,1\}$ as

$$\widehat{v}(\vartheta) = \begin{cases} 0 & \text{if } v(\vartheta^+) = v(\vartheta^-), \\ 1 & \text{otherwise.} \end{cases}$$

In the following we call such non-parametric clocks $\{\vartheta^+, \vartheta^- \mid \vartheta \in \Theta\}$, appearing as implicit pairs, *bit clocks* since they can be used to encode bits. The following definition expresses when a

parametric timed automaton over two parametric clocks and one parameter computes a function from $\mathbb{N} \times \{0,1\}^n$ to $\mathbb{N} \times \{0,1\}^m$. Notably, both before execution it assumes, and after execution it guarantees, a bit-compatible valuation that assigns its two parametric clocks values in the interval $[0, N-1]$, where N denotes the assigned value of its only parameter. In the following definition, it is important to note that the involved clocks are not (and in fact must not be) assumed to be initially set to zero.

Definition 56. *A $(2,1)$-PTA $\mathcal{A} = (Q, \Omega, \{p\}, R, q_{init}, \{q_{fin}\})$ whose parametric clocks are x and y and whose one parameter is p* computes a function $f : \mathbb{N} \times \{0,1\}^n \to \mathbb{N} \times \{0,1\}^m$ *if its set of clocks Ω contains two disjoint sets of*

- *non-parametric "input" bit clocks $\{in_0{}^+, in_0{}^-, \ldots, in_{n-1}^+, in_{n-1}^-\}$,*
- *non-parametric "output" bit clocks $\{out_0{}^+, out_0{}^-, \ldots, out_{m-1}{}^+, out_{m-1}{}^-\}$,*

such that there exists $N_0 \in \mathbb{N}$ such that for all $N > N_0$ and all bit-compatible $v_0 : \Omega \to [0, N-1]$ we have

1. *$q_{init}(v_0) \to_N^* q_{fin}(v')$ for some bit-compatible $v' : \Omega \to [0, N-1]$ and*
2. *for all $v' : \Omega \to \mathbb{N}$ for which $q_{init}(v_0) \to_N^* q_{fin}(v')$ we have*
   - *$v' \in [0, N-1]^\Omega$ is bit-compatible,*
   - *$\widehat{v'}(in_i) = \widehat{v_0}(in_i)$ for all $i \in [0, n-1]$,*
   - *$(v'(x) - v'(y) \bmod N, \prod_{j=0}^{m-1} \widehat{v'}(out_j)) = f(v_0(x) - v_0(y) \bmod N, \prod_{i=0}^{n-1} \widehat{v_0}(in_i))$, where $\prod$ denotes concatenation,*

   and for all $N \leq N_0$ and all bit-compatible $v_0, v' \in [0, N-1]^\Omega$ there is no N-run from $q_{init}(v_0)$ to $q_{fin}(v')$.

Importantly, the execution of any N-run $q_{init}(v_0) \to_N^* q_{fin}(v')$ does not have any side effect on the binary interpretation of the "input" bit clocks, i.e. the string $\prod_{i=0}^{n-1} \widehat{v_0}(in_i)$ equals $\prod_{i=0}^{n-1} \widehat{v'}(in_i)$.

For a given size specifier $n \in \mathbb{N}$, the semantic $[\![\varpi]\!]_n$ of a program ϖ with a set of variables $\mathcal{V}$ of size $|\mathcal{V}| = \Sigma_{x \in \mathcal{V}} |x|$ can be seen as a partial function from $\mathbb{N} \times \{0,1\}^{|\mathcal{V}|}$ to $\mathbb{N} \times \{0,1\}^{|\mathcal{V}|}$. More formally, the semantic $[\![\varpi]\!]_n$ of a program ϖ can be seen as the function $f_{\varpi,n} : \mathbb{N} \times \{0,1\}^{|\mathcal{V}|} \to \mathbb{N} \times \{0,1\}^{|\mathcal{V}|}$ with

$$f_{\varpi,n}(k, w) = (\textsc{Val}([\![\varpi]\!]_n(\eta_{k,w})(B)), \prod_{x \in \mathcal{V}} [\![\varpi]\!]_n(\eta_{k,w})(x))$$

where $\eta_{k,w}$ is the $(\mathcal{V}, n)$ assignment such that $\textsc{Val}(\eta_{k,w}(B)) = k$ and $\prod_{x \in \mathcal{V}} \eta_{k,w}(x) = w$. The goal of this section then is to show that, given a program ϖ and a size specifier $n \in \mathbb{N}$, one can compute in polynomial time in $n + |\varpi|$ a $(2,1)$-PTA computing $f_{\varpi,n}$.

The following lemma essentially has its roots in the PSPACE-hardness proof for the reachability problem for timed automata (without parameters) introduced by Alur and Dill [4], however constructed to satisfy the carefully chosen interface given by Definition 56.

Lemma 57. *For every* PSPACE*-computable function $g : \{0,1\}^n \to \{0,1\}^m$ one can compute in polynomial time in $n + m$ a $(2,1)$-PTA computing the function $f : \mathbb{N} \times \{0,1\}^n \to \mathbb{N} \times \{0,1\}^m$, where $f(k, w) = (k \bmod N_0, g(w))$ for all $(k, w) \in \mathbb{N} \times \{0,1\}^n$.*

Proof. Let us fix some PSPACE-computable function $g : \{0,1\}^n \to \{0,1\}^m$. Let us moreover fix some $t(n)$-space bounded deterministic Turing machine $\mathcal{M}$ computing g, where t is some fixed polynomial.

We explicitly store the value of our input by making use of our non-parametric "input" bit clocks $\{in_0{}^+, in_0{}^-, \ldots, in_{n-1}^+, in_{n-1}^-\}$. Similarly, we explicitly store the value of our output with the non-parametric "output" bit clocks $\{out_0{}^+, out_0{}^-, \ldots, out_{m-1}{}^+, out_{m-1}{}^-\}$. Since $f(k, w) = (k, g(w))$

we need to provide a computation that presents $g(w) \in \{0,1\}^m$ using the "output" bit clocks. Let Ω denote the set of clocks of the $(2,1)$-PTA $\mathcal{A}$ whose construction we discuss next.

For every non-parametric clock in $\mathcal{A}$ we reset it once it has value 2; this is achieved by suitable self-loops in every state of the construction except for the final state q_{fin}. Similarly, we establish that both of the parametric clocks x and y are being reset once they have reached value N, where N is the valuation of the only parameter p. This way the difference between the values of x and y will stay unchanged modulo N. Importantly, other than that neither x nor y will be modified during the following construction.

We will enforce that finally the values of all non-parametric clocks remain in $\{0,1\}$ and that the two parametric clocks have a value in $[0, N-1]$ as follows. A final state q_{fin} is preceded by a final gadget in which no time elapses that verifies via a sequence of suitable guards that the parametric and non-parametric clocks are as required. Hence for any starting bit-compatible valuation $v_0 : \Omega \to \mathbb{N}$, we ensure $v'(x) - v'(y) \equiv v_0(x) - v_0(y) \bmod N$ for all clock valuations v' such that $q_0(v_0) \to_N^* q_{fin}(v')$.

Let us consider now any pair of bit clocks ϑ^+ and ϑ^- and any current bit-compatible valuation $v : \Omega \to \mathbb{N}$. We have $\widehat{v}(\vartheta) = 1$ if, and only if, either $v(\vartheta^+) = 0$ and $v(\vartheta^-) = 1$ or conversely $v(\vartheta^+) = 1$ and $v(\vartheta^-) = 0$. Similarly, when we want to set the value $\widehat{v}(\vartheta)$ to 0, we reset both clocks ϑ^+ and ϑ^- at the same time, and when we want to set the value $\widehat{v}(\vartheta)$ to 1, we reset ϑ^- when $v(\vartheta^+) = 1$ without resetting ϑ^+.

For simulating $\mathcal{M}$ our $(2,1)$-PTA $\mathcal{A}$ will also use suitable $O(t(n))$ bit clocks, to store in binary the working tape of $\mathcal{M}$.

Given the current bit-compatible valuation $v : \Omega \to \mathbb{N}$, it is thus possible to inspect the input bit string $\prod_{i=0}^{n-1} \widehat{v}(in_i)$, read and write on the polynomially sized working tape, and to write the output $\prod_{j=0}^{m-1} \widehat{v}(out_j)$. Let us discuss this in more detail.

For simulating $\mathcal{M}$, we choose the states of our $(2,1)$-PTA $\mathcal{A}$ as

$$S \times \{0,\dots,n-1\} \times \{0,\dots,m-1\} \times \{0,\dots,t(n)-1\} \times \{0,1\} \times \{0,1\},$$

where S is the set of states of $\mathcal{M}$. We then simulate any step of $\mathcal{M}$ from a state q, current position i on the input tape, current position j on the output tape, current position h on the working tape, reading letter a on the input tape, reading letter b on the working tape, changing to a new state q', new input head position i', new output head position j', and new working head position h'. To do that, we add to $\mathcal{A}$ sequences of suitable rules from state (q,i,j,h,a,b) to state (q',i',j',h',a',b') for all $a', b' \in \{0,1\}$, by using suitable guards and reset operations that serve two purposes: first, checking whether a' and b' are indeed the values of the i'-th (resp. h'-th) cell of the input (resp. working) tape and second, writing on the j-th (resp. h-th) cell of the output (resp. working) tape.

Letting q_{init} denote some suitable initial state one can thus achieve that for all bit-compatible $v_0 : \Omega \to [0, N-1]$ and all $v' : \Omega \to \mathbb{N}$, if $q_0(v_0) \to_N^* q_{fin}(v')$ then v' is again a bit-compatible valuation from Ω to $[0, N-1]$. □

Remark 58. *The proof of Lemma 57 shows that if $g : \mathbb{N} \times \{0,1\}^n \to \mathbb{N} \times \{0,1\}^m$ is computable by a $(2,1)$-PTA, then so is the function $f : \mathbb{N} \times \{0,1\}^{n+\ell} \to \mathbb{N} \times \{0,1\}^m$, where $f(k,w) = g(k, w_1 \dots w_n)$ for all $k \in \mathbb{N}$ and all $w = w_1 \dots w_{n+\ell} \in \{0,1\}^{n+\ell}$: indeed, one can manipulate the 2ℓ additional input bit clocks by repeatedly resetting them once they have value 2, enforcing that the associated $\widehat{v}$-values stay throughout unchanged and that their value is finally strictly smaller than 2.*

Remark 59. *Of note is that Lemma 57 shows that $(2,1)$-PTA can compute the semantic of a basic program of the form "$y := g(x)$" where $g : \{0,1\}^{|x|} \to \{0,1\}^{|y|}$ is in* PSPACE.

The following lemma shows that $(2,1)$-PTA can compute modulo dynamically given numbers in binary.

Lemma 60. *One can compute in polynomial time in $n+m$ a $(2,1)$-PTA that computes the function $f : \mathbb{N} \times \{0,1\}^n \to \mathbb{N} \times \{0,1\}^m$, where $f(k,w) = (k, \textsc{Bin}_m(k \bmod \textsc{Val}(w)))$.*

Proof. We need to show that in time polynomial in $n+m$ one can construct a $(2,1)$-PTA $\mathcal{A}$ whose

set of clocks Ω contains the "input" bit clocks $\{in_0{}^+, in_0{}^-, \ldots, in_{n-1}^+, in_{n-1}^-\}$ and "output" bit clocks $\{out_0{}^+, out_0{}^-, \ldots, out_{m-1}{}^+, out_{m-1}{}^-\}$ that computes f. Let us assume some parameter value $N \in \mathbb{N}$ and some bit-compatible valuation $v_0 : \Omega \to [0, N-1]$ satisfying $w = \prod_{i=0}^{n-1} \widehat{v_0}(in_i)$.

Again, we establish here also that the parametric clocks x and y are being reset once they have reached value N — however we sometimes explicitly disallow x to reach value N in certain gadgets mentioned below. This will be the only modification of x and y. In the following, reading and writing the $\widehat{v}(\vartheta)$-value for every pair of bit clocks ϑ^+, ϑ^-, guaranteeing that $v(\vartheta^+), v(\vartheta^-) \in \{0,1\}$, and guaranteeing that the parametric clocks finally have values in $[0, N-1]$ can be done as in the proof of Lemma 57.

We need the eventual output bit string $\prod_{j=0}^{m-1} \widehat{v'}(out_j)$ to be equal to

$$\textsc{Bin}_m((v_0(x) - v_0(y) \bmod N) \bmod \textsc{Val}(w)).$$

Our automaton starts in some initial state q_{init}. From q_{init} we introduce a gadget that nondeterministically writes some value $u \in \{0,1\}^m$ in our "output" bit clocks that satisfies $\textsc{Val}(u) < \textsc{Val}(w)$. From the end of the latter gadget we have a rule that checks if our parametric clock x has value 0 (just after being reset with value N), leading to a state q_{wait}. Assume our current valuation then is $v : \Omega \to \mathbb{N}$. From q_{wait} we have a rule to a state q_{sub} letting no time elapse from which we claim there is a gadget that allows us to loop in q_{sub} for precisely $\textsc{Val}(w) = \textsc{Val}(\prod_{i=0}^{n-1} \widehat{v}(in_i)))$ time units. One constructs the latter gadget as follows. Subsequently for every $i \in [0, n-1]$ one reads $\widehat{v}(in_i)$ and in case $\widehat{v}(in_i) = 1$ lets precisely 2^i time units elapse via a suitable auxiliary clock and in case $\widehat{v}(in_i) = 0$ lets 0 time units elapse. The gadget ends with a sequence of rules leading back to q_{sub} by letting 0 time units elapse that verify that the parametric clock x has a value strictly smaller than N. Importantly, the parametric clock x is exceptionally *not reset* inside this gadget.

Finally, we add a rule from q_{sub} to a suitable gadget that lets precisely $\textsc{Val}(\prod_{j=0}^{m-1} \widehat{v}(out_j))$ time units elapse (analogously as done above), followed by a test that verifies that the value of y equals 0 (after just being reset at value N). In addition, we append this latter gadget with a final sequence of rules (again letting no time elapse) to our final state q_{fin} that test if both x and y have a value strictly smaller than N and test if all non-parametric clocks have a value strictly smaller than 2. Thus, every valuation $v' : \Omega \to \mathbb{N}$ for which $q_{init}(v_0) \to_N^* q_{fin}(v')$ holds is a bit-compatible valuation from Ω to $[0, N-1]$.

It is worth noting that by construction precisely $v_0(x) - v_0(y) \bmod N$ time units have passed in any computation q_{wait} to q_{fin}. Since we have repeatedly waited $\textsc{Val}(w)$ time units and finally verified that the remaining time is the guessed value initially nondeterministically written to our "output" bit clocks, we have

$$\begin{aligned} \prod_{j=0}^{m-1} \widehat{v'}(out_j) &= \textsc{Bin}_m((v_0(x) - v_0(y) \bmod N) \bmod \prod_{i=0}^{n-1} \widehat{v_0}(in_i))) \\ (v'(x) - v'(y) \bmod N, \prod_{j=0}^{m-1} \widehat{v'}(out_j)) &= f(v_0(x) - v_0(y) \bmod N, \prod_{i=0}^{n-1} \widehat{v_0}(in_i)) \end{aligned}$$

for any valuation $v' : \Omega \to \mathbb{N}$ with $q_{init}(v_0) \to_N^* q_{fin}(v')$, as required. □

Remark 61. *Of note is that Lemma 60 shows that $(2,1)$-PTA can compute the semantic of a basic program of the form "$y := B \bmod x$".*

We have just seen in Lemma 57 and Lemma 60 how variables with specified size can be stored using bit clocks. As seen in Lemma 60, we can use $v(x) - v(y) \bmod N$ to store some value in $[0, N-1]$. Since the values the difference $v(x) - v(y) \bmod N$ can take depends on the value N of the parameter, and since for a size specifier n we want to store values up to 2^{2^n}, we introduce now for any $n \in \mathbb{N}$ a gadget $(2,1)$-PTA that allows us to enforce that the parameter p can only be evaluated to numbers that are larger than 2^{2^n}.

Lemma 62. *One can compute in polynomial time in n some parametric timed automaton $\mathcal{A}_{big} = (Q_{big}, \Omega_{big}, \{p\}, R_{big}, q_{big,init}, \{q_{big,fin}\})$ with two parametric clocks $x, y \in \Omega_{big}$ and one parameter p such that*

1. *$q_{big,init}(\vec{0}) \xrightarrow{N}{}^{*} q_{big,fin}(v')$ for some $v' : \Omega_{big} \to \mathbb{N}$ for some $N \in \mathbb{N}$, and*
2. *for all $N \in \mathbb{N}$ and all $v' : \Omega_{big} \to \mathbb{N}$ we have $q_{big,init}(\vec{0}) \xrightarrow{N}{}^{*} q_{big,fin}(v')$ implies $N > 2^{2^n}$.*

Proof. Without loss of generality we may assume $2^{n+1} \geq 10$. Letting N denote the parameter value of its only parameter p, our $(2,1)$-PTA $\mathcal{A}_{big}$ will test whether $N-1$ is divisible by all numbers in the interval $[1, 2^{n+1}-1]$. This will be sufficient since $\mathsf{LCM}([1,k]) \geq 2^k$ for all $k \geq 9$ by [81], thus implying $N > N-1 \geq \mathsf{LCM}([1, 2^{n+1}-1]) \geq 2^{2^{n+1}-1} > 2^{2^n}$. Consider the following program which returns 1 if, and only if, all numbers in $[1, 2^{n+1}-1]$ divide $N-1$.

```
(1)   var I ∈ {0,1}^{n+1}
(2)   var J ∈ {0,1}^{n+1}
(3)   I := 0^{n+1}
(4)   while I ≠ 1^{n+1} loop
(5)       I := BIN_{n+1}(VAL(I) + 1)
(6)       J := BIN_{n+1}(N − 1 mod VAL(I))
(7)       if J ≠ 0^{n+1} then return 0
(8)   end loop
(9)   return 1
```

It remains to show that the program can be implemented by a $(2,1)$-PTA $\mathcal{A}_{big}$ with a suitable final state $q_{big,fin}$.

It is straightforward to initialize our two parametric clocks x and y in such a way that one can enforce valuations v that satisfy $v(x) - v(y) = N - 1 \bmod N$: indeed, starting from the valuation $\vec{0}$, we can wait one unit of time after which we reset x but not y.

We will use $O(n)$ suitable bit clocks for storing the variables I and J respectively.

Lines (3), (4) and (7) can easily directly be achieved by reading and writing the $O(n)$ many bits clocks reserved for storing I and J. Line (5) boils down to incrementing I when viewed as $n+1$ bit integer and is thus obviously a polynomial space computable function from $\{0,1\}^{n+1}$ to $\{0,1\}^{n+1}$ and hence computable using a suitable PTA based on Lemma 57. Line (6) is a function from $\mathbb{N} \times \{0,1\}^{n+1}$ to $\{0,1\}^{n+1}$ that can be implemented using a suitable PTA based on Lemma 60.

As in the proofs of Lemma 57 and Lemma 60 we reset the two parametric clocks x and y once they have reached value N but only in case we are outside any of the gadget PTA corresponding to line (5) and line (6), respectively. Similarly we realize the implementation of the bit clocks for I and J by resetting them once they have reached value 2. □

The following lemma shows that $(2,1)$-PTA can carry out L computations on an exponentially long string $w \cdot \text{BIN}_{2^n}(k)$.

Lemma 63. *For every language $U \in \mathsf{L}$ and fixed $n \in \mathbb{N}$, one can compute in polynomial time in n a $(2,1)$-PTA computing the function $f : \mathbb{N} \times \{0,1\}^n \to \mathbb{N} \times \{0,1\}$, where $f(k,w) = (k, \chi_U(w \cdot \text{BIN}_{2^n}(k)))$.*

Proof. Our PTA cannot easily "explicitly" store the value $\text{BIN}_{2^n}(k)$ in binary as in the proof of Lemma 57 via polynomially many (in n) bit clocks in such a way that, given the current valuation $v : \Omega \to \mathbb{N}$, it suffices to simply inspect their $\widehat{v}$-value: indeed, there are only singly-exponentially many different combinations of such $\widehat{v}$-values, yet $\text{BIN}_{2^n}(k)$ represents a number in $[0, 2^{2^n}-1]$ of doubly-exponential magnitude. We will rather store this number using the difference $v(x) - v(y) \bmod N$ between our only two parametric clocks x and y: since we can enforce that $N > 2^{2^n}$ by our gadget PTA $\mathcal{A}_{big}$, we can use the difference $v(x) - v(y) \bmod N$ to store $k \bmod N$. However, for computing $\chi_U(w \cdot \text{BIN}_{2^n}(k))$, we need to access certain bits of the exponentially long bit string $w \cdot \text{BIN}_{2^n}(k)$. For this, we access k in a different representation, namely in Chinese Remainder Representation that we introduce next.

