Parity games on bounded phase multi-pushdown systems *

M. F. Atig[1], A. Bouajjani[2], K. Narayan Kumar[3], and P. Saivasan[4]

[1] Uppsala University, Sweden / mohamed_faouzi.atig@it.uu.se
[2] LIAFA, Université Paris Diderot, France / abou@liafa.univ-paris-diderot.fr
[3] Chennai Mathematical Institute, India / kumar@cmi.ac.in
[4] TU Braunschweig, Germany / p.saivasan@tu-braunschweig.de

Abstract. In this paper we address the problem of solving parity games over the configuration graphs of bounded phase multi-pushdown systems. A non-elementary decision procedure was proposed for this problem by A. Seth. In this paper, we provide a simple and inductive construction to solve this problem. We also prove a non-elementary lower-bound, answering a question posed by A.Seth.

1 Introduction

Multithreaded programs are widely used in computer systems. They are notoriously complex and hard to get right. Therefore methods and tools for checking systematically their correctness are of paramount importance. Model checking is a well established algorithmic verification approach that allows to check automatically if a formal (automata-based) model of a program/system satisfies a property expressed in some specification logic, typically a temporal logic or a fixpoint calculus. The most expressive of these logics, when only regular properties are considered, is the propositional mu-calculus. It has been shown that the model checking problem of a given model against the proposition mu-calculus is tightly related to the problem of solving 2-player parity games on the state graph of the model. Solving games means to determine if one of the player has a winning strategy. This problem is decidable for finite-state systems, and even for classes of infinite-state models such as pushdown systems [12]. The latter are known to be natural formal models for sequential programs with recursive procedure calls. In this paper, we investigate the extension of the game-theoretic verification framework to the case of multi-threaded programs.

Natural models for multi-threaded (shared memory concurrent) programs are multi pushdown systems, i.e., several pushdown systems that can access to a shared (finite) memory. However, this model is clearly Turing powerful, and therefore, any nontrivial problem stated on this model is obviously undecidable. Then, one possibility to obtain decidability is to consider restrictions on the

* The authors acknowledge partial support by Indo-French Project AVeCSo, Indo-Swedish DST-VR Project P-02/2014, Infosys Foundation.

kind of behaviours for which the decision problem is stated. This can be useful in the context of finding errors. Indeed, good under-approximations of the set of behaviours are useful to orient the search toward some special classes of computations where most of the errors are visible. To this aim, parameterized under-approximation schema have been proposed and shown to be useful and efficient for the analysis of muti-threaded programs such as context-bounding in [5] and phase-bounding in [10]. Context-bounding consists in bounding the number of context-switches between threads, while phase-bounding is a more liberal concept where each phase corresponds to a sequence of operations where all pops are from one fixed stack, while pushes are non restricted and allowed to be on any stack. In fact, phase-bounding is more general than context-bounding in the sense that sets of behaviours explored under phase-bounding, for some fixed bound on the number of phases, would require an unbounded number of context-switches to be explored. Then, the issue we address in this paper is exploring the limits of the decidability of the problem of solving parity games on muti-pushdown systems under phase bounding, and establishing its complexity.

Anil Seth provides in [8] a decision procedure for solving the parity games on phase bounded muti-pushdown systems for a fixed initial state. The procedure has a non-elementary complexity (i.e., a tower of exponentials with a height depending on the number of phases). His result is based on an extension of Walukiewicz's proof for solving this problem in the case of pushdown systems [12]. This is proof is very difficult, based on reducing the problem of solving a parity game on a phase bounded muti-pushdown system to the problem of solving the parity game on a complex finite state graph. An important question is whether it is possible to provide a conceptually simpler proof allowing to have a better understanding of the structure of the problem. Also, a natural question is whether the high, non-elementary complexity is unavoidable.

The first contribution of this paper is to provide a proof that is based on a simple inductive argument on the number of phases by effectively reducing a $(k+1)$-phase game to a k-phase game, for any $k \geq 1$. Our proof exploits the global approach used in, e.g., [1, 3, 7] to construct the set of winning states in a parity game on pushdown systems. Roughly, the latter construction is used to construct the winning states in last phase, and the obtained set is plugged in the system to get a new one with one less phase, and so on.

The second contribution of the paper is to establish a non-elementary lower bound for the parity games on bounded-phase multi-pushdown systems, showing that this problem is inherently hard and that our construction is optimal for computing the set of winning states. The proof is based on a reduction of the satisfiability problem of first-order logic over natural numbers with ordering. The details missing in the paper can be found in [6]

2 Bounded phase multi pushdown systems

Multi pushdown systems (MPDS) are generalizations of pushdown systems with multiple stacks. The kinds of transitions performed by an MPDS are (i) pushing

a symbol into one of the stacks (*ii*) popping a symbol from one of the stacks and (*iii*) an internal move that changes the state but leaves the stacks unchanged.

Definition 1 (MPDS). *A* Multi-PushDown System *(MPDS) is a tuple* $\mathrm{M} = (n, Q, \Gamma, \Delta, q_0)$ *where* $n \geq 1$ *is the number of stacks,* Q *is the non-empty set of states,* Γ *is the finite set of stack symbols, containing a special symbol* $\perp$, $q_0 \in Q$ *is the initial state and* $\Delta \subseteq Q \times \mathbf{Op} \times Q$ *is the transition relation, where* $\mathbf{Op} = \bigcup_{i \in [1..n]} \mathbf{Op}_i \cup \{\mathbf{Int}\}$ *and* $\mathbf{Op}_i = \{\mathbf{Push}_i(a), \mathbf{Pop}_i(a) \mid a \in \Gamma \setminus \{\perp\}\} \cup \{\mathbf{Zero}_i\}$.

A configuration of the MPDS M is a (n+1) tuple $(q, \gamma_1, \gamma_2, \cdots, \gamma_n)$ with $q \in Q$, and $\gamma_1, \gamma_2, \ldots, \gamma_n \in (\Gamma \setminus \perp)^* \perp