Definition 64 (Chinese Remainder Representation)**.** *Let p_i denote the i-th prime number and assume $\prod_{i=1}^{m} p_i > k$ for some $m \in \mathbb{N}$. Then $\mathsf{CRR}_m(k)$ denotes the bit tuple $(b_{i,r})_{i\in[1,m],r\in[0,p_i-1]}$, where $b_{i,r} = 1$ if $k \bmod p_i = r$ and $b_{i,r} = 0$ otherwise.*

Since k will need to take values in $[0, 2^{2^n} - 1]$ and for every $j \in \mathbb{N}$ we have $\prod_{i=1}^{j} p_i > 2^j$ there exists some $m \in O(\log(2^{2^n})) = 2^{\mathrm{poly}(n)}$ such that $\prod_{i=1}^{m} p_i > k$. In other words, one can present k as

$$\mathsf{CRR}_m(k) = (b_{i,r})_{i\in[1,m],r\in[0,p_i-1]} \quad \text{for some } m \in 2^{\mathrm{poly}(n)} \quad . \tag{5.2}$$

Since by the Prime Number Theorem the i-th prime p_i is bounded by $O(i \log i)$ there exists some $\ell \in O(\log(m \log m)) = O(\log(2^{\mathrm{poly}(n)})) = \mathrm{poly}(n)$ such that ℓ bits are sufficient to store in *binary* precisely one of the primes p_i. Thus, similarly $O(\ell) = \mathrm{poly}(n)$ bits are sufficient to store in *binary* precisely one of the pairs of the form (i, r), where $i \in [1, m]$ and $r \in [0, p_i - 1]$. Moreover we have $|\mathsf{CRR}(k)| \in O(m^2 \log m) = 2^{\mathsf{poly}(n)} = 2^{\mathrm{poly}(n)}$.

Observe that we need to carry out L computations on our exponentially long string $w \cdot \textsc{Bin}_{2^n}(k)$. We only have an on-the-fly mechanism for accessing the Chinese Remainder Representation of k, notably still of exponential size in n. To have a chance to access concrete bits of k, we apply the following theorem that states that, given a number in Chinese Remainder Representation, one can compute in L its binary representation.

Theorem 65 (Theorem 3.3. in [24])**.** *The following problem is computable in* DLOGTIME*-uniform* NC^1 *(and thus in* L*):*
INPUT: $\mathsf{CRR}_m(k)$ *and* $j \in [1, m]$
OUTPUT: $\textsc{Bit}_j(k \bmod 2^m)$

Let us assume that we have $k < 2^{2^n}$ and recall that we have stored k as the difference $v(x) - v(y) \bmod N$ of our two parametric clocks x and y, assuming v to be our current clock valuation. Let us show how to compute $\chi_U(w \cdot \textsc{Bin}_{2^n}(k))$, where we recall that U is a language in L. Let us fix some logarithmically space bounded deterministic Turing machine $\mathcal{M}$ for U.

For simulating $\mathcal{M}$ our PTA $\mathcal{A}$ will use $O(\log(n + 2^n)) = \mathrm{poly}(n)$ auxiliary bit clocks $\mathcal{J}$ to store in binary the position of the input head of $\mathcal{M}$ and further $O(\log(n + 2^n)) = \mathrm{poly}(n)$ auxiliary bit clocks $\mathcal{W}$ in order to store the working tape of $\mathcal{M}$. Reading and writing on the working tape as well as updating the position of the input head can done analogously as in the proof of Lemma 57. It only remains to show how to access the cell content $\textsc{Bit}_j(w \cdot \textsc{Bin}_{2^n}(k))$ of the input head of $\mathcal{M}$, where we recall that j itself is stored inside the above-mentioned bit clocks $\mathcal{J}$.

To compute $\textsc{Bit}_j(w \cdot \textsc{Bin}_{2^n}(k))$ we apply Theorem 65 and simulate in turn a L machine $\mathcal{M}'$ whose input is assumed to be

$$\mathsf{CRR}(k) = (b_{i,r})_{i\in[1,m],r\in[0,p_i-1]} \text{ and } j \in [1, m],$$

where we already have direct access to j via the bit clocks $\mathcal{J}$ but need a special treatment in order to access the bit $b_{i,r}$ of $\mathsf{CRR}(k)$. Importantly, during the to-be discussed simulation of $\mathcal{M}'$ we never modify the $\widehat{v}$-values associated with the bit clocks in $\mathcal{J}$ and $\mathcal{W}$ that are being used in the (outermost) simulation of $\mathcal{M}$. Before discussing the access to $b_{i,r}$ let us first discuss the simulation of the working tape of $\mathcal{M}'$: this can be achieved by using $O(\log(|\mathsf{CRR}(k)| + n)) = O(\log(m^2 \cdot \log m + n)) = \mathrm{poly}(n)$ many auxiliary bit clocks $\mathcal{W}'$, say, where reading and writing the working tape is done again as in Lemma 57. It remains to discuss how to implement the input head in the simulation of $\mathcal{M}'$. As mentioned repeatedly above, input j can directly be accessed by the bit clocks $\mathcal{J}$. However, accessing $\mathsf{CRR}(k) = (b_{i,r})_{i\in[1,m],r\in[0,p_i-1]}$ cannot be done explicitly but on-the-fly: for this we reserve $O(\ell) = O(n) = \mathrm{poly}(n)$ additional auxiliary bit clocks $\mathcal{J}'$, say, to store in binary a pair of indices (i, r), where $i \in [1, m]$ and $r \in [0, p_i - 1]$. Given the binary access to (i, r) via the bit clocks $\mathcal{J}'$, one can compute via further suitable $\mathrm{poly}(\ell) = O(n) = \mathrm{poly}(n)$ bit clocks $\mathcal{H}$, say, the binary representation of the i-th prime number p_i in space polynomial in ℓ (and thus in n) by Lemma 57: indeed, given $i \in [1, m]$ in binary, i.e. using $\ell = \mathrm{poly}(n)$ bits, it is straightforward to compute the i-th prime in space polynomial in ℓ. Having a binary resentation of p_i via the bit clocks $\mathcal{H}$ one can finally compute $(v(x) - v(y) \bmod N) \bmod p_i$ via a gadget by Lemma 60. Our $(2, 1)$-PTA $\mathcal{A}$ can

thus indeed compute k mod p_i and thus decide if r equals the latter, which in turn is nothing but computing the to-be-computed input bit $b_{i,r}$ of $\mathsf{CRR}(k)$ for the simulation of $\mathcal{M}'$.

Concerning the implementation details of the simulation of $\mathcal{M}'$ it is important to remark (recalling Remark 58) that both during the sub-computation computing the i-th prime p_i (using Lemma 57) as well as during the sub-computation computing k mod p_i (using Lemma 60) one can guarantee that the $\widehat{v}$-values associated with the bit clocks in $\mathcal{J}, \mathcal{W}, \mathcal{J}'$ and $\mathcal{W}'$ are never being modified. □

Remark 66. *Of note is that Lemma 63 shows that $(2,1)$-PTA can simulate the semantics of the basic programs of the form "$y \coloneqq \chi_U(x \cdot B)$" where U is in L.*

It is straightforward that one can compute in polynomial time in $n+m$ a $(2,1)$-PTA that computes the function $f : \mathbb{N} \times \{0,1\}^n \to \mathbb{N} \times \{0,1\}^m$, where $f(k,w) = (0,w)$ or $f(k,w) = (k+1,w)$. The first boils down to resetting both parametric clocks x and y simultaneously, the second is realised by letting time elapse until the parametric clock y has value 1 (i.e. one time unit after it had value N and was reset), and then resetting it.

Thus for any basic program ϖ and size specifier n, one can compute in polynomial time in $n + |\varpi|$ a $(2,1)$-PTA that computes the semantic $[\![\varpi]\!]_n$ seen as a function $f_{\varpi,n}$. It remains to discuss combinations of programs. Given two functions $f : \mathbb{N} \times \{0,1\}^n \to \mathbb{N} \times \{0,1\}^m$ and $g : \mathbb{N} \times \{0,1\}^m \to \mathbb{N} \times \{0,1\}^\ell$ computable by $(2,1)$-PTA, it is straightforward to build $(2,1)$-PTA that computes the composition of f and g. Hence, recalling remark 58, given two programs ϖ, ϖ' and size specifier n one can compute in polynomial time in $n + |\varpi;\varpi'|$ a $(2,1)$-PTA that computes $f_{\varpi;\varpi',n}$. It is also straightorward to show that given two programs ϖ, ϖ', a variable x and size specifier n, one can compute in polynomial time in $n + |x| + |\varpi| + |\varpi'|$ a $(2,1)$-PTA that computes $f_{\text{while } x \text{ do } \varpi,n}$ or $f_{\text{if } x \text{ do } \varpi \text{ else } \varpi',n}$.

By induction, given a program ϖ and a natural n, one can build in polynomial time in $n + |\varpi|$ a $(2,1)$-PTA $\mathcal{A}_{\varpi,n}$ that simulate the semantic of ϖ.

Lemma 67. *For every program ϖ with variables $\mathcal{V}$, for every size specifier $n \in \mathbb{N}$, one can compute in polynomial time in $n + |\varpi|$ a $(2,1)$-PTA $\mathcal{A}_{\varpi,n}$ computing the function $f_{\varpi,n} : \mathbb{N} \times \{0,1\}^{|\mathcal{V}|} \to \mathbb{N} \times \{0,1\}^{|\mathcal{V}|}$ with $f_{\varpi,n}(k,w) = (\textsc{Val}([\![\varpi]\!]_n(\eta_{k,w})(B)), \prod_{x\in\mathcal{V}} [\![\varpi]\!]_n(\eta_{k,w})(x))$ where $\eta_{k,w}$ is the $(\mathcal{V},n)$ assignment such that $\textsc{Val}(\eta_{k,w}(B)) = k$ and $\prod_{x\in\mathcal{V}} \eta_{k,w}(x) = w$.*

We now reduce the problem of deciding whether a program ϖ returns 1 for a size specifier n and input $\eta_{\varpi,n}$ to $(2,1)$-PTA REACHABILITY. Given a program ϖ and size specifier $n \in \mathbb{N}$, we compute in polynomial time in $n + |\varpi|$ the automaton $\mathcal{A}_{\varpi,n}$. Note that since $\mathcal{A}_{\varpi,n}$ computes the function $f_{\varpi,n} : \mathbb{N} \times \{0,1\}^{|\mathcal{V}|} \to \mathbb{N} \times \{0,1\}^{|\mathcal{V}|}$, $\mathcal{A}_{\varpi,n}$ has parametric clocks x and y, and a set of clocks containing $2 \cdot |\mathcal{V}|$ "input" bit clocks, and $2 \cdot |\mathcal{V}|$ "output" bit clocks. The initial clock valuation $\vec{0}$ has all clocks mapped to 0 and thus all "input" bit evaluated to 0, much like $\eta_{\varpi,n}$ maps every variable x to $0^{|x|}$. Since $\mathcal{A}_{\varpi,n}$ computes the function $f_{\varpi,n}$, there exists $N_0 \in \mathbb{N}$ such that for all $N > N_0$, there exists some q_{fin}, and some bit-compatible $v' : \Omega \to \{0, N-1\}$ such that $q_{init}(\vec{0}) \to_N^* q_{fin}(v')$ and moreover $v'(x) - v'(y) \bmod N = \textsc{Val}([\![\varpi]\!]_n(\eta_{\varpi,n})(B))$. Since we say ϖ returns k for size specifier n and input η if $\textsc{Val}([\![\varpi]\!]_n(\eta)(B)) = k$, then ϖ returns 1 if and only if $v'(x) - v'(y) \bmod N = 1$. It is straightorward from q_{fin} to check that $v'(x) - v'(y) \bmod N = 1$ and to move to a final state if and only if that is the case. Hence the following result.

Theorem 68. *The following is computable in polynomial time:*

Input: a program ϖ, $n \in \mathsf{poly}(|\varpi|)$

Output: a $(2,1)$-PTA $\mathcal{A}$ such that there exists some $\mu \in \mathbb{N}^P$ and an accepting μ-run in $\mathcal{A}$ if, and only if, ϖ returns 1 for size specifier n and input $\eta_{\varpi,n}$.

Combined with Theorem 55, this provides the following reduction.

Theorem 69. *Given a language L in EXPSPACE, a natural $n \in \mathbb{N}$, the following is computable in polynomial time in n:*

Input: a word $w \in \{0,1\}^n$

Output: a $(2,1)$-PTA $\mathcal{A}$ such that there exists some $\mu \in \mathbb{N}^P$ and an accepting μ-run in $\mathcal{A}$ if, and only if, $w \in L$.

From this, we deduce the following lower bound.

Theorem 70. $(2,1)$-PTA REACHABILITY *is* EXPSPACE*-hard.*

5.3 An EXPSPACE upper bound via reduction to PTOCA-reachability

We recall that parametric one-counter automata are automata that can manipulate a counter that can be incremented or decremented, parametrically or not, compared against constants or parameters, and with divisibility tests modulo constants. We state in this Section a theorem (Theorem 71), proven essentially already in [17] — however, for a slightly more expressive model of parametric one-counter automata — that states that $(2,1)$-PTA REACHABILITY can be reduced in exponential time to the reachability problem of parametric one-counter automata over one parameter. We recall the Small Parameter Theorem (Theorem 21) which tells us that for every PTOCA over one parameter and every sufficiently large parameter value N, accepting N-runs with counter values all in $[0, h \cdot N]$ can be turned into accepting N' runs forr some smaller N'. We will show that the two theorem together implies an EXPSPACE upper bound for $(2,1)$-PTA REACHABILITY. We then provide a thorough proof of the reduction.

Bundala and Ouaknine [17] include for the purpose of their construction some operations of the form $+[0,p]$ that allow to nondeterministically add to the counter a value that lies in $[0,\mu(p)]$, where $\mu(p)$ is the parameter valuation of parameter p. As we shall show in this Section, when reducing the reachability problem for parametric timed automata with two parametric clocks and one parameter to parametric one-counter automata one does not require these $+[0,p]$-transitions nor binary increments.

The following theorem states an exponential time reduction from $(2,1)$-PTA REACHABILITY to the reachability problem of particular parametric one-counter automata over one parameter.

Theorem 71. *The following is computable in exponential time:*
INPUT: A $(2,1)$-PTA $\mathcal{A}$.
OUTPUT: A PTOCA $\mathcal{C}$ over one parameter
such that

1. *for all $N \in \mathbb{N}$ all accepting N-runs π in $\mathcal{C}$ satisfy* $\textsc{Values}(\pi) \subseteq [0, 4 \cdot \max(N, |\mathcal{C}|)]$*, and*
2. *there exists some $\mu \in \mathbb{N}^P$ and an accepting μ-run in $\mathcal{A}$ if, and only if, there exists some $\mu' \in \mathbb{N}^P$ and an accepting μ'-run in $\mathcal{C}$.*

We will show that even though the output of Theorem 71 is not exactly a 4-bounded PTOCA, we can still use the results from Chapter 4. Recall now the Small Parameter Theorem from Section 4.2.

Theorem (Small Parameter Theorem)**.** *Let $\mathcal{C} = (Q, \{p\}, R, q_{init}, F)$ be a PTOCA with one parameter p. If there exists an accepting N-run in $\mathcal{C}$ with values all in $[0, h \cdot N]$ for some $N > M_{\mathcal{C},h}$, then there exists an accepting $(N - \Gamma_{\mathcal{C},h})$-run in $\mathcal{C}$.*

Where $\mathsf{M}_{\mathcal{C},h}$ and $\Gamma_{\mathcal{C},h}$ are both constants asymptotically bounded by $2^{\mathsf{poly}(h,|\mathcal{C}|)}$ with more precise definitions on page 33.

Let us first establish that this theorem is enough to prove the desired EXPSPACE upper bound. The proof is nearly the same as that of the PSPACE upper bound on h-BOUNDED 1-PTOCA REACHABILITY seen on page 34.

Corollary 72. $(2,1)$-PTA REACHABILITY *is in* EXPSPACE*.*

Proof. Given a $(2,1)$-PTA $\mathcal{A}$, we apply Theorem 71 and translate $\mathcal{A}$ in exponential time into a PTOCA $\mathcal{C} = (Q, P, R, q_0, F)$ with $P = \{p\}$, such that

1. for all $N \in \mathbb{N}$ all accepting N-runs π in $\mathcal{C}$ satisfy $\textsc{Values}(\pi) \subseteq [0, 4 \cdot \max(N, |\mathcal{C}|)]$, and
2. there exists some $\mu \in \mathbb{N}^P$ and an accepting μ-run in $\mathcal{A}$ if, and only if, there exists some $\mu' \in \mathbb{N}^P$ and an accepting μ'-run in $\mathcal{C}$.

We first claim that if there exists an accepting N-run π in $\mathcal{C}$, then there exists one satisfying $N \in [0, \max\{M_{\mathcal{C},4}, |\mathcal{C}|\}]$ and $\textsc{Values}(\pi) \subseteq [0, 4 \cdot \max\{M_{\mathcal{C},4}, |\mathcal{C}|\}]$. All accepting N-runs π of $\mathcal{C}$ satisfy $\textsc{Values}(\pi) \subseteq [0, 4 \cdot \max\{N, |\mathcal{C}|\}]$ by Point 1, so if $N > \max\{M_{\mathcal{C},4}, |\mathcal{C}|\}$, then $4 \cdot N = 4 \cdot \max\{N, |\mathcal{C}|\}$ and hence there exists some accepting $(N - \Gamma_{\mathcal{C},4})$-run in $\mathcal{C}$ by the Small Parameter Theorem (Theorem 21). Remarking that in case $N > \max\{M_{\mathcal{C},4}, |\mathcal{C}|\}$ we have $N - \Gamma_{\mathcal{C},4} > M_{\mathcal{C},4} - \Gamma_{\mathcal{C},4} > 0$, one can repeat the above argument for $N - \Gamma_{\mathcal{C},4}$ and possibly for $N - 2\Gamma_{\mathcal{C},4}$ and so on, thus implying the desired existence.

Thus by Point 2 it suffices to check in exponential space in $|\mathcal{A}|$ whether there exists some accepting N-run π in $\mathcal{C}$ satisfying $\textsc{Values}(\pi) \subseteq [0, 4 \cdot \max\{N, |\mathcal{C}|\}]$ for some $N \in [0, \max\{M_{\mathcal{C},4}, |\mathcal{C}|\}]$. Since $M_{\mathcal{C},4} \in 2^{\mathrm{poly}(|\mathcal{C}|)} = 2^{2^{\mathrm{poly}(|\mathcal{A}|)}}$, the latter is simply a reachability question in a doubly-exponentially large finite graph all of whose vertices and edges can be represented using exponentially many bits, and thus decidable in exponential space.

□

The proof of the Small Parameter Theorem (Theorem 21) can be found in Section 4.2. The proof of Theorem 71 follows.

5.3.1 Overview of the proof of the reduction

In this section provide a proof overview for Theorem 71.

A more general (but strictly speaking incomparable) result involving two parametric clocks but an arbitrary number of parameters instead of only one has already been proven in [17], however with a different PTOCA formalism: Bundala and Ouaknine's model for PTOCA differs in that it contains operations that allow to nondeterministically add to the counter a value that lies in $[0, p]$. By restricting ourselves to the case of only one parameter p, we will prove in a thorough analysis that we no longer need such operations in the construction.

As in [17] we follow the following proof strategy:

- In Subsection 5.3.2, we reduce the reachability problem of a parametric timed automaton $\mathcal{A} = (Q_{\mathcal{A}}, \Omega_{\mathcal{A}}, P, R_{\mathcal{A}}, q_{\mathcal{A}}, F_{\mathcal{A}})$ — in our setting later with two parametric clocks — to the reachability problem of a so-called *parametric* 0/1 *timed automaton* $\mathcal{B} = (Q_{\mathcal{B}}, \Omega_{\mathcal{B}}, P, R_{\mathcal{B},0}, R_{\mathcal{B},1}, q_{\mathcal{B}}, F_{\mathcal{B}})$, where $\Omega_{\mathcal{B}} \subseteq \Omega_{\mathcal{A}}$ contains only the non-parametric clocks of $\Omega_{\mathcal{A}}$, and $\mathsf{Consts}(\mathcal{B}) = \{0\}$.
- In Subsection 5.3.3 we present the region abstraction technique introduced by Alur and Dill in [4] to mimic region-restricted runs (runs inside a region) of parametric 0/1 timed automata with one parameter by arithmetic progressions.
- Finally, we present the final step of the reduction in Subsection 5.3.4, where it is shown how to use the above-mentioned technique to mimic reset-free region-restricted runs in $\mathcal{B}$, and furthermore how to provide a construction in order to mimic resets in $\mathcal{B}$. The precise construction itself mainly deviates from [17] in the gadget construction for resets.

5.3.2 Removing non-parametric clocks and non-parametric guards

In this subsection we show how non-parametric guards and non-parametric clocks can be eliminated from parametric timed automata. Initially introduced in [5] we define the notion of parametric 0/1 timed automata: these are essentially parametric timed automata in which each rule dictates whether a unit of time passes or not. Alur, Henzinger and Vardi have already shown in [5] how the reachability problem for parametric timed automata can be reduced to the reachability problem for parametric 0/1 timed automata that do not contain any non-parametric clocks. We will provide in

Lemma 73 below an analogous reduction by not only eliminating all non-parametric clocks, but also all non-parametric guards (except for empty guards).

A *parametric* 0/1 *timed automaton* (0/1-*PTA* for short) is a tuple

$$\mathcal{B} = (Q, \Omega, P, R_0, R_1, q_{init}, F),$$

where $\mathcal{B}_i = (Q, \Omega, P, R_i, q_{init}, F)$ is a PTA for all $i \in \{0,1\}$. For simplicity we define its *size* as $|\mathcal{B}| = |\mathcal{B}_0| + |\mathcal{B}_1|$. Analogously, a clock $\omega \in \Omega$ is *parametric* if it is parametric in $\mathcal{B}_0$ or in $\mathcal{B}_1$. We analogously denote the constants of $\mathcal{B}$ by $\mathsf{Consts}(\mathcal{B})$ and its configurations by $\mathsf{Conf}(\mathcal{B})$.

A parametric 0/1 timed automaton $\mathcal{B} = (Q, \Omega, P, R_0, R_1, q_{init}, F)$ and a parameter valuation $\mu : P \to \mathbb{N}$ induce the labeled transition system $T^{\mu}_{\mathcal{B}} = (\mathsf{Conf}(\mathcal{B}), \lambda_{\mathcal{B}}, \to_{\mathcal{B},\mu})$ where $\lambda_{\mathcal{B}} = (R_0 \cup R_1) \times \{0,1\}$ and where $\to_{\mathcal{B},\mu}$ is defined such that for all $q(z), q'(z') \in \mathsf{Conf}(\mathcal{B})$, for all $(\delta, i) \in \lambda_{\mathcal{B}}$ with $\delta = (q, g, U, q') \in R_i$ $q(v) \xrightarrow{\delta,i}_{\mathcal{B},\mu} q'(v')$ if $v + i \vDash_{\mu} g$, $v'(u) = 0$ for all $u \in U$ and $v'(\omega) = v(\omega) + i$ for all $\omega \in \Omega \setminus U$.

As expected, we write $q(v) \xrightarrow{\delta,i}_{\mathcal{B},\mu} q'(v')$ if $q(v) \xrightarrow{\delta,i}_{\mathcal{B},\mu} q'(v')$ for some $i \in \{0,1\}$, and some $\delta \in R_i$. The notions of a (reset-free, accepting) μ-run (resp. N-run) for $\mathcal{B}$ are also defined as expected.

The convention used in this and the following subsections is that parametric 0/1 timed automata are denoted by $\mathcal{B}$. The main result of this subsection is the following lemma, stated slightly less general in [5] in that there is no requirement that $\mathsf{Consts}(\mathcal{B}) = \{0\}$.

Lemma 73 ([5])**.** *The following is computable in exponential time:*

INPUT: A PTA $\mathcal{A} = (Q_{\mathcal{A}}, \Omega_{\mathcal{A}}, P, R_{\mathcal{A}}, q_{\mathcal{A}}, F_{\mathcal{A}})$.
OUTPUT: A 0/1*-PTA* $\mathcal{B} = (Q_{\mathcal{B}}, \Omega_{\mathcal{B}}, P, R_{\mathcal{B},0}, R_{\mathcal{B},1}, q_{\mathcal{B}}, F_{\mathcal{B}})$*, where* $\Omega_{\mathcal{B}} \subseteq \Omega_{\mathcal{A}}$ *contains precisely the parametric clocks of* $\Omega_{\mathcal{A}}$*,* $\mathsf{Consts}(\mathcal{B}) = \{0\}$*, and such that there exists some* $\mu \in \mathbb{N}^P$ *and an accepting* μ*-run in* $\mathcal{A}$*, if, and only if, there exists some* $\mu' \in \mathbb{N}^P$ *and an accepting* μ'*-run in* $\mathcal{B}$*.*

We adjust the proof from [5]. While the idea of the construction remains the same, ours slightly deviates in that we explicitly have $\mathsf{Consts}(\mathcal{B}) = \{0\}$, i.e. we remove all non-parametric guards of the form $\omega \bowtie c$ with $c \neq 0$ as well as all non-parametric clocks.

Proof. Let us assume without loss of generality that $\mathcal{A}$ contains at least one parametric clock and let us fix one such clock x. We define the empty guard g_ϵ as $g_\epsilon = x \geq 0$ and observe that this guard is always satisfied. Let $c_{max} = \max(\mathsf{Consts}(\mathcal{A}))$ denote the largest constant appearing in $\mathcal{A}$. Note that once the value assigned to a clock ω by a valuation v is strictly above c_{max}, the precise value $v(\omega)$ is no longer of importance, merely the fact that $v(\omega)$ exceeds c_{max} is relevant. Since we work with discrete time configurations, the value assigned to ω is always a non-negative integer. We will eliminate all non-parametric clocks of $\Omega_{\mathcal{A}}$ by storing in the states of $\mathcal{B}$ the values of clocks up to $c_{max} + 1$, where $c_{max} + 1$ will stand for any value greater than $c_{\max}$. Moreover we eliminate all non-empty non-parametric guards by also storing in the states of $\mathcal{B}$ the values of parametric clocks in the same fashion. Formally, we define $\Omega_{\mathcal{B}} = \{\omega \in \Omega_{\mathcal{A}} \mid \omega \text{ is parametric}\}$, $Q_{\mathcal{B}} = Q_{\mathcal{A}} \times [0, c_{max}+1]^{\Omega_{\mathcal{A}}}$, P is the same in both automata, $F_{\mathcal{B}} = F_{\mathcal{A}} \times [0, c_{max} + 1]^{\Omega_{\mathcal{A}}}$, and $q_{\mathcal{B}} = (q_{\mathcal{A}}, v_0)$, where $v_0(\omega) = 0$ for all $\omega \in \Omega_{\mathcal{A}}$.

We ensure that the stored clocks progress simultaneously with the remaining parametric clocks by exploiting the fact that the rules dictate whether or not time elapses, and build the rules of $\mathcal{B}$ such that the +1 rules correspond to the progress of time in $\mathcal{A}$ whereas the +0 rules correspond to using a rule in $\mathcal{A}$. Formally,

- for every $q \in Q_{\mathcal{A}}$, $v \in [0, c_{max} + 1]^{\Omega_{\mathcal{A}}}$, we introduce a rule of the form $((q, v), g_\epsilon, \varnothing, (q, v'))$ in $R_{\mathcal{B},1}$, where $v'(\omega) = \min\{v(\omega) + 1, c_{max} + 1\}$ for all $\omega \in \Omega_{\mathcal{A}}$,
- for every $(q, g, U, q') \in R_{\mathcal{A}}$ with $g \in \mathcal{G}(\Omega_{\mathcal{B}}, P)$ a parametric guard, every $v \in [0, c_{max} + 1]^{\Omega_{\mathcal{A}}}$ we introduce a rule $((q, v), +0, g, U', (q', v')) \in R_{\mathcal{B},0}$, where v' is obtained from v except for assigning 0 to every clock in U and $U' = U \cap \Omega_{\mathcal{B}}$ is the subset of parametric clocks of U, and

- for every $(q, g, U, q') \in R_{\mathcal{A}}$ with $g \in \mathcal{G}(\Omega_{\mathcal{A}}, P)$ a non-parametric guard, every $v \in [0, c_{max}+1]^{\Omega_{\mathcal{A}}}$ such that $v \vDash g$, we introduce a rule $((q, v), +0, g_\epsilon, U', (q', v')) \in R_{\mathcal{B},0}$, where v' is obtained from v except for assigning 0 to every clock in U and $U' = U \cap \Omega_{\mathcal{B}}$ is the subset of parametric clocks of U.

□

For the remaining subsections, let us fix a PTA $\mathcal{A} = (Q_{\mathcal{A}}, \Omega_{\mathcal{A}}, P, R_{\mathcal{A}}, q_{\mathcal{A}}, F_{\mathcal{A}})$ with two parametric clocks x and y, and with $P = \{p\}$. Let us also fix the 0/1-PTA $\mathcal{B} = (Q_{\mathcal{B}}, \Omega_{\mathcal{B}}, P, R_{\mathcal{B},0}, R_{\mathcal{B},1}, q_{\mathcal{B}}, F_{\mathcal{B}})$ produced by Theorem 73 applied to PTA $\mathcal{A}$, and recall that $\mathcal{B}$ satisfies

- $P = \{p\}$,
- $\Omega_{\mathcal{B}} = \{x, y\}$, where x and y are parametric,
- $\mathsf{Consts}(\mathcal{B}) = \{0\}$, and
- there exists some $\mu \in \mathbb{N}^P$ and an accepting μ-run in $\mathcal{A}$ if, and only if, there exists some $\mu' \in \mathbb{N}^P$ and an accepting μ'-run in $\mathcal{B}$.

5.3.3 Capturing reset-free runs via the region abstraction technique

In this section we perform another preliminary construction before providing the proof of Theorem 71. We build parametric one-counter automata without tests and with updates only in $\{+0, +1\}$ that can mimic the behavior of parametric 0/1 timed automata with two parametric clocks and one parameter inside a reset-free run having only clocks valuations in a certain set. We first simply remove rules resetting at least one clock. We then show how to remove non-empty guards from parametric 0/1 timed automata taking inspiration from the region abstraction technique for timed automata first introduced in [4]. The technique appears already in the proofs of reduction from parametric timed automata with two clocks to parametric one-counter automata given in [52, 55] (for empty sets of parameters) and in [17]. We refer to [8] for further discussions on the region abstraction technique.

Recall that our fixed 0/1-PTA $\mathcal{B}$ satisfies $P = \{p\}$, $\Omega_{\mathcal{B}} = \{x, y\}$, where x and y are parametric, and $\mathsf{Consts}(\mathcal{B}) = \{0\}$.

Let us now explain the set of regions. For any valuation μ that assigns to our only parameter p the value N we prefer to write $\vDash_N$ instead of $\vDash_\mu$. Moreover, we prefer to view clock valuations $v : \{x, y\} \to \mathbb{N}$ as pairs $(v(x), v(y))$. Sets of clock valuations will correspondingly be denoted as subsets of $\mathbb{N} \times \mathbb{N}$. The regions are essentially, when assigning N to the one parameter p, maximal subsets of $\mathbb{N} \times \mathbb{N}$ equivalent with regards to the sets of guards of $\mathcal{B}$ their valuations satisfy. In other words, the regions we define are equivalence classes for the relation $\sim_N$, where $v \sim_N v'$ if for all possible guards g of $\mathcal{B}$ we have $v \vDash_N g$ if, and only if, $v' \vDash_N g$. Since the latter guards can only compare (using comparisons $<, \leq, =, \geq, >$) the clock valuations against values from the set $\{0, N\}$, it follows that $\sim_N$ has at most the following 16 equivalence classes, each of which we call *region* in the following.

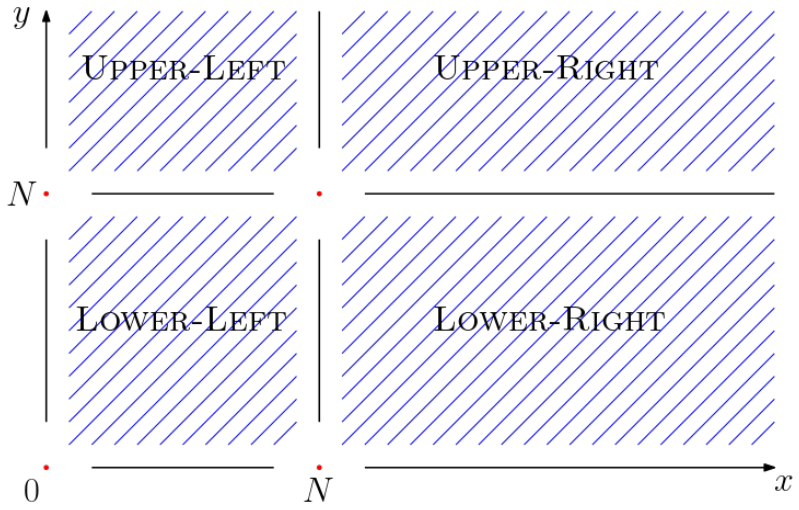

Figure 5.3: An illustration of the different regions.

$$(0,0),(0,N),(N,0),(N,N)$$
to respectively denote the singleton sets $\{(0,0)\},\{(0,N)\},\{(N,0)\},\{(N,N)\}$,

$$(0,0)\leftrightarrow(0,N),(N,0)\leftrightarrow(N,N),(0,N)\leftrightarrow(0,+\infty),(N,N)\leftrightarrow(N,+\infty),$$
to respectively denote the sets
$$\{(0,i)\mid 0<i<N\},\{(N,i)\mid 0<i<N\},\{(0,i)\mid i>N\},\{(N,i)\mid i>N\},$$

$$(0,0)\leftrightarrow(N,0),(0,N)\leftrightarrow(N,N),(N,0)\leftrightarrow(+\infty,0),(N,N)\leftrightarrow(+\infty,N),$$
to respectively denote the sets
$$\{(i,0)\mid 0<i<N\},\{(i,N)\mid 0<i<N\},\{(i,0)\mid i>N\},\{(i,N)\mid i>N\},$$

Lower-Left, Upper-Left, Lower-Right, Upper-Right
to respectively denote the sets
$$\{(i,j)\mid 0<i,j<N\},\{(i,j)\mid 0<i<N,j>N\},\{(i,j)\mid i>N,0<j<N\},\{(i,j)\mid i,j>N\}.$$

We refer to Figure 5.3 for an illustration of the different regions.

As expected, for every guard g of $\mathcal{B}$ and every region $\mathcal{R}$ we write $\mathcal{R}\vDash_N g$ if $v\vDash_N g$ for all $v\in\mathcal{R}$. For each region $\mathcal{R}$ we say a run $q_0(v_0)\xrightarrow{\delta_1,i_1}_{\mathcal{B},\mu} q_1(v_1)\cdots\xrightarrow{\delta_n,i_n}_{\mathcal{B},\mu} q_n(v_n)$ of $\mathcal{B}$ is *$\mathcal{R}$-restricted* if $v_j\in\mathcal{R}$ for all $j\in[0,n]$.

We remark that, in any $\mathcal{R}$-restricted run in $\mathcal{B}$, the set of guards being satisfied or not are the same for all configurations appearing in it. Thus, the set of guards that are satisfied only depend on the region and not the particular configurations of the $\mathcal{R}$-restricted run. We simply write $\mathcal{R}\vDash g$ when a region $\mathcal{R}$ satisfies guard g.

We use this property to remove guards from the parametric 0/1 timed automaton $\mathcal{B}$ while still mimicking reset-free $\mathcal{R}$-restricted runs.

For each region $\mathcal{R}$ we introduce the *region automaton* $\mathcal{B}_{\mathcal{R}}$ obtained from $\mathcal{B}$ instantiating all comparisons appropriately and by removing all rules that reset some clock. We fix g_ϵ to be the empty guard $x \geq 0$. Formally, the automaton $\mathcal{B}_{\mathcal{R}}$ is the 0/1-PTA obtained from $\mathcal{B}$ by

- removing all rules (q, g, U, q') with $U \neq \varnothing$,
- removing all rules $(q, g, \varnothing, q')$ for which $\mathcal{R} \not\models g$, and
- replacing all rules $(q, g, \varnothing, q')$ for which $\mathcal{R} \models g$ by $(q, g_\epsilon, \varnothing, q')$.

The following lemma is immediate.

Lemma 74. *From the 0/1-PTA $\mathcal{B} = (Q_{\mathcal{B}}, \{x, y\}, \{p\}, R_{\mathcal{B},0}, R_{\mathcal{B},1}, q_{\mathcal{B}}, F_{\mathcal{B}})$ with $\mathsf{Consts}(\mathcal{B}) = \{0\}$ one can compute in polynomial time (in $|\mathcal{B}|$) the sixteen 0/1-PTA $\{\mathcal{B}_{\mathcal{R}} | \ \mathcal{R}$ is a region$\}$ such that for all $N \in \mathbb{N}$, all regions $\mathcal{R}$, and all configurations $q(v)$ and $q'(v')$ for which $v, v' \in \mathcal{R}$ the following are equivalent:*

- *There exists an $\mathcal{R}$-restricted reset-free N-run from $q(v)$ to $q'(v')$ in $\mathcal{B}$.*
- *There exists an N-run from $q(v)$ to $q'(v')$ in $\mathcal{B}_{\mathcal{R}}$.*

5.3.4 Capturing reset-free runs via arithmetic progressions

Given a one-counter automaton $\mathcal{C}$ and two of its states q and q' we define the set $\Pi(\mathcal{C}, q, q')$ of counter values that configurations in state q' can have from runs starting in $q(0)$:

$$\Pi(\mathcal{C}, q, q') = \{v \in \mathbb{N} \mid q(0) \to^* q'(v)\}.$$

For all $a \geq 0$ and $b \geq 1$ we define the arithmetic progression $a + b\mathbb{N}$ as $a + b\mathbb{N} = \{a + b \cdot n \mid n \in \mathbb{N}\}$. The following theorem is an immediate consequence of a result by To analysing the succinctness between unary finite automata and arithmetic progressions [100].

Theorem 75 (Theorem 2 in [100])**.** *Let $\mathcal{C} = (Q, \varnothing, R, q_{init}, F)$ be a simple one-counter automaton with $+0, +1$ updates only. Then for every two states $q, q' \in Q$ one can compute in polynomial time a set $\{(a_j, b_j) \in \mathbb{N}^2 \mid j \in [1, r]\}$ such that $\Pi(\mathcal{C}, q, q') = \bigcup_{1 \leq j \leq r} a_j + b_j\mathbb{N}$, where moreover $r \in O(|Q|^2)$, $a_j \in O(|Q|^2)$, and $b_j \in O(|Q|)$ for all $j \in [1, r]$.*

We remark that Theorem 75 also holds in the presence of transitions that decrement the counter, cf. Lemma 6 in [50].

Remark 76. *Let $\mathcal{R}$ be a region and let $\mathcal{B}_{\mathcal{R}} = (Q_{\mathcal{B}_{\mathcal{R}}}, \{x, y\}, \{p\}, R_{\mathcal{B}_{\mathcal{R}},0}, R_{\mathcal{B}_{\mathcal{R}},1}, q_{\mathcal{B}_{\mathcal{R}},init}, F_{\mathcal{B}_{\mathcal{R}}})$ be the 0/1-PTA for $\mathcal{R}$. Then all rules in $R_{\mathcal{B}_{\mathcal{R}},0} \cup R_{\mathcal{B}_{\mathcal{R}},1}$ have as guard the empty guard g_ε. Let $\widehat{\mathcal{B}_{\mathcal{R}}} = (Q_{\mathcal{B}_{\mathcal{R}}}, \varnothing, R, q_{\mathcal{B}_{\mathcal{R}},init}, F_{\mathcal{B}_{\mathcal{R}}})$ be the one-counter automaton, where*

$$R = \{(q, +i, q') \mid (q, g_\varepsilon, \varnothing, q') \in R_{\mathcal{B}_{\mathcal{R}},i}, i \in \{0, 1\}\}$$

only contains $+0$ and $+1$ updates and does not contain any $= 0$-tests, Then for all $(k, \ell) \in \mathbb{N} \times \mathbb{N}$, all $q, q' \in Q$, and all $n \in \mathbb{N}$ the following are equivalent:

- *There is a run from $q(k, \ell)$ to $q'(k + n, \ell + n)$ in $\mathcal{B}_{\mathcal{R}}$.*
- *There is a run from $q(0)$ to $q'(n)$ in $\widehat{\mathcal{B}_{\mathcal{R}}}$.*

Notably, every run from $q(k, \ell)$ to $q'(k', \ell')$ in $\mathcal{B}_{\mathcal{R}}$ satisfies $k' = k + n$ and $\ell' = \ell + n$ for some $n \in \mathbb{N}$.

We apply Theorem 75 to all one-counter automata $\widehat{\mathcal{B}_R}$ from Remark 76. This yields the following characterization.

Lemma 77. *From the 0/1-PTA $\mathcal{B} = (Q_{\mathcal{B}}, \{x, y\}, \{p\}, R_{\mathcal{B},0}, R_{\mathcal{B},1}, q_{\mathcal{B}}, F_{\mathcal{B}})$ for every two states $q, q' \in Q_{\mathcal{B}}$, for all regions $\mathcal{R}$ one can compute in polynomial time (in $|\mathcal{B}|$) a set $\{(a_j, b_j) \in \mathbb{N}^2 | \ j \in [1, r]\}$ such that for all $N, t \in \mathbb{N}$ and all $v, v + t \in \mathcal{R}$ the following are equivalent:*

- *There exists a reset-free $\mathcal{R}$-restricted N-run from $q(v)$ to $q'(v + t)$ in $\mathcal{B}$.*
- *$t \in \bigcup_{1 \leq j \leq r} a_j + b_j\mathbb{N}$.*

Moreover, $r \in O(|Q_{\mathcal{B}}|^2)$, $a_j \in O(|Q_{\mathcal{B}}|^2)$, and $b_j \in O(|Q_{\mathcal{B}}|)$ for all $j \in [1, r]$.

Proof of Theorem 71: Construction of $\mathcal{C}$

Let us recall the fixed 0/1-PTA $\mathcal{B} = (Q_\mathcal{B}, \Omega_\mathcal{B}, \{p\}, R_\mathcal{B}, q_\mathcal{B}, F_\mathcal{B})$ the 0/1-PTA obtained from $\mathcal{A}$ by Theorem 73, and recall that $\mathcal{B}$ satisfies

- $P = \{p\}$,
- $\Omega_\mathcal{B} = \{x, y\}$ and x and y are parametric, and
- $\mathsf{Consts}(\mathcal{B}) = \{0\}$.

Recall also the set of regions of $\mathcal{B}$ defined in Section 5.3.3. We want to construct some PTOCA $\mathcal{C} = (Q_\mathcal{C}, \{p\}, R_\mathcal{C}, q_c, F_\mathcal{C})$ such that there exists some $\mu \in \mathbb{N}^P$ and an accepting μ-run in $\mathcal{B}$ if, and only if, there exists some $\mu' \in \mathbb{N}^P$ and an accepting μ'-run in $\mathcal{C}$ and moreover for all $N \in \mathbb{N}$, every accepting N-run π in $\mathcal{C}$ satisfies $\textsc{Values}(\pi) \subseteq [0, 4 \cdot \max(N, |\mathcal{C}|)]$.

The to-be-constructed PTOCA $\mathcal{C}$ (again over one parameter that will be evaluated to the same value as the only parameter of $\mathcal{B}$) will test whether an accepting N-run exists in $\mathcal{B}$ by using the definitions of regions and Lemma 77 from the last subsection, but also using additional gadgets to mimic the reset of a clock inside a particular region.

In what follows we denote the current value of the counter of $\mathcal{C}$ by z. For the time being in our construction z can be negative: we will later show how to obtain non-negativity and the required restriction that all N-runs π of $\mathcal{C}$ satisfy $\textsc{Values}(\pi) \subseteq [0, 4 \cdot \max\{N, |\mathcal{C}|\}]$.

The idea of the reduction is to factorize any possible accepting N-run into maximal reset-free subruns. We will use the current counter value z of $\mathcal{C}$ to store the clock valuation difference $v(x) - v(y)$, thus initially 0. We remark that between two consecutive resets, the difference $v(x) - v(y)$ stays the same throughout, but after some clock of $\Omega_\mathcal{B}$ (either x or y) is reset, this particular reset clock will be equal to zero but not necessarily the other one. The counter of $\mathcal{C}$ therefore needs to be modified accordingly. As expected, we construct $\mathcal{C}$ in such a way that after a reset of y, the counter value z equals $v(x)$, and after a reset of x the counter value z equals $-v(y)$. See Figure 5.4 for an idea of the relationship between $v(x), v(y)$ and z along the curve of the clock values.

Notice that once the value of a clock becomes strictly larger than N, its exact value is irrelevant to any future parametric comparison in $\mathcal{B}$, hence one only needs to remember that its value is strictly larger than N. Thus, our counter z will only track the values $v(x)$ and $v(y)$ up to N and possibly remember which of the two clock values exceeds N. Therefore, when a reset occurs and we store the value of the other clock in the counter, if it exceeds this N we can and will replace it by $N+1$, and if it is strictly below $-N$, we can and will replace it by $-N-1$. Let us therefore assume for now that the value of the counter z following the last reset is in this interval $[-N-1, N+1]$. Initially this is surely true as initially the value of the counter is 0. We will show how to provide this invariant on the next reset assuming it holds on the last reset.

Recall the definition of regions from Subsection 5.3.3. Let us assume a subrun $q(v) \to_N q'(v') \to_N^* q''(v'') \to_N q'''(v''')$ starting and ending by a reset of at least one of the two clocks $\{x, y\}$ and where $q'(v') \to_N^* q''(v'')$ is reset-free. We want $\mathcal{C}$ to be able to check whether such a run can exist.

For the rest of the proof let us assume without loss of generality that y is reset along $q(v) \to_N q'(v')$, where the latter configuration can hence be written as $q'(z, 0)$, as we want the counter z to store the value of x.

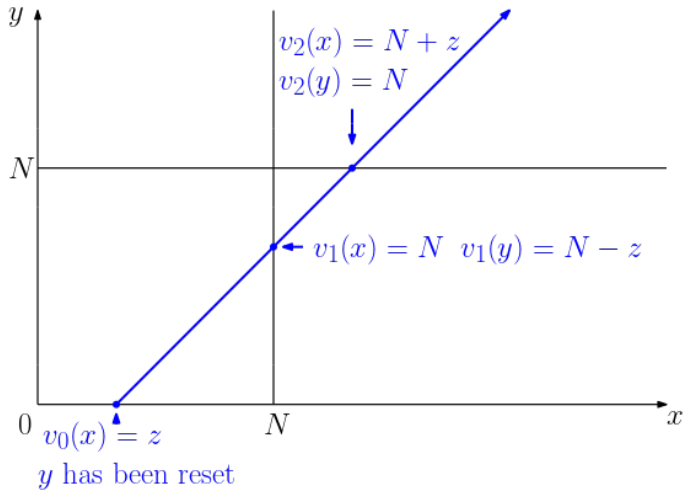

Figure 5.4: Curve of the clock values after a reset of clock y. Initially the difference z between the values of x and y is equal to the value of x.

The PTOCA $\mathcal{C}$ guesses

- the regions $\mathcal{R}_0$, $\mathcal{R}_1$, ..., $\mathcal{R}_l$ visited and the order in which they are visited, where here by convention $\mathcal{R}_k$ denotes the region assumed to be the k-th visited region,
- the states $s_0, \ldots, s_l$ when each region is visited for the first time,
- the states $q_0, \ldots, q_l$ when each region is visited for the last time,
- the state q'' in which the next reset of $\mathcal{B}$ occurs, and
- which clock is going to be reset next (either x or y).

Note that there are only a finite number of regions. Our PTOCA $\mathcal{C}$ then checks that the sequence $\mathcal{R}_0$, $\mathcal{R}_1$, ... $\mathcal{R}_l$ is valid, retaining the counter value z.

First $\mathcal{C}$ checks that $(z, 0)$ lies in $\mathcal{R}_0$ i.e. that z is equal to 0 if $\mathcal{R}_0 = (0, 0)$, strictly between 0 and N if $\mathcal{R}_0 = (0, 0) \leftrightarrow (N, 0)$, equal to N if $\mathcal{R}_0 = (N, 0)$ and strictly above N if $\mathcal{R}_0 = (N, 0) \leftrightarrow (+\infty, 0)$. and moreover checks that the guessed regions are adjacent, and that the regions can be visited in the guessed order.

Then $\mathcal{C}$ checks reachability within each individual region using Lemma 77 as follows. To each region $\mathcal{R}_k$ one can associate a set $\{(a_{k,j}, b_{k,j}) \in \mathbb{N}^2 \mid j \in [1, r_k]\}$ obtained by Lemma 77. This allows $\mathcal{C}$ to check, for every $k < l$, for every $v \in \mathcal{R}_k$, $v + t \in \mathcal{R}_k$, reachability of $q_k(v + t)$ from $s_k(v)$ in the region $\mathcal{R}_k$ by checking whether or not $t \in \bigcup_{1 \leq j \leq r} a_j + b_j\mathbb{N}$. In order to check reachability inside a region $\mathcal{R}_k$ of the form (α, β) or $(\alpha, \beta) \leftrightarrow (\gamma, \eta)$ for $\alpha, \beta \in \{0, N\}$, and $\gamma, \eta \in \{0, N, +\infty\}$, it suffices to check that $\bigcup_{1 \leq j \leq r} a_{k,j} + b_{k,j}\mathbb{N}$ contains 0, as the clock values cannot both increment and remain inside these regions, i.e. for any such $\mathcal{R}_k$, for all $v \in \mathcal{R}_k$, $v + t \in \mathcal{R}_k$ implies that $t = 0$. Indeed, one can check easily check whether $0 \in \bigcup_{1 \leq j \leq r} a_{k,j} + b_{k,j}\mathbb{N}$ by computing $\{(a_{k,j}, b_{k,j}) \in \mathbb{N}^2 \mid j \in [1, r_k]\}$, which can be done in polynomial time in $|\mathcal{B}|$.

Now, to check that an N-run exists in $\mathcal{B}$ in a given region $\mathcal{R}_k$ of the form Lower-Left, Lower-Right, Upper-Left or Upper-Right, the automaton $\mathcal{C}$ furthermore distinguishes whether the computation in the region $\mathcal{R}_k$ starts on the left side or on the bottom side, and whether the computation in the region $\mathcal{R}_k$ ends on the right side or on the top side, and uses the semilinearity property to check that the value added to the clocks is indeed in $\bigcup_{1 \leq j \leq r} a_{k,j} + b_{k,j}\mathbb{N}$. Note that the first configuration of Lower-Left is necessarily of the form $s_k(z + 1, 1)$ as y has been assumed to be the last clock to be reset, the first configuration of Lower-Right is of the form $s_k(z+1, 1)$ or $s_k(N+1, N+1-z)$, depending on whether it has been reached from the bottom

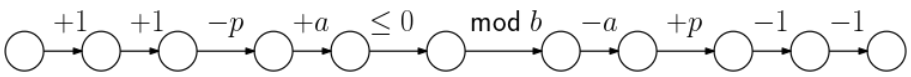

Figure 5.5: Gadget testing reachability for Case 1.

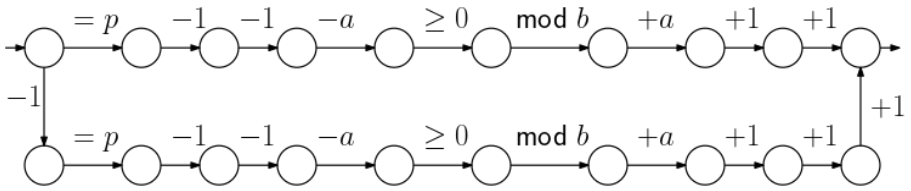

Figure 5.6: Gadget testing reachability for Case 2.

or from the left corner (or possibly both), and finally note that UPPER-LEFT cannot be reached if y was the last clock to be reset.

Thus, to check reachability inside $\mathcal{R}_k$, our PTOCA $\mathcal{C}$ guesses an offset $a = a_{k,j}$ and a period $b = b_{k,j}$ among the generators of $\{(a_{k,j}, b_{k,j}) \in \mathbb{N}^2 \mid j \in [1, r_k]\}$ that it will use to reach q_k. Secondly we define three gadgets in order to handle the three regions possibly traversed, namely LOWER-LEFT,LOWER-RIGHT, and UPPER-RIGHT.

Case 1. Checking reachability in the LOWER-LEFT *region.*

Here the region is necessarily reached from bottom side as y was the last clock to be reset. Moreover, as clocks progress at the same rate, the region is necessarily exited in the right corner (or both in the right and upper corner). Here $\mathcal{C}$ checks that $q_k(N-1, N-1-z)$ is reachable from $s_k(z+1, 1)$, i.e. $\mathcal{C}$ checks that $(N-1)-(z+1) \in a+b\mathbb{N}$ which in turn is equivalent to checking if $z+2-N+a = -n \cdot b$ for some $n \in \mathbb{N}$. See Figure 5.4 for an illustration of the trajectories of the counter values. In order to restore the value z the PTOCA $\mathcal{C}$ does this by a carefully chosen gadget shown in Figure 5.5. Since $(z+1, 1) \in$ LOWER-LEFT it follows $z \in [0, N-2]$, thus the counter value along the gadget stays inside the interval $[-(N-2), \max(N, a)]$.

Case 2. Checking reachability in the LOWER-RIGHT *region when reached from the bottom side.*

Here the region is necessarily exited in the top side, and we will show how $\mathcal{C}$ can check that $q_k(N+z-1, N-1)$ is reachable from $s_k(z+1, 1)$ and then restore z. Indeed, since y was the last clock that was reset, due to our convention that $z \in [-(N+1), N+1]$ and by our case we must have $z+1 \in \{N+1, N+2\}$, and therefore $z \in \{N, N+1\}$. Our PTOCA distinguishes the two cases $z = N$ and $z = N+1$ explicitly as follows. To check that that $q_k(N+z-1, N-1)$ is reachable from $s_k(z+1, 1)$ we need to test if $N-2 \in a+b\mathbb{N}$. Our PTOCA $\mathcal{C}$ first tests if z equals N or if z equals $N+1$, then does the test by a carefully chosen sequence of operations that allow to restore the counter value $z \in \{N, N+1\}$ as can be seen in the gadget in Figure 5.6. Since $(z+1, 1) \in$ LOWER-RIGHT the counter value along the gadget stays inside the interval $[-(a+2), N+1]$.

Case 3. Checking reachability in the LOWER-RIGHT *region when reached from the left side.*

Here the region is necessarily exited in the top side, and $\mathcal{C}$ checks that $q_k(N+z-1, N-1)$ is reachable from $s_k(N+1, N+1-z)$, i.e. $\mathcal{C}$ checks that $(z+N-1)-(N+1) \in a+b\mathbb{N}$ or equivalently if $z-2 \in a+b\mathbb{N}$. Since $(N+1, N+1-z) \in$ LOWER-RIGHT it follows $z \in [0, N]$. Again by a carefully chosen sequence of operations that allow to restore the counter value $z \in [0, N]$ we can realize this test as seen in the gadget in Figure 5.7. Since $(N+1, N+1-z) \in$ LOWER-RIGHT the counter value along the gadget stays inside the interval $[-(a+2), N]$.

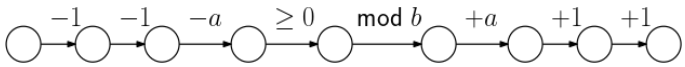

Figure 5.7: Gadget testing reachability for Case 3.

No other region is reachable from Upper-Right. Moreover, if y was among the last clocks to be reset, as the clocks valuations increment at the same rate, region Upper-Left is not reachable. Thus the three treated above cases conclude the question of reachability inside a region. Next, in order to test whether or not it is possible to reach $\mathcal{R}_{k+1}$ in state s_{k+1} from $\mathcal{R}_k$ and state q_k, we check whether or not in $\mathcal{B}$ there exists some +1 rule of the form $(q_k, g, \varnothing, s_{k+1})$ such that $\mathcal{R}_k \vDash g$ (and there is hence a corresponding rule in $\mathcal{B}_{\mathcal{R}_k}$).

To finish the construction our PTOCA $\mathcal{C}$ needs to be able to simulate clock resets in an N-run in $\mathcal{B}$. The process will depend on the guessed region $\mathcal{R}_l$ in which the reset is assumed to occur. For $\mathcal{R}_l$ of the form (α, β) with $\alpha, \beta \in \{0, N\}$, the precise value of each clock is known: if x is the next clock to be reset, then the new counter value should be $-v(y)$, i.e. $-\beta$, and if y is the next clock to be reset, then the new counter value should be $v(x)$, i.e. α. For $\mathcal{R}_l$ of the form $(\alpha, \beta) \leftrightarrow (\gamma, \beta)$, with $\alpha, \beta \in \{0, N\}$, and with $\gamma \in \{0, N, +\infty\}$, the precise value of each clock again is known: if x is the next clock to be reset, then the new counter value should be $-v(y)$, i.e. $-\beta$. If y is the next clock to be reset, then the new counter value should be $v(x)$, which, when z is the value of x when y was last reset, is equal to z plus the value of y, i.e. $z + \beta$. If z has has absolute value at most N, then $z + N$ has absolute value at most $2 \cdot N$. We thus test whether or not the absolute value of $z + \beta$'s exceeds $N + 1$ or not, and, if it is the case, we set it to $N + 1$ before performing any other operation.

The case when $\mathcal{R}_l$ is of the form $(\alpha, \beta) \leftrightarrow (\alpha, \delta)$ with $\alpha, \beta \in \{0, N\}$, and with $\delta \in \{0, N, +\infty\}$ is only possible if $\alpha = \delta = N$ (we refer to Figure 5.4 for an illustration of why the three other regions of the form $(\alpha, \beta) \leftrightarrow (\alpha, \delta)$ are not reachable) and is done as follows. The case when y is the next clock to be reset is again easy, we set the new counter value to N. If x is the next clock to be reset, then the new counter value should be $-v(y)$. To do so, observe that $v(y)$, when z was the value of x when y was last reset, is equal to $N - z$, thus the new counter value should be $-(N - z) = z - N$. Since $z \in [0, N + 1]$ by our case the new counter value has absolute value at most N.

Observe that since we have assumed without loss of generality that y was the last clock to be reset, we cannot have a reset inside the region Upper-Left. Thus, it remains to simulate resets in the regions Lower-Left, Lower-Right, and Upper-Right. For this observe that the precise value of each clock is not known, however it is feasible to nondeterministically guess the value of the clocks when the reset occurs, based on the region and whether it was reached from the bottom side or the left side. This case distinction allows us to know the exact starting clock valuation v_l of the $\mathcal{R}_l$-restricted run preceding the reset. From this, we guess an element t of $\bigcup_{1 \le j \le r_l} a_{l,j} + b_{l,j}\mathbb{N}$ to increment the clock valuation by t in such a way that $v_l + t \in \mathcal{R}_l$. We will distinguish which of the two clocks x and y will be reset next.

Case 1. Simulating resets in the Lower-Left *region.*

Let us first discuss the case when y (and only y) is the next clock to be reset. In this case $\mathcal{C}$ nondeterministically guesses a configuration $q(z + 1 + \delta, 1 + \delta)$ with $z + 1 + \delta \le N - 1$ reachable from $s_l(z + 1, 1)$, i.e. $\delta \in \bigcup_{1 \le j \le r_l} a_{l,j} + b_{l,j}\mathbb{N}$. To do that $\mathcal{C}$ adds a number of the form $1 + a + b \cdot n$ for some $n \in \mathbb{N}$ to the counter and checks that it is at most $N - 1$, as seen in Figure 5.8. We remark that counter values along this gadget stay inside $[0, N - 1]$.

Let us now discuss the case when x (and only x) is the next clock to be reset. In this case $\mathcal{C}$ nondeterministically establishes a counter value of the form $-\delta - 1$ such that $-(\delta + 1) \ge z - N + 1$, where $\delta = a + b \cdot n$ for some $n \in \mathbb{N}$, as seen in Figure 5.9 We remark that the counter values along this gadget stay inside $[-(N - 1), N - 1]$.

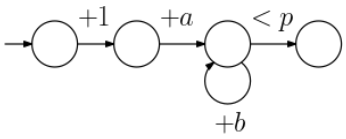

Figure 5.8: A gadget implementing a reset of clock y in the Case 1.

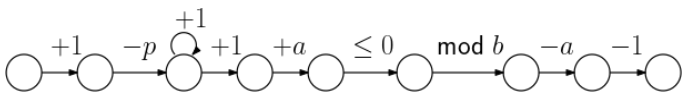

Figure 5.9: A gadget implementing a reset of clock x in the Case 1.

The case when x and y are next to be reset simultaneously can be done analogously by setting the new counter to 0 and is not discussed in detail here.

Case 2. Simulating resets in the LOWER-RIGHT *region when reached from the left side.*

Let us first discuss the case when y (and only y) is the next clock to be reset. In this case our PTOCA $\mathcal{C}$ nondeterministically guesses a configuration $q(N+1+\delta, N+1+\delta-z)$ with $N+1+\delta-z \leq N-1$ reachable from $s_l(N+1, N+1-z)$, i.e. where $\delta \in \bigcup_{1\leq j\leq r_l} a_{l,j} + b_{l,j}\mathbb{N}$ is of the form $a+b\cdot n$ with $a, b, n \in \mathbb{N}$. Then $\mathcal{C}$ will have counter value $N+1+\delta > N$, and thus $\mathcal{C}$ sets the counter value to $N+1$. To do that, $\mathcal{C}$ works as seen in Figure 5.10. We remark that the counter values along this gadget stay inside $[-1, 2N]$.

Let us now discuss the case when x (and only x) is the next clock to be reset. In this case our PTOCA $\mathcal{C}$ establishes the new counter value $z-\delta-N-1$, realized by the gadget seen in Figure 5.11. We remark that the counter values along this gadget stay inside $[-(N-1), N+1]$.

The case when x and y are next to be reset simultaneously can be done analogously by setting the new counter to 0 and is not discussed in detail here.

Case 3. Simulating resets in the LOWER-RIGHT *region when reached from the bottom side.*

Let us first discuss the case when y (and only y) is the next clock to be reset. In this case our PTOCA $\mathcal{C}$ nondeterministically guesses a configuration $s_l(z+1+\delta, 1+\delta)$ with $1+\delta \leq N-1$ reachable from $s_l(z+1, 1)$. We need to check that there exists $\delta \in \bigcup_{1\leq j\leq r_l} a_{l,j} + b_{l,j}\mathbb{N}$ which moreover satisfies the inequality $1+\delta \leq N-1$, or equivalently $z+1+\delta \leq N-1+z$. Moreover, as by assumption $z \leq N+1$, and moreover $(z+1, 1) \in$ LOWER-RIGHT, we must have $z \in \{N, N+1\}$. Our PTOCA distinguishes the two cases $z = N$ and $z = N+1$ explicitly similarly as checking reachability in the LOWER-RIGHT region when reached from the bottom side. The gadget can be found in Figure 5.12. We remark that the counter values along this gadget stay inside $[1, 2N]$.

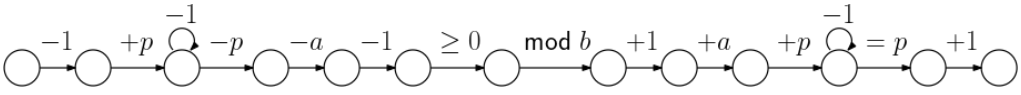

Figure 5.10: A gadget implementing a reset of clock y in the Case 2.

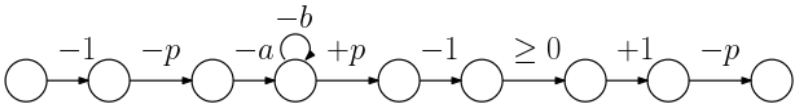

Figure 5.11: A gadget implementing a reset of clock x in the Case 2.

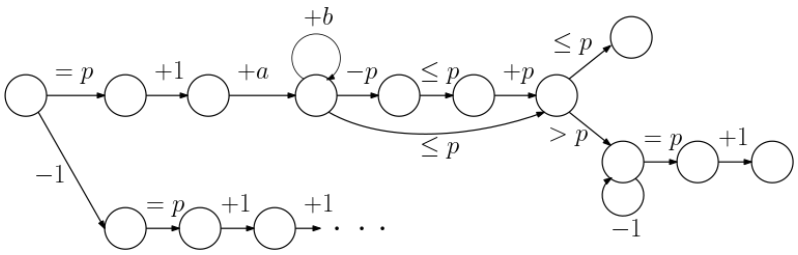

Figure 5.12: A gadget implementing a reset of clock y in the Case 3 with details for the case $z = N$. The $\cdots$ corresponds to the case $z = N + 1$ and works the same way.

Let us now discuss the case when x (and only x) is the next clock to be reset. The gadget can be found in Figure 5.13. We remark that the counter values along this gadget stay inside $[-(N-1), N+1]$.

The case when x and y are next to be reset simultaneously can be done analogously by setting the new counter to 0 and is not discussed in detail here.

Case 4. Simulating resets in the Upper-Right *region.*

Here by definition of the region the values of the clocks are above $N + 1$ and hence again their precise value is not relevant, only the existence of a way to reach the configuration when the reset occurs. Here we precompute in our reduction whether $\bigcup_{1 \le j \le r_l} a_{l,j} + b_{l,j}\mathbb{N}$ is not empty, and then set the counter to $N + 1$ (if y is to next to be reset) and to $-(N + 1)$ (if x is next to be reset) and to 0 if both are to be reset.

We notice that for each gadget implementation for testing reachability inside a region and for implementing the resets of clock x, clock y or both simultaneously, the value of the counter stays inside the interval $[-2 \cdot \max(a + 2, N), 2 \cdot \max(a + 2, N)]$, where a is the value of the offset used in the gadget.

Checking reachability and simulating resets when x was the last clock to be reset, instead of y, works again in a symmetrical way and can be dually shown to be such that the value of the counter stays inside the same interval. Testing reachability of a guessed final state inside a region works the same way as the implementation of a reset in the region, with $\mathcal{C}$ guessing a final state in which the computation ends instead of a state in which the next reset occurs.

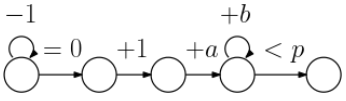

Figure 5.13: A gadget implementing a reset of clock x in the Case 3.

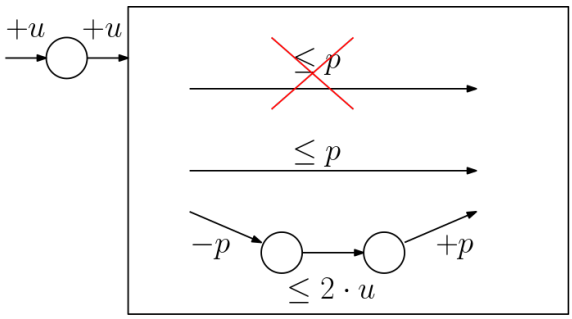

Figure 5.14: A gadget for adjusting a $\leq p$-test when initially offsetting the counter by $2 \cdot u$.

Finally we show how to achieve non-negativity. First, our final automaton checks whether or not the value N is greater than $(2 + c_{\max})$, where $c_{\max}$ is the maximal of all offsets $a_{k,j}$ and all periods $b_{k,j}$ used in any gadget. Then, fixing $u = \max(c_{\max} + 2, N)$, we transition into a new PTOCA obtained from the PTOCA described above (the construction where we allowed the counter to take negative values) by first adding two $+u$ gadgets before entering the initial state, as seen in Figure 5.14. Furthermore, any comparison operation $\leq p$ (resp. $\leq c$) is replaced by a gadget as seen in Figure 5.14, using an appropriate adjusted gadget for $\leq (2 \cdot u)$ comparison. Comparisons of the form $> p$, $= p$, $< p$, and $\leq p$ (resp. $> c$, $= c$, $< c$, and $\leq c$) are performed in an analogous manner.

Finally, for any modulo test, to simulate a $\mathsf{mod}\ b$ rule, we have two parallel branches,

- firstly a $\geq (2 \cdot u)$ comparison followed by determining the residual modulo b of the current counter value, say r_1, using the states (by repeatedly subtracting at most b from the counter, performing $\mathsf{mod}\ b$, then adding the same amount as subtracted), then subtracting u, then determining the new residual modulo b, say r_2, keeping track of it using the states too (by repeatedly subtracting at most b to the counter, then performing $\mathsf{mod}\ b$, and then adding the same amount as subtracted),
- secondly a $\leq (2 \cdot u)$ comparison, followed by a similar gadget but where instead of using a $-u$ operation, we use a $+u$ operation and instead of subtracting at most b, adding at most b.

We then compare the two residual r_1 and r_2 stored in the states, and check whether or not $r_1 - 2 \cdot (r_1 - r_2)$, the residual the counter value would have had without the $2 \cdot u$ offset, is equal to 0 (in the states), before restoring the counter value to the value it had before entering the gadget. Notice that this enforces that the value of the counters stays between 0 and $4 \cdot (\max(N, (2 + c_{\max}))$, and by observing that $|\mathcal{C}| \geq 2 + c_{\max}$, this enforces that the counter value stays between 0 and $4 \cdot (\max(N, |\mathcal{C}|))$.

5.4 Discussion and open problems

In this section we have shown that the reachability problem for parametric timed automata with two parametric clocks and one parameter is complete for exponential space.

For the lower bound proof, inspired by [47, 49], we have built a programming language which can be simulated by $(2, 1)$-PTA and which can compute EXPSPACE functions, making use of two results from complexity theory. First, we made use of a serializability characterization of EXPSPACE from [47] which is a padded version of the serializability characterization of PSPACE from [59], which in turn has its roots in Barrington's Theorem [9]. Second, we made use of a result of Chiu, Davida, Litow that states that numbers in Chinese Remainder Representation can be translated into binary representation in NC^1 (and thus in logarithmic space). We are convinced that our programming language can serve as a unifying framework in that it provides an interface for proving lower bounds for various problems involving automata.

For the EXPSPACE upper bound we first followed the approach of Bundala and Ouaknine [17]

by providing an exponential time translation from reachability in parametric timed automata with two parametric clocks and one parameter (i.e. $(2,1)$-PTA) to reachability in parametric threshold one-counter automata (PTOCA) over one parameter, yet on a slightly less expressive PTOCA model than the one introduced in [17]. We then studied the reachability in PTOCA with one parameter p. A repeated application of our Small Parameter Theorem (Theorem 21) allows to conclude that such a 1-PTOCA has an accepting N-run all of whose counter values lie in $[0, 4 \cdot N]$ if, and only if, there exists such an accepting N-run for some N that is at most exponential in the size of the 1-PTOCA. Since the translation from $(2,1)$-PTA to 1-PTOCA is computable in exponential time, this gives a doubly exponential upper bound on the parameter value of the original $(2,1)$-PTA and hence an EXPSPACE upper bound for $(2,1)$-PTA REACHABILITY (Corollary 72).

We hope that extensions of our techniques provide a line of attack for finally showing decidability (and the precise complexity) of $(2,*)$-PTA REACHABILITY. For reducing $(2,n)$-PTA REACHABILITY to n-PTOCA REACHABILITY however it seems that the PTOCA model indeed requires the presence of so-called $+[0,p]$-transitions. How our techniques can be extended to handle $+[0,p]$-transitions and an arbitrary number of parameters remains yet to be explored.

Chapter 6

Parametric pushdown automata

This chapter studies the computational complexity of reachability games and parity games in parametric pushdown automata. Parametric pushdown automata provide a formalism to reason about recursive programs making use of parametric constraints. Recently, different variants of parametric pushdown automata have been introduced in the literature [56, 39, 44] however as parameterized asynchronous shared-memory systems. These systems consist of a leader pushdown automaton and arbitrarily many identical contributor pushdown automata, communicating via a shared memory in the form of a register which can take finitely many values. This variant have been shown to have applications to the dataflow analysis of concurrent programs [66].

We consider here a different approach to extending pushdown automata with parameters, by allowing them to test equality of the stack content against parameters. Perhaps a similar model consists in Pushdown automata with transitions that are conditioned by regular conditions on their stack content, which can in particular test the stack content against specific words. They can be used to ensure that some word over the stack alphabet appears in the configurations of a run, but only for a specific value (or, rather, specific regular languages), and have been used to establish that CTL* model checking remains decidable when the formulas are allowed to include regular predicates on the stack content, and to obtain model checking algorithms for LTL and CTL* model checking for pushdown automata [43]. Pushdown automata with transitions that are conditioned by regular conditions on their stack content can be viewed as a special case of stack automata as seen in [62]. Instead of checking the stack content against regular conditions, or, in a more limited manner, against specified word values, we consider checks against unspecified word values that can be instantiated using parameters. This allows for instance to ensure equality between two stack contents appearing in two distincts configurations in a run.

Extensions of pushdown automata with storage for later comparisons have been introduced for instance as register pushdown automata. Such automata possess registers in addition to their stack, and can keep data values in both. Register pushdown automata have been shown to have applications for malware detection and XML schema checking [93, 94]. In theory, since registers can be unfilled and later be filled again an unbounded number of times, the number of different data values a pushdown register automata can store in its registers in a run is unbounded. In [80] it was shown however that a register pushdown automaton can only really "remember" at most $3r$ data values, where r is the number of registers, i.e. for any run of a register pushdown automaton with r registers there exists an equivalent run with the same initial and final configurations, but in which every configuration contains register assignments drawn from only $3r$ elements. This leads to the question of how useful the ability to unfill and later refill registers really is compared to a model that would only specify valuations once.

We provide formal definitions of parametric pushdown automata, parametric pushdown reachability games and parametric pushdown parity games in Section 6.1. We give an overview of our contribution in Section 6.1.1. The contribution consists in proving parametric pushdown reachability games and parametric pushdown parity games belong to $(n+1)$-EXP in case the number of parameters n is fixed, and providing a nonelementary lower bound for parametric

pushdown reachability games in general. We provide the proof of our result, which stretches along Section 6.3, Section 6.4, Section 6.5. In Section 6.6 we close the chapter with a discussion about the methods used, with some directions for future work.

6.1 Definitions

Parametric pushdown automata extend pushdown automata by allowing the stack to be compared against parameters that can be assigned values which are words over the stack alphabet. A parametric pushdown automaton is then a finite automaton extended with a finite set of parameters P and with a stack that can be manipulated by pushing or popping stack symbols and such that, moreover, the automaton can use the top of the stack, or check that the stack content corresponds to a parameter, to decide which transition to take next.

Formally, a *parametric pushdown automaton* (*PPDA* for short) is a tuple $\mathcal{Z} = (Q, \Gamma, P, R, q_{init}, \gamma_{init}, F)$, where

- Q is a non-empty finite *set of states*,
- Γ is a non-empty finite *stack alphabet*,
- P is a finite *set of parameters*, with $\Gamma \cap P = \varnothing$,
- $R \subseteq Q \times (\Gamma \uplus P) \times Q \times \mathsf{Op}(\Gamma)$ is finite *set of rules*,
- $q_{init} \in Q$ is an *initial state*,
- $\gamma_{init} \in \Gamma$ is an *initial stack symbol*, and
- $F \subseteq Q$ is a *set of final states*.

The *size* of $\mathcal{Z}$ is defined as $|\mathcal{Z}| = |Q| + |\Gamma| + |P| + |R|$. We also refer to $\mathcal{Z}$ as an n-parametric pushdown automaton if $|P| = n$. A *stack content* is a word from Γ^*. As before we write the top of the stack at the right of the word. By $\mathsf{Conf}(\mathcal{Z}) = Q \times \Gamma^*$ we denote the set of *configurations* of $\mathcal{Z}$. As usual we rather write $q(w)$ instead of (q, w). A *parameter valuation* is a function μ from P to Γ^*.

A parametric pushdown automaton $\mathcal{Z} = (Q, \Gamma, P, uR, q_{init}, \gamma_{init}, F)$ and a parameter valuation $\mu : P \to \Gamma^*$ induce the transition system $T^{\mu}_{\mathcal{Z}} = (\mathsf{Conf}(\mathcal{Z}), \to_{\mathcal{Z},\mu})$ where for all $q, q' \in Q$, for all $w, w' \in \Gamma^*$, and for all $a \in \Gamma$, $q(wa) \to_{\mathcal{Z},\mu} q'(w')$ if there exists a rule in R of the form (q, a, q', op) or (q, p, q', op) with $\mu(p) = wa$, such that either of the following holds

- $op = push^{\gamma}$ and $w' = wa\gamma$,
- $op = pop$ and $w' = w$, or
- $op = skip$ and $w' = wa$.

A *μ-run* from $q_0(a_0)$ to $q_n(a_n)$ in $\mathcal{Z}$ is a corresponding path in the transition system $T^{\mu}_{\mathcal{Z}}$ induced by $\mathcal{Z}$ and μ. As with PDA, we say π is *accepting* if $q_0 = q_{init}$, $a_0 = \gamma_{init}$, and $q_n \in F$.

In the particular case where $P = \{p\}$ is a singleton for some parameter p and $\mu(p) = u \in \Gamma^*$, we prefer to write $q(w) \to_{\mathcal{Z},u} q'(w')$ to denote $q(w) \to_{\mathcal{Z},\mu} q'(w')$ and will call the μ-run an *u-run*. In case the automaton $\mathcal{Z}$ is obvious from context, we write $\to_{\mu}$ (resp. $\to_u$) instead of $\to_{\mathcal{Z},\mu}$ (resp. $\to_{\mathcal{Z},u}$).

PPDA reachability Note that the reachability problem for parametric pushdown automata consists in the following decision problem.

<u>n-PPDA REACHABILITY</u>
INPUT: An n-PPDA $\mathcal{Z}$.
QUESTION: Does an accepting μ-run for some $\mu \in (\Gamma^*)^P$ exist in $\mathcal{Z}$?

PPDA games We are moreover interested in games played on parametric pushdown automata's transition systems. Recall that given a transition system, one needs only to provide a partition of the set of configuration S into two sets S_0 and S_1 to obtain an arena.

Thus, for an n-parametric pushdown automaton $\mathcal{Z} = (Q, \Gamma, P, R, q_{init}, \gamma_{init}, F)$, given a partition of Q into Q_0 and Q_1, we naturally partition the configurations of $\mathcal{Z}$ into $\mathsf{Conf}_{\mathcal{Z},0} = Q_0 \times \Gamma^*$ and $\mathsf{Conf}_{\mathcal{Z},1} = Q_1 \times \Gamma^*$.

With these notations in mind one can define the arena

$$A_{(\mathcal{Z},Q_0,Q_1,\mu)} = (\mathsf{Conf}_{\mathcal{Z},0}, \mathsf{Conf}_{\mathcal{Z},1}, \rightarrow_{\mathcal{Z},\mu})$$

induced by a PPDA $\mathcal{Z}$, a partition of its set of states, and a parameter valuation $\mu : P \to \Gamma^*$.

Given additionally a partition of P into P_0 and P_1, we define $A_{(\mathcal{Z},P_0,P_1,Q_0,Q_1)}$ as the arena that contains both the configurations and transitions of the parameter valuation arena A_{P_0,P_1,Γ^*}, as see on page 26, and that moreover contains the configurations and transitions of $A_{(\mathcal{Z},P_0,P_1,Q_0,Q_1,\mu)}$ — albeit with configurations additionally including μ in the tuple to avoid confusion — for all parameter valuations $\mu : P \to \Gamma^*$. Moreover, for every configuration μ in A_{P_0,P_1,Γ^*}, we add a transition from μ towards $q_{init}(\gamma_{init},\ \mu)$ in $A_{(\mathcal{Z},P_0,P_1,Q_0,Q_1,\mu)}$.

Given a priority function $\Omega : Q \to [0, m]$, we consider the function $\widehat{\Omega} : \mathsf{Conf}(\mathcal{Z}) \cup (\Gamma^* \cup \bot)^{[0,n]} \to [0, m]$ where we set $\widehat{\Omega}(q, w, \mu) = \Omega(q)$ for all $w \in \Gamma^*$ and for all $\mu \in (\Gamma^*)^P$, and $\widehat{\Omega}(\mu) = 0$ for all $\mu \in (\Gamma^* \cup \bot)^P$.

We are interested in the following games and problems.

Parametric pushdown reachability game

INPUT: A parametric pushdown automaton $\mathcal{Z} = (Q, \Gamma, \{p_0, p_1, \ldots, p_k\}, R, q_{init}, \gamma_{init}, F)$, where $Q = Q_0 \uplus Q_1$, and $P = P_0 \uplus P_1$.

QUESTION: Does player 0 have a winning strategy from $\mu_\bot$ in the reachability game $\mathcal{G} = (A_{(\mathcal{Z},P_0,P_1,Q_0,Q_1)}, \textsc{Win}_{F \times \Gamma^* \times (\Gamma^*)^P})$?

Parametric pushdown parity game

INPUT: A parametric pushdown automaton $\mathcal{Z} = (Q, \Gamma, \{p_0, p_1, \ldots, p_k\}, R, q_{init}, \gamma_{init}, F)$, where $Q = Q_0 \uplus Q_1$, and $P = P_0 \uplus P_1$, and a priority function $\Omega : Q \to [0, m]$.

QUESTION: Does player 0 have a winning strategy from $\mu_\bot$ in the parity game $\mathcal{G} = (A_{(\mathcal{Z},P_0,P_1,Q_0,Q_1)}, \textsc{Win}_{\widehat{\Omega}})$?

We write n-parametric reachability parity game and n-parametric pushdown parity game if the number of parameters $n = |P|$ of $\mathcal{Z}$ is fixed by the problem.

6.1.1 Contribution

The following theorems state our main results concerning Parametric pushdown parity game and Parametric pushdown reachability game. The contribution is two-fold.

Theorem 78. *n-parametric pushdown parity game and n-parametric pushdown reachability game are in $(n+1)$-EXP.*

Theorem 79. *Parametric pushdown parity game and Parametric pushdown reachability game are not in* ELEMENTARY.

For the nonelementary lower bound, we reduce the FO satisfiability problem on words, known to be nonelementary from [99], to the problem of deciding whether a player has a winning strategy for parametric pushdown reachability games.

For the upper bound, we start by replacing parameters by pebbles acting as registers, leading to a more general model, pebble pushdown automata. We then show that higher-order pushdown automata parity games can be used to solve parity games on the transition systems of pebble pushdown automata, using one additional stack level for each pebble. Since solving parity games

on higher-order pushdown automata with level n stack is n-EXP-complete [19, 20], this provides an $(n+1)$-EXP upper bound for solving parametric parity games on parametric pushdown automata with n parameters.

6.1.2 Overview

In Section 6.2, we provide preliminary definitions. Section 6.3 shows a nonelementary lower bound for the problem. Section 6.4 will deal with the introduction of pebble pushdown games. Section 6.5 is finally devoted to using higher-order pushdown automata parity games to solve pebble pushdown automata parity games.

6.2 Logics

We now briefly review some standard definitions from mathematical logic.

Definition 80. *A* vocabulary τ *is a set of relational symbols (denoted* $E_1, \ldots, E_n, \ldots$*), each of which has a specified* arity *in* $\mathbb{N}$*. A symbol* $E \in \tau$ *is called monadic if its arity is one, i.e., if it is used to denote sets. A* τ-structure *(also called a* model*)*

$$\mathfrak{A} = (A, (E^{\mathfrak{A}})_{E \in \tau})$$

consists of a set A *together with an interpretation of each* k*-ary relation symbol* E *from* τ *as a* k*-ary relation on* A*; that is, a set* $E^{\mathfrak{A}} \subseteq A^k$*. A structure* $\mathfrak{A}$ *is called finite if* A *and* τ *are finite sets. The* universe *of a structure is typically denoted by a Roman letter corresponding to the name of the structure; that is, the universe of* $\mathfrak{A}$ *is the set* A*, the universe of* $\mathfrak{B}$ *is* B*, and so on. We shall also occasionally write* $a \in \mathfrak{A}$ *instead of* $a \in A$*.*

For example, if τ consists of a single relational symbol $\to$ of arity 2, then possible structures for τ consist of transition systems.

Next, we define first-order formulae over a vocabulary τ, or FO$[\tau]$ for short (or simply FO when the vocabulary is obvious from context). We define free variables, and the semantics of FO$[\tau]$ formulae.

Definition 81. *We assume a countably infinite set of first-order variables* Var*. First-order variables will be typically denoted by* $x, y, z, \ldots$*, with subscripts and superscripts. We inductively define formulae of the first-order logic over vocabulary* τ *as follows:*

- *If* x_1, x_2 *are first-order variables, then* $x_1 = x_2$ *is an (atomic) formula.*
- *If* $x_1, \ldots, x_k$ *are first-order variables and* $E \in \tau$ *is a* k*-ary relation symbol, then* $E(x_1, \ldots, x_k)$ *is an (atomic) formula.*
- *If* ϕ_1, ϕ_2 *are formulae, then* $\phi_1 \wedge \phi_2$ *,* $\phi_1 \vee \phi_2$*, and* $\neg\phi_1$ *are formulae.*
- *If* ϕ *is a formula, and* x *a first-order variable, then* $\exists x\ \phi$ *and* $\forall x\ \phi$ *are formulae.*

A formula that does not use existential ($\exists$) and universal ($\forall$) quantifiers is called *quantifier-free*. Given a set of formulae S, formulae constructed from formulae in S using only the Boolean connectives $\vee$, $\wedge$, and $\neg$ are called *Boolean combinations* of formulae in S. We shall use the standard shorthand $\phi \to \psi$ for $\neg\phi \vee \psi$ and $\phi \leftrightarrow \psi$ for $(\phi \to \psi) \wedge (\psi \to \phi)$.

The sets of free variables of a formula are defined as follows:

- $\mathit{free}(x_1 = x_2) = \{x_1, x_2\}$,
- $\mathit{free}(E(x_1, \ldots, x_k)) = \{x_1, \ldots, x_k\}$,
- $\mathit{free}(\neg\phi) = \mathit{free}(\phi)$,
- $\mathit{free}(\phi \vee \psi) = \mathit{free}(\phi) \cup \mathit{free}(\psi)$,
- $\mathit{free}(\phi \wedge \psi) = \mathit{free}(\phi) \cup \mathit{free}(\psi)$,
- $\mathit{free}(\exists x\ \phi) = \mathit{free}(\phi) \setminus \{x\}$,
- $\mathit{free}(\forall x\ \phi) = \mathit{free}(\phi) \setminus \{x\}$.

If $\overrightarrow{x}$ is the tuple of all the free first-order variables of ϕ, we write $\phi(\overrightarrow{x})$. A *sentence* is a formula without free variables. For a finite set $\mathcal{V} \subseteq \mathrm{Var}$, and a structure $\mathfrak{A}$, we define a $(\mathcal{V}, \mathfrak{A})$ *variable assignment* μ as a partial function from Var to A whose domain is $\mathcal{V}$. For a first-order variable $x \in \mathrm{Var}$, an element $a \in A$, and a $(\mathcal{V}, \mathfrak{A})$ variable assignment μ where $x \notin \mathcal{V}$, $\mu[x \mapsto a]$ denotes the $((\mathcal{V} \cup \{x\}), \mathfrak{A})$ variable assignment μ' that maps x to a and, otherwise, coincides with μ.

Given a τ-structure $\mathfrak{A}$, we define inductively, for each formula ϕ with $\overrightarrow{x}$ as free variables, the notion that $\mathfrak{A}$ satisfies ϕ for the $(\mathcal{V}, \mathfrak{A})$ variable assignment μ, where $\overrightarrow{x} \subseteq \mathcal{V}$, which we denote by $\mathfrak{A} \models_\mu \phi$. If ϕ is a sentence, we just write $\mathfrak{A} \models \phi$.

1. $\mathfrak{A} \models_\mu (x = y)$ if $\mu(x) = \mu(y)$.
2. $\mathfrak{A} \models_\mu E(x_1, \ldots, x_k)$ if $(\mu(x_1), \ldots, \mu(a_k)) \in E^{\mathfrak{A}}$.
3. $\mathfrak{A} \models_\mu \neg\phi$ if $\mathfrak{A} \models_\mu \phi$ does not hold.
4. $\mathfrak{A} \models_\mu \phi_1 \wedge \phi_2$ if $\mathfrak{A} \models_\mu \phi_1$ and $\mathfrak{A} \models_\mu \phi_2$.
5. $\mathfrak{A} \models_\mu \phi_1 \vee \phi_2$ if $\mathfrak{A} \models_\mu \phi_1$ or $\mathfrak{A} \models_\mu \phi_2$.
6. $\mathfrak{A} \models_\mu \exists y\ \phi(y, \overrightarrow{x})$ iff there exists $a \in A$ such that $\mathfrak{A} \models_{\mu[y \mapsto a]} \phi$.
7. $\mathfrak{A} \models_\mu \forall y\ \phi(y, \overrightarrow{x})$ iff for all $a \in A$, $\mathfrak{A} \models_{\mu[y \mapsto a]} \phi$.

Concerning the logic FO, we are interested in the following decision problem.

$\mathsf{FO}[\tau]$ MODEL CHECKING

INPUT: A τ-structure $\mathfrak{A}$ and a $\mathsf{FO}[\tau]$-sentence ϕ.
QUESTION: Does $\mathfrak{A} \models \phi$?

6.3 A nonelementary lower bound

The aim of this section is to show a nonelementary lower bound for the problem of solving parametric pushdown reachability games. It is trivial to show that the problem of solving parametric pushdown reachability game reduces itself to the problem of solving parametric pushdown parity game.

We reduce the satisfiability problem for first-order logic on words to solving PARAMETRIC PUSHDOWN REACHABILITY GAME. We know from [99] that satisfiability is nonelementary.

First we introduce in more detail what we call the satisfiability problem for first-order logic on the class of words. Then we provide the reduction.

6.3.1 FO satisfiability on words

In order to define FO satisfiability on words, we need to define precisely τ-structures that correspond to words.

For a finite alphabet Σ, let $\tau(\Sigma)$ be the vocabulary consisting of a binary relation symbol $\leq$, and a unary relation symbol E_a for every $a \in \Sigma$. A word structure over Σ is a $\tau(\Sigma)$-structure $\mathfrak{W}$ with the following properties:

- W is a finite interval $[0, N]$ for some $N \in \mathbb{N}$,
- $<^{\mathfrak{W}}$ is the natural order of $\mathbb{N}$,
- For every $i \in W$ there exists precisely one $a \in \Sigma$ such that $i \in E_a^{\mathfrak{W}}$.

We refer to elements $i \in W$ as the *positions* in the word (structure) and, for every position $i \in W$, to the unique a such that $i \in E_a^{\mathfrak{W}}$ as the letter at i.

There is an obvious one to one correspondance between any word w from the set Σ^* of all words over Σ and any word structure over Σ. We identify words with the corresponding word structures and write $w \in \Sigma^*$ to refer both to the word and the structure.

It is well-known that if we are interested in the complexity of first-order or monadic second-order model checking or satisfiability on words, the alphabet can be assumed to be $\{0, 1\}$ without loss of generality. Thus when considering the satisfiability problem for first-order logic on the class of words, we can restrict ourselves to the alphabet $\{0, 1\}^*$, and only consider the following problem.

FO SATISFIABILITY ON WORDS
INPUT: A $\mathsf{FO}[\tau(\{0,1\})]$-sentence ϕ.
QUESTION: Does there exist a word $w \in \{0,1\}^*$, such that $w \models \phi$?

The result that is previously known and motivates the upcoming reduction consists in the following.

Theorem 82. *[99]*
The FO SATISFIABILITY ON WORDS *problem is not in* ELEMENTARY.

6.3.2 Reduction from FO SAT on words

We consider reduction towards the problem of PARAMETRIC PUSHDOWN REACHABILITY GAME. The following theorem states a polynomial time reduction from FO SATISFIABILITY ON WORDS to PARAMETRIC PUSHDOWN REACHABILITY GAME.

Theorem 83. FO SATISFIABILITY ON WORDS *is polynomial time reducible to* PARAMETRIC PUSHDOWN REACHABILITY GAME.

Proof. We start with a $\mathsf{FO}[\tau(\{0,1\})]$-formula ϕ. We construct a parametric pushdown automaton $\mathcal{Z}$, disjoint unions $Q = Q_0 \uplus Q_1$ and $P = P_0 \uplus P_1$, and a priority mapping Ω. We do so such that player 0 has a winning strategy from $\mu_\perp$ in the parity game $\mathcal{G} = (A_{(\mathcal{Z},P_0,P_1,Q_0,Q_1)}, \mathrm{WIN}_{\widehat{\Omega}})$ if and only if there exists a word $w \in \{0,1\}^*$ such that $w \models \phi$, where $A_{(\mathcal{Z},P_0,P_1,Q_0,Q_1)}$ and $\widehat{\Omega}$ correspond to the definitions on page 107.

We assume without loss of generality that $\phi = \forall x_1 \exists x_2 \forall x_3 \dots \exists x_k \; \psi$ is written in the prenex normal form, where an even number k of quantifiers alternate between existential and universal ones, and that ψ is quantifier-free and in disjunctive normal form, i.e.

$$\phi = \forall x_1 \exists x_2 \forall x_3 \dots \exists x_k \bigvee_{i=1}^{l} \Big(\bigwedge_{j=1}^{h_i} \psi_{i,j}(x_1, \dots, x_k) \Big).$$

Our construction will include the following states and gadgets.

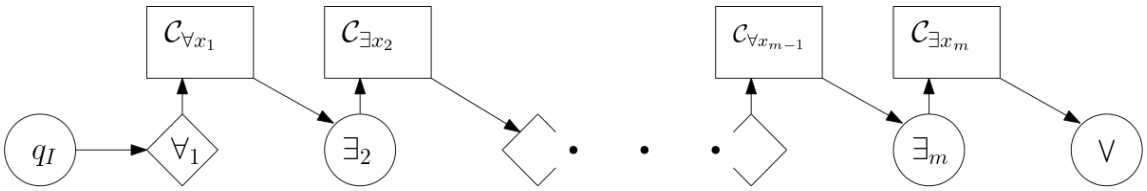

Figure 6.1: The begining of the automaton, and thus, of the game, essentially corresponds to a preprocessing steps dealing with the quantifiers. Players take turns proving the parameters chosen correspond to prefixes of w. Once every variable has been checked, players enters the second part of the game.

q_{init}	the initial state
$\forall_i$	$\forall i \in \{1, 3, \ldots, k-1\}$
$\exists_i$	$\forall i \in \{2, 4 \ldots, k\}$
$\mathcal{C}_{\forall x_i}$, a gadget	$\forall i \in \{1, 3, \ldots, k-1\}$
$\mathcal{C}_{\exists x_i}$, a gadget	$\forall i \in \{2, 4 \ldots, k\}$
$\vee$	
$\wedge_i$	$\forall i \in \{1, \ldots, l\}$
$\mathcal{C}_{\psi_{i,j}}$, a gadget	$\forall i \in \{1, \ldots, l\} \forall j \in \{1, \ldots, h_i\}$

Player 0 starts by chosing a valuation w for the first parameter (parameter p_0). The goal of this first parameter is to remember the word w, which corresponds to the word "guessed" by player 0. This parameter is meant in part to enforce that the stack remains a prefix of w, with each stack content of the play corresponding to a position in the word. To enforce this, we allow in $\mathcal{Z}$, after any decision, for a player to challenge the assumption that the last movement of the other player led to a position in the word. A player thus challenged only has the ability to add things onto the stack, and win if and only if able to find a way back to the value of parameter p_0. The gadgets for these intermediary potential challenges will be excluded from the illustrations to preserve clarity of representation. The other parameters are meant to correspond to the variables, with player 1 parameters corresponding to universal variables and player 0 parameters corresponding to existential variables.

Once players have chosen values for their respective variables, the parameter assignment μ is fixed. It is then time for player 0 to ensure that w satisfies ψ if the variables $x_1, \ldots, x_k$ are interpreted by $|\mu(p_1)| - 1, \ldots, |\mu(p_k)| - 1$ respectively. This is the goal of the reachability game.

The first step of the game is to check that the values assigned to the parameters correspond indeed to possible variables, that is, that the values of the parameters are prefixes of w. See Figure 6.1 for an illustration of what is happening.

Recall ψ is quantifier-free and in disjunctive normal form, i.e.

$$\phi = \forall x_1 \exists x_2 \forall x_3 \ldots \exists x_k \bigvee_{i=1}^{l} (\bigwedge_{j=1}^{h_i} \psi_{i,j}(x_1, \ldots, x_k)).$$

Player 0, being the existential player, chooses one of the conjunctive clauses in ψ, essentially claiming to be able to prove it. Then player 1, the universal player, chooses one of the atomic formulas in the clause to test. The process can be seen in Figure 6.2. Testing the atomic formula is the purpose of a gadget $\mathcal{C}_{\psi_{i,j}}$ built such that player 0 has a winning strategy in $\mathcal{C}_{\psi_{i,j}}$ if and only if w satisfies $\psi_{i,j}$ if the variables $x_1, \ldots, x_k$ are interpreted by $|\mu(p_1)| - 1, \ldots, |\mu(p_k)| - 1$ respectively.

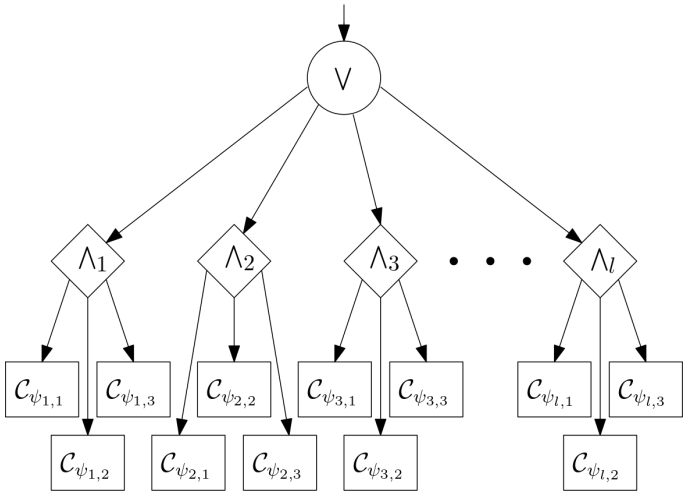

Figure 6.2: Player 0 chooses which conjunctive clause to test and player 1 chooses which of the atomic formula in the clause to test. State $\vee$ belongs to player 0. For $i \in \{1, \ldots, l\}$, state $\wedge_i$ belongs to player 1.

Now we need only to describe the gadgets $\mathcal{C}_{\psi_{i,j}}$. The gadget depends on the type of the atomic formula. For a formula checking the letter at the position of a variable x, player 0 goes to the position corresponding to x, then checks that the top of the stack x is the right letter. For one checking that $x = x'$ where x and x' are both variables, player 0 goes to the position corresponding to x and checks against the parameter corresponding to x' as well. If the formula is a negation, players exchange their roles.

Thus we can in polynomial time in $|\phi|$ build a parametric pushdown automaton $\mathcal{Z}$ with $k+1$ parameters, disjoint unions $Q = Q_0 \uplus Q_1$ and $P = P_0 \uplus P_1$, and a priority mapping Ω such that player 0 has a winning strategy from $\mu_\perp$ in the parity game $\mathcal{G} = (A_{(\mathcal{Z},P_0,P_1,Q_0,Q_1)}, \mathrm{WIN}_{\widehat{\Omega}})$ if and only if there exists a word $w \in \{0,1\}^*$ such that $w \vDash \phi$. □

Then, the nonelementary lower bound follows from Theorem 82.

Theorem 84. Parametric pushdown reachability game *is not in* ELEMENTARY.

As a consequence, solving parametric pushdown parity games is nonelementary too.

Corollary 85. Parametric pushdown parity game *is not in* ELEMENTARY.

6.4 Reduction to pebble pushdown automata parity games

A pebble automaton is a two-way finite state automaton that uses a fixed, finite number of pebbles that it can drop on, and lift from words, using them as markers. Pebble automata recognize regular languages only, provided the life times of the pebbles, i.e. the times between dropping a pebble and lifting it again, are properly nested [46, 38]. Automata with nested pebbles were also introduced for tree-walking automata. It is known that tree-walking automata do not recognize all regular tree languages [11]. Using pebbles is a remedy against getting lost along a tree, but the unrestricted use of pebbles leads to a class of tree languages much larger than the regular tree languages, in fact to all tree languages in NSPACE($log\ n$). Thus, in both pebble word automata and pebble tree-walking automata, the placement of the pebble follows a strict stack discipline. It is traditional hence to represent syntactically the dropping and lifting of pebbles by operations *lift* and *drop*; a

drop simply records the current position with a fresh pebble (such a pebble should be available) and a *lift* pops the last dropped pebble if the current position corresponds to the one recorded by it.

In this section, we extend these ideas to pushdown automata. Instead of using pebbles as markings on their input, pebble pushdown automata have the ability to lift or drop pebbles on their universe of stack contents; a *drop* simply records the current stack content with a fresh pebble (such a pebble should be available) and a *lift* pops the last dropped pebble while requiring that it was placed on the current node. One can think of a pebble as a register that can store a stack content for later comparisons.

We first define more formally our pebble pushdown automaton framework, and then provide a reduction from the problem of solving parametric pushdown parity games to the problem of solving pebble pushdown parity games.

6.4.1 Pebble pushdown automata

An *n-pebble pushdown automaton* is a tuple $\mathcal{I} = (Q, \Gamma, R, q_{init}, \gamma_{init}, F)$ where

- Q is a *finite set of states*,
- Γ is a *finite stack alphabet*,
- $R \subseteq Q \times \Gamma \times \{0, 1, \dots n\} \times \mathcal{P}(\{1, \dots, n\}) \times Q \times (\mathsf{Op}(\Gamma) \cup \{drop, lift\})$ is a *finite set of rules*, where the fourth element S of a rule r is a subset of $\mathcal{P}(\{1, \dots, i\})$ where i is the third element of r, and if the last element is *lift* then we additionally require $i \in S$,
- $q_{init} \in Q$ is an *initial state*,
- $\gamma_{init} \in \Gamma$ is an *initial stack symbol*, and
- $F \subseteq Q$ is a *set of final states*.

Recall that for a partial function $f : S \rightharpoonup X$ (see page 12), for notational purposes, we consider some element $\bot_X \notin X$ and associate f with the function returning the bottom element $\bot_X$ when f is undefined. Thus we write $(X \uplus \{\bot_X\})^S$ for the set of all partial functions from S to X, which we here abbreviate as $(X \uplus \{\bot\})^S$.

By $\mathsf{Conf}(\mathcal{I}) = Q \times \Gamma^* \times (\Gamma^* \uplus \{\bot\})^{\{1,\dots,n\}}$ we denote the set of *configurations* of $\mathcal{I}$. As expected, we rather write $q(w, \mu)$ instead of (q, w, μ). An *i-configuration* for $i > 0$ of $\mathcal{I}$ is a configuration $q(z, \mu)$ where $\mathrm{Dom}(\mu) = \{1, \dots, i\}$, while a 0-configuration is a configuration $q(z, \mu)$ where $\mathrm{Dom}(\mu) = \varnothing$.

The idea of a transition $(q, a, i, S, q', m) \in R$ is that, if the automaton $\mathcal{I}$ is in state q with pebbles $1, \dots, i$ dropped — or without pebble dropped if $i = 0$ — with top stack symbol a, and stack content which corresponds to the stack contents of the pebbles from S and only these pebbles, then $\mathcal{I}$ goes to state q' and makes modifications to the stack or the pebbles according to m. Note a pebble can be lifted only if the stack content which corresponds to the pebble is the same as the current stack content. This is enforced by syntactically requiring the last pebble dropped i is in the set S used for testing the presence of certain pebbles.

A *pebble set* of $\mathcal{I}$ is a set $U \subseteq \{1, \dots, n\}$. For a stack alphabet Γ, a *U-pebble assignment* is a function which maps each $j \in U$ to a word in Γ^*. The $\varnothing$-pebble assignment is denoted by $\mu_{init} : \{1, \dots, n\} \rightharpoonup \Gamma^*$ and is the totally undefined function.

An n-pebble pushdown automaton $\mathcal{I} = (Q, \Gamma, R, q_{init}, \gamma_{init}, F)$ induces a transition system $T_{\mathcal{I}} = (\mathsf{Conf}(\mathcal{I}), \rightarrow_{\mathcal{I}})$ such that for all $(q, a, i, S, q', m) \in R$, with $a \in \Gamma$, for all words $w \in \Gamma^*$, and for all $\{1, \dots, i\}$-pebble assignments μ such that $\mu(j) = wa$ for each $j \in S$ and $\mu(j) \neq wa$ for each $j \in \{1, \dots, i\} \setminus S$, the following holds

- if $m \in \mathsf{Op}(\Gamma)$, either
  - $m = push^{\gamma}$ and $(q, wa, \mu) \rightarrow_{\mathcal{I}} (q', wa\gamma, \mu)$,
  - $m = pop$ and $(q, wa, \mu) \rightarrow_{\mathcal{I}} (q', w, \mu)$, or
  - $m = skip$ and $(q, wa, \mu) \rightarrow_{\mathcal{I}} (q', wa, \mu)$,
- if $m = drop$, $(q, wa, \mu) \rightarrow_{\mathcal{I}} (q', wa, \mu')$, where μ' is the $\{1, \dots, i, i+1\}$-pebble assignment such that $\mu'(j) = \mu(j)$, for each $j \leq i$, and $\mu'(i+1) = wa$, and
- if $m = lift$, and $i \in S$, i.e. the last pebble dropped belong of the set of pebble we test the presence of, $(q, wa, \mu) \rightarrow_{\mathcal{I}} (q', wa, \mu')$, where μ' is the $\{1, \dots, i-1\}$-pebble assignment such that $\mu'(j) = \mu(j)$, for each $j < i$.

We are interested in games over pebble pushdown automata, mainly, parity games.

Again, given a transition system, one needs only to provide a partition of the set of configurations to obtain an arena. Given a partition of Q into Q_0 and Q_1, we partition the configurations of $T_{\mathcal{I}}$ into $\mathsf{Conf}_{\mathcal{I},0} = Q_0 \times \Gamma^* \times (\Gamma^* \uplus \{\bot\})^{\{1,\dots n\}}$ and $\mathsf{Conf}_{\mathcal{I},1} = Q_1 \times \Gamma^* \times (\Gamma^* \uplus \{\bot\})^{\{1,\dots n\}}$.

With these notations in mind one can define the arena

$$A_{(\mathcal{I},Q_0,Q_1)} = (\mathsf{Conf}_{\mathcal{I},0}, \mathsf{Conf}_{\mathcal{I},1}, \rightarrow_{\mathcal{I}}).$$

As expected, given a priority function $\Omega : Q \to \{0, \dots m\}$, one naturally set the extension of Ω as $\overline{\Omega} : \mathsf{Conf}(\mathcal{I}) \to \{0, \dots m\}$ and $\overline{\Omega}(q, w, \mu) = \Omega(q)$ for all $w \in \Gamma^*$ and $\mu \in (\Gamma^* \uplus \{\bot\})^{\{1,\dots,n\}}$.

Concerning pebble pushdown automata, we are interested in the following decision problem.

n-PEBBLE PUSHDOWN PARITY GAME

INPUT: An n-pebble pushdown automaton $\mathcal{I} = (Q, \Gamma, R, q_{init}, \gamma_{init}, F)$, where $Q = Q_0 \uplus Q_1$, and a priority mapping $\Omega : Q \to \{0, \dots m\}$.

QUESTION: Does player 0 have a winning strategy from $q_{init}(\gamma_{init}, \mu_{init})$ for the parity game $\mathcal{G} = (A_{(\mathcal{I},Q_0,Q_1)}, \textsc{Win}_{\overline{\Omega}})$?

6.4.2 From parametric pushdown automata to pebble pushdown automata

We now provide a reduction from the problem of solving parametric pushdown parity games to the problem of solving pebble pushdown parity games.

Theorem 86. n-PARAMETRIC PUSHDOWN PARITY GAME *is polynomial time reducible to* n-PEBBLE PUSHDOWN PARITY GAME.

Sketch. Let us fix some n-parametric pushdown automaton $\mathcal{Z} = (Q, \Gamma, P, q_{init}, \gamma_{init}, F)$, disjoint union $Q = Q_0 \uplus Q_1$ and $P = P_0 \uplus P_1$, and some priority function $\Omega : Q \times \Gamma^* \times (\Gamma^* \uplus \{\bot\})^R \to [0, m]$.

We construct an n-pebble pushdown automaton $\mathcal{I} = (Q', \Gamma, R', q'_{init}, \gamma_{init}, F')$, a disjoint union $Q' = Q'_0 \uplus Q'_1$ and a mapping $\Omega' : Q \times \Gamma'^* \times (\Gamma'^* \uplus \{\bot\})^{\{1,\dots,n\}} \to \{1, \dots, m\}$, such that player 0 has a winning strategy from $(q'_{init}, \gamma_{init}, \mu_{init})$ in the parity game $\mathcal{G}' = (A_{(\mathcal{I},Q'_0,Q'_1)}, \textsc{Win}_{\overline{\Omega'}})$ if and only if player 0 has a winning strategy from $\mu_{\bot}$ in the parity game $\mathcal{G} = (A_{(\mathcal{Z},P_0,P_1,Q_0,Q_1)}, \textsc{Win}_{\widehat{\Omega}})$, where $A_{(\mathcal{Z},P_0,P_1,Q_0,Q_1)}$ and $\widehat{\Omega}$ correspond to the definitions on page 107.

The transition graph of the n-pebble pushdown automaton $\mathcal{I}$ will simulate the possible transitions graphs (one for every parameter assignment) of the pushdown automaton with n parameters by using pebbles to represent the parameters. The n-pebble pushdown automaton will first simulate

the parameter valuation arena. Players take turns chosing particular stack contents on which to place pebbles, using a gadget similar as the one from Figure 6.1. Once every parameter has been assigned with the dropping of a corresponding pebble, the pebble assignment μ is fixed.

Further configurations in the n-pebble pushdown automaton then consist of a word representing the status of the stack, a state, and a fixed pebble assignment. For a fixed pebble assignment there is then a one for one correspondance between configurations of the n-parametric pushdown automaton and these of the n-pebble pushdown automaton. The set of states of the n-pebble pushdown automaton apart from the initial gadget is the same as the set of states of the n-parametric pushdown automaton, and so are player 0 states, player 1 states and the priority mapping. □

6.5 Reduction to higher-order pushdown automata

Higher-order pushdown automata (HPDA for short) were introduced as a generalization of pushdown automata [1, 51, 77]. A stack of a pushdown automaton is seen as a *level* 1 *stack*. A pushdown automaton of level 2 (or 2-HPDA) then works with a stack of level 1 stacks. In addition to the ability to push and to pop a symbol on the top-most level 1 stack, an 2-HPDA can copy or remove the entire topmost level 1 stack. The definition generalizes to any $n \geq 2$, and n-HPDA are similarly defined for all level n as automata working with a stack of level $(n-1)$ stacks.

We recall the definition from [20] which itself is taken from [67]. We then provide a reduction from the problem of solving pebble pushdown parity games to the problem of solving higher-order pushdown parity games.

6.5.1 Higher-order pushdown automata

A *level* 1 *stack* (or 1-*stack*) over an alphabet Γ is simply a stack over Γ, i.e. a word in Γ^*. A *level* n *stack* (or n-*stack*) over an alphabet Γ, for $n \geq 2$, is a non-empty sequence $\langle s_0\rangle\langle s_1\rangle \dots \langle s_m\rangle$ of $(n-1)$-stacks over Γ, for some $m \in \mathbb{N}$. The set of n-stacks over Γ is denoted by $\mathscr{S}_n(\Gamma)$, or simply $\mathscr{S}_n$ in case the set Γ is obvious from context. The set of all stacks over Γ is written $\mathscr{S}(\Gamma) = \bigcup_{n\in\mathbb{N}} \mathscr{S}_n(\Gamma)$. We define ϵ_1 as $\epsilon \in \Gamma^*$ and we inductively define $\epsilon_n = \langle \epsilon_{n-1}\rangle$ in $\mathscr{S}_n$ for all $n > 1$.

A *higher-order stack operation* is a partial function from $\mathscr{S}(\Gamma)$ to $\mathscr{S}(\Gamma)$ which preserves the level of the input (i.e. the image of an n-stack is an n-stack for all $n \in \mathbb{N}$). The *level* of an operation op is the smallest $n \in \mathbb{N}$ such that $\mathrm{Dom}(op) \cap \mathscr{S}_n \neq \varnothing$. The operations additionally respect the hierarchicality of higher-order stacks, i.e. in a level $n+1$ stack only the topmost level n stack can be accessed. An operation op of level n, applied to a level $n+1$ stack $\langle s_0\rangle\langle s_1\rangle \dots \langle s_m\rangle$ of length $m \in \mathbb{N}$, thus returns the output $\langle s_0\rangle\langle s_1\rangle \dots \langle op(s_m)\rangle$ if applicable. The definition for all levels of stacks greater than n follows the same pattern.

The following operations can be performed on a 1-stacks of length $m \in \mathbb{N}$.

$$\begin{aligned} push_1^\gamma(w_0 w_1 \dots w_m) &= w_0 w_1 \dots w_m \gamma \text{ for all } \gamma \in \Gamma, \\ pop_1(w_0 w_1 \dots w_{m-1} w_m) &= w_0 w_1 \dots w_{m-1}, \text{ if } m \geq 1 \\ top(w_0 w_1 \dots w_m) &= w_m. \end{aligned}$$

The operations added at level $n+1$ are the copy of the topmost n-stack and the removal of the topmost n-stack. More formally, if $\langle s_0\rangle\langle s_1\rangle \dots \langle s_m\rangle$ is a stack of level $n > 1$, the following operations are possible.

$$\begin{aligned} push_n(\langle s_0\rangle\langle s_1\rangle \dots \langle s_m\rangle) &= \langle s_0\rangle\langle s_1\rangle \dots \langle s_m\rangle\langle s_m\rangle, \\ push_j(\langle s_0\rangle\langle s_1\rangle \dots \langle s_m\rangle) &= \langle s_0\rangle, \langle s_1\rangle \dots \langle push_j(s_m)\rangle, \text{ if } 2 \leq j < n \\ push_1^\gamma(\langle s_0\rangle\langle s_1\rangle \dots \langle s_m\rangle) &= \langle s_0\rangle\langle s_1\rangle \dots \langle push_1^\gamma(s_m)\rangle, \text{ for all } \gamma \in \Gamma, \\ pop_n(\langle s_0\rangle\langle s_1\rangle \dots \langle s_{m-1}\rangle\langle s_m\rangle) &= \langle s_0\rangle\langle s_1\rangle \dots \langle s_{m-1}\rangle, \\ pop_j(\langle s_0\rangle\langle s_1\rangle \dots \langle s_{m-1}\rangle\langle s_m\rangle) &= \langle s_0\rangle\langle s_1\rangle \dots \langle s_{m-1}\rangle\langle pop_j(s_m)\rangle, \text{ if } 1 \leq j < n \\ top(\langle s_0\rangle\langle s_1\rangle \dots \langle s_m\rangle) &= top(s_m). \end{aligned}$$

The operations pop_1 and top are undefined on a stack whose top 1-stack is empty.

Given a stack alphabet Γ and $n \in \mathbb{N}$, we denote by $\mathsf{Op}_n(\Gamma)$ the *base set of n-stack operations* as

- $push_j$ for all $2 \leq j \leq n$,
- $push_1^\gamma$ for all $\gamma \in \Gamma$,
- pop_j for all $1 \leq j \leq n$,
- $skip$, corresponding to the identity function of $\mathscr{S}(\Gamma)$.

A *higher-order pushdown automata of level n* (or n-HPDA for short) is a tuple $\mathcal{H} = (Q, \Gamma, R, q_{init}, \gamma_{init}, F)$, where

- Q is a non-empty finite *set of states*,
- Γ is a non-empty finite *stack alphabet*,
- $R \subseteq Q \times \Gamma \times Q \times \mathsf{Op}_n(\Gamma)$ is a finite *set of rules*,
- q_{init} is the *initial state*,
- $\gamma_{init} \in \Gamma$ is the *initial stack symbol*, and
- $F \subseteq Q$ is a *set of final states*.

By $\mathsf{Conf}(\mathcal{H}) = Q \times \mathscr{S}_n(\Gamma)$ we denote the set of *configurations* of an n-HPDA $\mathcal{H}$. As usual we abbreviate $(q, s) \in \mathsf{Conf}(\mathcal{H})$ as $q(s)$.

An n-HPDA $\mathcal{H} = (Q, \Gamma, R, q_{init}, F)$ induces the transition system $\mathcal{T}_\mathcal{H} = (Conf(\mathcal{H}), \rightarrow_\mathcal{H})$ where for all q, q' in Q, for all s, s' in $\mathscr{S}_n(\Gamma)$, $q(s) \rightarrow_\mathcal{H} q'(s')$ if there exists some rule $(q, \gamma, q', op) \in R$ such that $top(s) = \gamma$ and $s' = op(s)$.

Again we are interested in parity games. As expected, given an n-HPDA $\mathcal{H}$ and a partition of Q into Q_0 and Q_1, we partition the configurations of $T_\mathcal{H}$ into $\mathsf{Conf}_{\mathcal{H},0} = Q_0 \times \mathscr{S}_n(\Gamma)$ and $\mathsf{Conf}_{\mathcal{H},1} = Q_1 \times \mathscr{S}_n(\Gamma)$. With these notations in mind one can define the arena

$$A_{(\mathcal{H},Q_0,Q_1)} = (\mathsf{Conf}_{\mathcal{H},0}, \mathsf{Conf}_{\mathcal{H},1}, \rightarrow_\mathcal{H})$$

induced by an n-HPDA $\mathcal{H}$ and a partition of its set of states.

As expected, given a priority function $\Omega : Q \to [0, m]$, we naturally extend the function as follows, by setting $\Omega_{\mathscr{S}_n(\Gamma)} : \mathsf{Conf}(\mathcal{H}) \to [0, m]$ and $\Omega_{\mathscr{S}_n(\Gamma)}(q, s) = \Omega(q)$ for all $s \in \mathscr{S}_n(\Gamma)$.

Concerning higher-order pushdown automata, we are interested in the following decision problem.

n-HPDA PARITY GAME

INPUT: An n-HPDA $\mathcal{H} = (Q, \Gamma, R, q_{init}, \gamma_{init}, F)$, where $Q = Q_0 \uplus Q_1$, and a priority mapping $\Omega : Q \to \{0, \dots m\}$.

QUESTION: Does player 0 have a winning strategy from $q_{init}(push_1^{\gamma_{init}}(\epsilon_n))$ for the parity game $\mathcal{G} = (A_{(\mathcal{H},Q_0,Q_1)}, \textsc{Win}_{\Omega_{\mathscr{S}_n(\Gamma)}})$?

It was shown in [19] that n-HPDA PARITY GAME can be solved in n-EXP. This also gives an n-EXP algorithm for the μ-calculus model checking over transitions systems induced by n-HPDA. In [20] the matching lower bound was showed, even in the case of reachability games, hence showing n-EXP-completeness of n-HPDA PARITY GAME.

Theorem 87. *[19, 20] n-HPDA PARITY GAME is n-EXP-complete.*

In a 2-HPDA, the operation $push_2$ allows to "copy" the top level 1 stack. The current word is hereby stored away and left untouched until the next operation pop_2, while $push_1$ and pop_1 can be

performed on the additional "copy" in the meantime. This behavior is similar to that of dropping and lifting a pebble. The main difference is that there is no operation to test that the "copy", after many updates, is again equal to the "original", i.e. there is no operation to syntactically test that the two topmost level 1 stacks are identical.

In [22, 106, 21] however Carayol and Wöhrle introduced a variant of n-HPDA by extending the pop_j operations for $2 \leq j \leq n$ with a built-in equality test. For $2 \leq j \leq n$ the new operation $pop_j^=$ has the same effect as pop_j, but can only be applied if the two top level j stacks coincide. In [21] it is seen as a symmetrical operation in comparison to $push_j$.

A *higher-order pushdown automaton with equality pop of level n* (n-HPDA$^=$ for short) is a higher-order pushdown automaton of level n where in $\mathsf{Op}_n(\Gamma)$ the operation pop_j is replaced by $push_j^=$ for $2 \leq j \leq n$. We denote this new set of operations by $\mathsf{Op}_n^=(\Gamma)$. More formally the new operations $push_j^=$ are defined as

$$\begin{aligned} pop_k^=(\langle s_0\rangle\langle s_1\rangle \dots \langle s_m\rangle\langle s_m\rangle) &= \langle s_0\rangle\langle s_1\rangle \dots \langle s_m\rangle, \quad \text{and} \\ pop_j^=(\langle s_0\rangle\langle s_1\rangle \dots \langle s_m\rangle) &= \langle s_0\rangle\langle s_1\rangle \dots \langle pop_j^=(s_m)\rangle, \text{ if } 2 \leq j < k. \end{aligned}$$

Transition systems induced by higher-order pushdown automata with equality pop of level n are then defined as expected. Carayol and Wöhrle proved [106, 21] that the two models, namely higher-order pushdown automata with equality pop of level n and higher-order pushdown automata of level n, generate the same classes of transition systems.

Theorem 88. *[106, 21] If $\mathcal{H}$ is an n-HPDA (resp. n-HPDA$^=$) then there exists an n-HPDA$^=$ (resp. n-HPDA) $\mathcal{H}'$ such that $\mathcal{H}$ and $\mathcal{H}'$ induce isomorphic transition systems.*

From n-HPDA to n-HPDA$^=$ (Proposition 3.12 in [106]) the proof relies on recreating a correct stack content to simulate higher-order *pop* operations by $pop^=$, essentially "guessing" the correct stack content to be able to apply $pop^=$. In the other direction (Proposition 3.18 in [106]), the author enrich the stack alphabet with new symbols stating which instructions have to be executed to recreate a previous stack content. It is proven that such en encoding is possible since there is for every stack s of level n a unique shortest sequence of instructions which creates s from ϵ_n.

Defined like n-HPDA PARITY GAME, we write n-HPDA$^=$ PARITY GAME in case the input is an n-HPDA$^=$ rather than an n-HPDA.

The algorithm from [19] actually provides an algorithmic solution to parity games on the graphs of the Caucal hierarchy. We skip a formal definition of a graph of level n in the Caucal hierarchy, and refer the reader to [23] for more details. The n-EXP upper bound on n-HPDA parity games follows from the fact that every transition system induced by an n-HPDA $\mathcal{H}$ is a graph of the Caucal hierarchy [19, 106], whose vertices are almost in one-to-one correspondence with the configurations of $\mathcal{H}$. In [106] the converse direction is proven, i.e. that every graph of level n of the Caucal hierarchy is generated by an n-HPDA. In [21] it is similarly shown that every transition system induced by an n-HPDA$^=$ $\mathcal{H}$ is a graph of the Caucal hierarchy. The algorithm from [19] hence lead to a solution for n-HPDA$^=$ parity games as well.

Theorem 89. n-HPDA$^=$ PARITY GAME *is in* n-EXP.

6.5.2 From pebble pushdown automata to higher-order pushdown automata

We use HPDA$^=$ parity games to simulate pebble pushdown automata parity games. A similar approach was used in [21] to show that $(n+2)$-level stack automata could be used to simulate n-pebble alternating two-way word automata.

Let us start by discussing the simulation of the dropping and lifting of a pebble in the case of a pebble pushdown automaton $\mathcal{I}$ with only one pebble. As long as no pebble is dropped, a 2-HPDA$^=$ $\mathcal{H}$ can simulate the behavior of a pebble pushdown automaton $\mathcal{I}$ with a 2-stack containing a single

1-stack on which it performs the same operations $\mathcal{I}$ performs on its stack. When the automaton $\mathcal{I}$ is in configuration $q(w)$ and drops a pebble, instead of making a pop or a push, we need to store the information that the pebble has been dropped on w. To simulate the next configuration in such a case, we use $push_2$ to store the information that the pebble has been dropped on the word w, leading to the new 2-stack $\langle w\rangle\langle w\rangle$. Configurations afterwards have 2-stacks of length two where the first component remains w until the pebble is lifted. Lifting the pebble can be done only at the position the pebble was dropped, by using $pop_2^=$. Checking that the node corresponds to the one where the pebble has been dropped simply makes use of the composition of $pop_2^=$ and $push_2$. Checking the absence of the pebble can be done by challenging the opponent to prove the presence of the pebble.

Now, let us detail the case when there is a second pebble, the construction for each additional pebble being highly similar, where every additional pebble after the first one would require the addition of another stack level. Similarly as before, as long as no pebble is dropped, the stack of $\mathcal{H}$ is a 3-stack containing a single 2-stack that contains a single 1-stack. To simulate configurations in the case one pebble have been dropped, the stack contains a single 2-stack of the form $\langle w_1\rangle\langle w\rangle$. In the case the two pebbles have been dropped, the stack is of the form $\langle\langle w_1\rangle\langle w_2\rangle\rangle\langle\langle w_1\rangle\langle w\rangle\rangle$.

Intuitively speaking, w_1 is, as before, the position where the first pebble is placed, and w_2 is where the second pebble is placed. Dropping pebbles like this again involves the cloning operation, only, each pebble operates at a different stack level: thus, dropping the first pebble will use $push_2$ and dropping the second pebble will use $push_3$. Knowing the number of pebbles dropped can be done by keeping track using the states of $\mathcal{H}$. Lifting or checking for the presence (or absence) of the pebbles functions on a similar basis as before. Checking the presence of the first pebble uses operation $pop_2^=$ followed by $push_2$, while checking the presence of the second pebble uses operation $pop_3^=$ followed by $push_3$. Lifting the first pebble uses $pop_2^=$, but check first that the second pebble has already been lifted. Lifting the second pebble simply uses $pop_3^=$.

By expanding this reasoning inductively, we conclude that given $n \in N$, and given an n-pebble pushdown automaton $\mathcal{I} = (Q, \Gamma, R, q_{init}, \gamma_{init}, F)$, where $Q = Q_0 \uplus Q_1$, one can compute an $(n+1)$-HPDA $\mathcal{H} = (Q', \Gamma, R', q'_{init}, \gamma'_{init}, F')$ where $Q' = Q'_0 \uplus Q'_1$, such that player 0 has a winning strategy from $q'_{init}(push_1^{\gamma'_{init}}(\epsilon_n))$ in $A_{(\mathcal{H},Q'_0,Q'_1)}$ if and only if player 0 has a winning strategy from $q(\gamma_{init}, \mu_{init})$ in $A_{(\mathcal{I},Q_0,Q_1)}$. Hence the following reduction.

Theorem 90. n-PEBBLE PUSHDOWN PARITY GAME *is polynomial time reducible to* $(n+1)$-HPDA$^=$ PARITY GAME.

The reduction implies decidability of pebble pushdown automata parity game. Furthermore, by Theorem 89, it implies the following complexity result.

Theorem 91. n-PEBBLE PUSHDOWN AUTOMATA PARITY GAME *is in* $(n+1)$-EXP.

Finally, the following complexity result is due the above and Theorem 86.

Corollary 92. n-PARAMETRIC PUSHDOWN AUTOMATA PARITY GAME *is in* $(n+1)$-EXP.

6.6 Discussion and open problems

In this section we have shown that deciding the winner of a parametric pushdown parity game or reachability game is nonelementary in general, but decidable and in $(n+1)$-NEXP when the number n of parameters is fixed.

For the lower bound we reduced the FO satisfiability problem on words — known to be nonelementary from [99] — to the problem of deciding whether player 0 has a winning strategy for a parametric pushdown reachability game.

For the decidability upper bound, we used pebble pushdown parity games to solve parametric pushdown parity games, and higher-order pushdown parity games to solve pebble pushdown parity games. Since solving parity games on higher-order pushdown automata with level n stack is

n-EXP-complete [19, 20], this provides an $(n+1)$-EXP upper bound for solving parametric parity games on parametric pushdown automata with n parameters.

Since a particular case of pushdown automata consists in one-counter automata, extensions of automata with a counter generally inherit known upper bounds on pushdown automata. This however is not the case in the parametric extensions considered in this thesis, as parametric updates are not trivially handled by our PPDA model, i.e. a PPDA over a unary alphabet plus a bottom-of-stack symbol is not trivially a PTOCA nor a POCA. For solving parametric pushdown parity games, on the other hand, we introduced the notion of pebble pushdown parity games, and pebble pushdown automata over unary alphabet plus a bottom-of-stack symbol can be seen as a form of one-counter automata extended with pebbles. A pebble one-counter automata is then a one-counter automata that can use a set of pebbles as markings on the set of non-negative integers, with a *drop* recording the current counter value, and a *lift* poping the last dropped pebble while requiring that the current counter value corresponds to the one from the last dropped pebble. Unlike for parametric extensions, decidability of what should be called pebble one-counter parity games follows from decidability of pebble pushdown parity games. We believe it natural to study such a pebble one-counter automata model on its own. In particular we believe that the question of whether or not solving pebble one-counter reachability games is nonelementary is worth investigating.

Of note is that reachability games and parity games are not the only problems one can consider, as one can also explore the complexity of reachability itself. In particular one can study the precise complexity of the n-PPDA REACHABILITY problem. Since reachability can be viewed as a reachability game, we know that n-PPDA REACHABILITY is decidable in $(n+1)$-EXP, but the precise complexity of the problem remains unknown. We are furthermore convinced that it is worthwhile to investigate further comparisons between parametric pushdown automata, pebble pushdown automata, and register pushdown automata, especially since the latter have been shown to only really "remember" at most $3r$ data values, where r is the number of registers.

Conclusion and perspectives

In this thesis we have studied the impact of adding a set of parameters to the complexity of different verification problems. Most notably, we have studied the reachability problem for PTA and PTOCA. We have filled some gaps in past knowledge, but there remains a lot of open problems to tackle.

The main open problem that remain to be solved is that of reachability in parametric time automata with two parametric clocks and an arbitrary number of parameters.

Open problem 93. *Is* $(2, *)$-PTA REACHABILITY *decidable ?*

The question is highly nontrivial even for the subclass of parametric time automata with two parametric clocks and two parameters. Of note is that it has been shown that there is an easy reduction from the existential fragment of Presburger Arithmetic with divisibility to reachability in PTA over two parametric clocks [5]. For proving an upper bound for $(2, *)$-PTA REACHABILITY, Bundala and Ouaknine [17] have provided a reduction from $(2, n)$-PTA to n-PTOCA, albeit ones with so called $+[0, p]$-transitions that allow to nondeterministically add to the counter a value that lies in $[0, \mu(p)]$, where $\mu(p)$ is the parameter valuation of some parameter p. For the case of one parameter, we proved it was possible to perform the reduction in exponential time without these transitions nor binary updates, but for more than one parameter it might not be possible. Following in the footsteps of Bundala and Ouaknine, it seem to us that the question of $(2, *)$-PTA reachability is better asked in terms of PTOCA reachability, since not only has it been crucial for determining the precise complexity of $(2, 1)$-PTA REACHABILITY, but also since in the nonparametric model a similar reduction [54] was essential for determining the precise complexity of reachability in two-clock timed automata [42].

This lead to the open problem of wether reachability for n-PTOCA allowing $+[0, p]$-transitions is decidable. A subset of this model, and one which seem essential to study, in particular envisionning the possiblity of finding a better reduction from PTA to PTOCA without $+[0, p]$-transitions, is that of PTOCA.

Open problem 94. *Is* PTOCA REACHABILITY *decidable ?*

In [17], the proof that 1-PTOCA REACHABILITY is decidable uses a reduction to existential Presburger arithmetic with divisibility. It is unclear how to generalise this technique to the case of more than one parameter. We hope the generalization of the technique presented in this thesis proves more fruitful, even though it presents several complex obstacles to face. For instance, in the presence of two parameters, one can build a 2-PTOCA for which reachability holds only if the first parameter is a multiple of the second parameter. This means that the parameter values can depend on one another, and then, "de-scaling" the value of a single parameter in a semirun can lead to an inadequation since the two values don't have the same relationship anymore. In this situation, we need to decrease the value of both parameters simultaneously proportionally.

A third open problem is that of reachability in PPDA. Since reachability games are decidable, it follows that reachability in PPDA is decidable. However, the question of the precise complexity of the problem remains to be explored.

Open problem 95. *What is the complexity of* PPDA REACHABILITY *?*

For solving parametric pushdown parity games, we introduced the notion of pebble pushdown parity games. Similarily as for PPDA, the question of the precise complexity of the reachability problem stands out, not only as a way to provide an upper bound for PPDA REACHABILITY, but also by itself.

Open problem 96. *What is the complexity of* PEBBLE PUSHDOWN AUTOMATA REACHABILITY *?*

Additionally, as mentioned in Section 6.6, pebble pushdown automata over unary alphabet plus a bottom-of-stack symbol can be seen as pebble one-counter automata. This contrasts with PPDA which are *a priori* incomparable to PTOCA. While the complexity upper bounds from pebble pushdown parity games apply to pebble one-counter parity games, it is conceivably possible that better complexity bounds exist. In particular, the technique used to prove the nonelementary lower bound do not carry over to the case of an unary alphabet, and it is not clear whether or not pebble one-counter reachability games are nonelementary.

Open problem 97. *Are* PEBBLE ONE-COUNTER PARITY GAME *and* PEBBLE ONE-COUNTER REACHABILITY GAME *in* ELEMENTARY *?*

Bibliography

[1] Alfred V Aho. Nested stack automata. *Journal of the ACM (JACM)*, 16(3):383–406, 1969.

[2] Bowen Alpern and Fred B Schneider. Defining liveness. *Information processing letters*, 21(4):181–185, 1985.

[3] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking for real-time systems. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990*, pages 414–425. IEEE Computer Society, 1990. doi:10.1109/LICS.1990.113766.

[4] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[5] Rajeev Alur, Thomas A Henzinger, and Moshe Y Vardi. Parametric real-time reasoning. In *Proc. STOC'93*, pages 592–601. ACM, 1993.

[6] Étienne André. What's decidable about parametric timed automata? *Int. J. Softw. Tools Technol. Transf.*, 21(2):203–219, 2019. doi:10.1007/s10009-017-0467-0.

[7] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[8] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[9] D. A. M. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38:150–164, 1989.

[10] Nikola Benes, Peter Bezdek, Kim Guldstrand Larsen, and Jirí Srba. Language emptiness of continuous-time parametric timed automata. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2015. doi:10.1007/978-3-662-47666-6_6.

[11] Mikołaj Bojańczyk and Thomas Colcombet. Tree-walking automata do not recognize all regular languages. *SIAM Journal on Computing*, 38(2):658–701, 2008.

[12] Benedikt Bollig, Karin Quaas, and Arnaud Sangnier. The complexity of flat freeze LTL. *Logical Methods in Computer Science*, 15(3):32:1–32:26, 2019. URL: https://arxiv.org/abs/1609.06124, doi:10.23638/LMCS-15(3:32)2019.

[13] Ahmed Bouajjani, Marius Bozga, Peter Habermehl, Radu Iosif, Pierre Moro, and Tomáš Vojnar. Programs with lists are counter automata. In *International Conference on Computer Aided Verification*, pages 517–531. Springer, 2006.

[14] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In Antoni W. Mazurkiewicz and Józef Winkowski,

editors, *Proceedings of the 8th International Conference on Concurrency Theory (CONCUR'97)*, number 1243 in Lecture Notes in Computer Science, pages 135–150. Springer, 1997.

[15] M. Bozga, R. Iosif, and Y. Lakhnech. Flat parametric counter automata. In *Proc. ICALP'06*, volume 4052 of *LNCS*. Springer, 2006.

[16] Julian Bradfield and Igor Walukiewicz. The mu-calculus and model checking. In *Handbook of Model Checking*, pages 871–919. Springer, 2018.

[17] Daniel Bundala and Joel Ouaknine. On parametric timed automata and one-counter machines. *Information and Computation*, 253:272–303, 2017.

[18] Alan Burns. How to verify a safe real-time system: The application of model checking and timed automata to the production cell case study. *Real-time systems*, 24(2):135–151, 2003.

[19] Thierry Cachat. Higher order pushdown automata, the Caucal hierarchy of graphs and parity games. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP 2003), Eindhoven (The Netherlands)*, number 2719 in Lecture Notes in Computer Science, pages 556–569. Springer, 2003.

[20] Thierry Cachat and Igor Walukiewicz. The complexity of games on higher order pushdown automata. *arXiv preprint arXiv:0705.0262*, 2007.

[21] Arnaud Carayol. *Automates infinis, logiques et langages*. PhD thesis, Université Rennes 1, 2006.

[22] Arnaud Carayol and Stefan Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2003), Mumbai (India)*, Lecture Notes in Computer Science. Springer, 2003.

[23] Didier Caucal. On infinite terms having a decidable monadic theory. In Krzysztof Diks and Wojciech Rytter, editors, *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS 2002), Warsaw (Poland)*, number 2420 in Lecture Notes in Computer Science, pages 165–176. Springer, 2002.

[24] Andrew Chiu, George Davida, and Bruce Litow. Division in logspace-uniform NC^1. *Theoretical Informatics and Applications. Informatique Théorique et Applications*, 35(3):259–275, 2001.

[25] Alessandro Cimatti, Luigi Palopoli, and Yusi Ramadian. Symbolic computation of schedulability regions using parametric timed automata. In *2008 Real-Time Systems Symposium*, pages 80–89. IEEE, 2008.

[26] Edmund M Clarke and E Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on logic of programs*, pages 52–71. Springer, 1981.

[27] Edmund M Clarke, E Allen Emerson, and A Prasad Sistla. Automatic verification of finite state concurrent system using temporal logic specifications: a practical approach. In *Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 117–126, 1983.

[28] H. Comon and Y. Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In *Proc. CAV'98*, volume 1427 of *LNCS*. Springer, 1998.

[29] Christopher L Conway, Kedar S Namjoshi, Dennis Dams, and Stephen A Edwards. Incremental algorithms for inter-procedural analysis of safety properties. In *International Conference on Computer Aided Verification*, pages 449–461. Springer, 2005.

[30] Costas Courcoubetis and Mihalis Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1(4):385–415, 1992.

[31] Martin Davis. The undecidable (reprint ed.), 1965.

[32] S. Demri and R. Gascon. The effects of bounding syntactic resources on Presburger LTL. In *Proc. TIME'07*. IEEE Computer Society Press, 2007. doi:10.1109/TIME.2007.63.

[33] Stéphane Demri and Arnaud Sangnier. When model-checking freeze LTL over counter machines becomes decidable. In C.-H. Luke Ong, editor, *Foundations of Software Science and Computational Structures, 13th International Conference, FOSSACS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6014 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2010.

[34] E Allen Emerson. Model checking and the mu-calculus. *Descriptive Complexity and Finite Models*, 31:185–214, 1996.

[35] E Allen Emerson and Edmund M Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *International Colloquium on Automata, Languages, and Programming*, pages 169–181. Springer, 1980.

[36] E Allen Emerson, Charanjit S Jutla, and A Prasad Sistla. On model-checking for fragments of μ-calculus. In *International Conference on Computer Aided Verification*, pages 385–396. Springer, 1993.

[37] E Allen Emerson and CL Lei. Efficient model checking in fragments of the propositional mu-calculus. In *IEEE Symposium on Logic in Computer Science*, pages 267–278. IEEE Computer Society Press, 1986.

[38] Joost Engelfriet and Hendrik Jan Hoogeboom. Tree-walking pebble automata. In *Jewels are forever*, pages 72–83. Springer, 1999.

[39] Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asynchronous shared-memory systems. *Journal of the ACM (JACM)*, 63(1):1–48, 2016.

[40] Javier Esparza, David Hansel, Peter Rossmanith, and Stefan Schwoon. Efficient algorithms for model checking pushdown systems. In E. Allen Emerson and A. Prasad Sistla, editors, *Proceedings of the 12th International Conference on Computer Aided Verification (CAV 2000)*, number 1855 in Lecture Notes in Computer Science, pages 232–247. Springer, 2000.

[41] Javier Esparza and Jens Knoop. An automata-theoretic approach to interprocedural data-flow analysis. In *International Conference on Foundations of Software Science and Computation Structure*, pages 14–30. Springer, 1999.

[42] John Fearnley and Marcin Jurdziński. Reachability in two-clock timed automata is pspace-complete. *Information and Computation*, 243:26–36, 2015.

[43] Alain Finkel, Bernard Willems, and Pierre Wolper. A direct symbolic approach to model checking pushdown systems. *Electronic Notes in Theoretical Computer Science*, 9:27–37, 1997.

[44] Marie Fortin, Anca Muscholl, and Igor Walukiewicz. Model-checking linear-time properties of parametrized asynchronous shared-memory pushdown systems. In *International Conference on Computer Aided Verification*, pages 155–175. Springer, 2017.

[45] Fei Gao, Frederic Mallet, Min Zhang, and Mingsong Chen. Modeling and verifying uncertainty-aware timing behaviors using parametric logical time constraint. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 376–381. IEEE, 2020.

[46] Noa Globerman and David Harel. Complexity results for two-way and multi-pebble automata and their logics. *Theoretical Computer Science*, 169(2):161–184, 1996.

[47] Stefan Göller, Christoph Haase, Joël Ouaknine, and James Worrell. Model Checking Succinct and Parametric One-Counter Automata. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 575–586. Springer, 2010. doi:10.1007/978-3-642-14162-1_48.

[48] Stefan Göller and Mathieu Hilaire. Reachability in two-parametric timed automata with one parameter is EXPSPACE-complete. In *38th International Symposium on Theoretical Aspects of Computer Science*, 2021.

[49] Stefan Göller and Markus Lohrey. Branching-time model checking of one-counter processes and timed automata. *SIAM J. Comput.*, 42(3):884–923, 2013. doi:10.1137/120876435.

[50] Stefan Göller, Richard Mayr, and Anthony Widjaja To. On the computational complexity of verifying one-counter processes. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 235–244. IEEE Computer Society, 2009. doi:10.1109/LICS.2009.37.

[51] Sheila A Greibach. Full afls and nested iterated substitution. *Information and Control*, 16(1):7–35, 1970.

[52] Christoph Haase. *On the complexity of model checking counter automata*. PhD thesis, Oxford University, 2012.

[53] Christoph Haase, Stephan Kreutzer, Joel Ouaknine, and James Worrell. Reachability in succinct and parametric one-counter automata. In *Proceedings of the CONCUR'09*, volume 5710 of *Lecture Notes in Computer Science*, pages 369–383. Springer, 2009.

[54] Christoph Haase, Joël Ouaknine, and James Worrell. On the relationship between reachability problems in timed and counter automata. In *International Workshop on Reachability Problems*, pages 54–65. Springer, 2012.

[55] Christoph Haase, Joël Ouaknine, and James Worrell. Relating reachability problems in timed and counter automata. *Fundam. Informaticae*, 143(3-4):317–338, 2016. doi:10.3233/FI-2016-1316.

[56] Matthew Hague. Parameterised pushdown systems with non-atomic writes. *arXiv preprint arXiv:1109.6264*, 2011.

[57] Michael A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, 1978.

[58] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. What good are digital clocks? In Werner Kuich, editor, *Automata, Languages and Programming, 19th International Colloquium, ICALP92, Vienna, Austria, July 13-17, 1992, Proceedings*, volume 623 of *Lecture Notes in Computer Science*, pages 545–558. Springer, 1992.

[59] Ulrich Hertrampf, Clemens Lautemann, Thomas Schwentick, Heribert Vollmer, and Klaus W. Wagner. On the power of polynomial time bit-reductions. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference*, pages 200–207. IEEE Computer Society Press, 1993.

[60] Markus Holzer. On emptiness and counting for alternating finite automata. In *Developments in Language Theory*, pages 88–97. World Scientific, 1995.

[61] Philipp Hönig, Rüdiger Lunde, and Florian Holzapfel. Formal verification of technical systems

using smartiflow and ctl. In *2nd International Conference on Applications in Information Technology (ICAIT-2016). Aizu-Wakamatsu, Japan*, 2016.

[62] John E Hopcroft and Jeffrey D Ullman. *Formal languages and their relation to automata*. Addison-Wesley Longman Publishing Co., Inc., 1969.

[63] Paul Hunter. Reachability in succinct one-counter games. *CoRR*, abs/1407.1996, 2014. URL: `http://arxiv.org/abs/1407.1996`, `arXiv:1407.1996`.

[64] O. H. Ibarra, T. Jiang, N. Trân, and H. Wang. New decidability results concerning two-way counter machines and applications. In *ICALP*, volume 700 of *LNCS*. Springer, 1993.

[65] Petr Jančar and Zdeněk Sawa. A note on emptiness for alternating finite automata with a one-letter alphabet. *Information Processing Letters*, 104(5):164–167, 2007.

[66] Vineet Kahlon. Parameterization as abstraction: A tractable approach to the dataflow analysis of concurrent programs. In *2008 23rd Annual IEEE Symposium on Logic in Computer Science*, pages 181–192. IEEE, 2008.

[67] Teodor Knapik, Damian Niwiński, and Paweł Urzyczyn. Higher-order pushdown trees are easy. In *International Conference on Foundations of Software Science and Computation Structures*, pages 205–222. Springer, 2002.

[68] S. R. Kosaraju. Decidability of reachability in vector addition systems. In *14th Annual Symposium on Theory of Computing*, pages 267–281. ACM Press, 1982.

[69] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical computer science*, 27(3):333–354, 1983.

[70] Orna Kupferman, Nir Piterman, and Moshe Y Vardi. From liveness to promptness. In *International Conference on Computer Aided Verification*, pages 406–419. Springer, 2007.

[71] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In *Research Report LSV-04-16*. LSV, ENS de Cachan, 2004.

[72] François Laroussinie, Nicolas Markey, and Ph Schnoebelen. Model checking timed automata with one or two clocks. In *International Conference on Concurrency Theory*, pages 387–401. Springer, 2004.

[73] Antonia Lechner. *Extensions of Presburger arithmetic and model checking one-counter automata*. PhD thesis, University of Oxford, 2016.

[74] Antonia Lechner, Richard Mayr, Joël Ouaknine, Amaury Pouly, and James Worrell. Model checking flat freeze LTL on one-counter automata. *Log. Methods Comput. Sci.*, 14(4), 2018. `doi:10.23638/LMCS-14(4:20)2018`.

[75] Antonia Lechner, Joël Ouaknine, and James Worrell. On the complexity of linear arithmetic with divisibility. In *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 667–676. IEEE, 2015.

[76] J. Leroux and G. Sutre. Flat counter automata almost everywhere! In *Proc. ATVA'05*, volume 3707 of *LNCS*. Springer, 2005.

[77] AN Maslov. Multilevel stack automata. *Problemy peredachi informatsii*, 12(1):55–62, 1976.

[78] E. W. Mayr. An algorithm for the general petri net reachability problem. In *Proc. STOC'81*, pages 238–246, New York, NY, USA, 1981. ACM.

[79] Marvin L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Annals of Mathematics. Second Series*, 74:437–455, 1961.

[80] Andrzej S Murawski, Steven J Ramsay, and Nikos Tzevelekos. Reachability in pushdown register automata. *Journal of Computer and System Sciences*, 87:58–83, 2017.

[81] Mohan Nair. On Chebyshev-type inequalities for primes. *The American Mathematical Monthly*, 89(2):126–129, 1982.

[82] Luyao Niu and Andrew Clark. Secure control under linear temporal logic constraints. In *2018 Annual American Control Conference (ACC)*, pages 3544–3551. IEEE, 2018.

[83] Joël Ouaknine and James Worrell. Universality and language inclusion for open and closed timed automata. In Oded Maler and Amir Pnueli, editors, *Hybrid Systems: Computation and Control, 6th International Workshop, HSCC 2003 Prague, Czech Republic, April 3-5, 2003, Proceedings*, volume 2623 of *Lecture Notes in Computer Science*, pages 375–388. Springer, 2003. doi:10.1007/3-540-36580-X_28.

[84] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[85] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (FOCS 1977)*, pages 46–57. IEEE, 1977.

[86] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in cesar. In *International Symposium on programming*, pages 337–351. Springer, 1982.

[87] Jean-Pierre Queille and Joseph Sifakis. Fairness and related properties in transition systems—a temporal logic to deal with fairness. *Acta informatica*, 19(3):195–220, 1983.

[88] Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical society*, 74(2):358–366, 1953.

[89] L. E. Rosier and H. C. Yen. A multiparameter analysis of the boundedness problem for vector addition systems. *Journal of Computer and System Sciences*, 32(1):105–135, 1986.

[90] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[91] Stefan Schwoon. *Model-checking pushdown systems*. PhD thesis, Technische Universität München, 2002.

[92] Dana Scott and Jacobus Willem de Bakker. A theory of programs. *Unpublished manuscript, IBM, Vienna*, 1969.

[93] Ryoma Senda, Yoshiaki Takata, and Hiroyuki Seki. Forward regularity preservation property of register pushdown systems. *IEICE TRANSACTIONS on Information and Systems*, 104(3):370–380, 2021.

[94] Ryoma Senda, Yoshiaki Takata, and Hiroyuki Seki. LTL model checking for register pushdown systems. *IEICE Transactions on Information and Systems*, 104(12):2131–2144, 2021.

[95] Olivier Serre. Parity games played on transition graphs of one-counter processes. In *International Conference on Foundations of Software Science and Computation Structures*, pages 337–351. Springer, 2006.

[96] A Prasad Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6(5):495–511, 1994.

[97] Ales Smrčka and Tomás Vojnar. Verifying parametrised hardware designs via counter automata. In *Haifa Verification Conference*, pages 51–68. Springer, 2007.

[98] Marius Stanica and Hervé Guéguen. Using timed automata for the verification of iec 61499 applications. *IFAC Proceedings Volumes*, 37(18):375–380, 2004.

[99] Larry J. Stockmeyer. *The complexity of decision problems in automata and logic.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1974.

[100] Anthony Widjaja To. Unary finite automata vs. arithmetic progressions. *Inf. Process. Lett.*, 109(17):1010–1014, 2009. doi:10.1016/j.ipl.2009.06.005.

[101] Alan Mathison Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society, Series 2*, 45:161–228, 1939.

[102] Leslie G Valiant and Michael S Paterson. Deterministic one-counter automata. *Journal of Computer and System Sciences*, 10(3):340–350, 1975.

[103] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the First IEEE Symposium on Logic in Computer Science*, pages 322–331, 1986.

[104] Igor Walukiewicz. Pushdown processes: Games and model checking. In *International Conference on Computer Aided Verification*, pages 62–74. Springer, 1996.

[105] Igor Walukiewicz. Pushdown processes: Games and model-checking. *Information and computation*, 164(2):234–263, 2001.

[106] Stefan Wöhrle. *Decision problems over infinite graphs: Higher-order pushdown systems and synchronized products.* Dissertation, RWTH Aachen, 2005.

[107] Gaoyan Xie, Zhe Dang, and Oscar H Ibarra. A solvable class of quadratic diophantine equations with applications to verification of infinite-state systems. In *International Colloquium on Automata, Languages, and Programming*, pages 668–680. Springer, 2003.

[108] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.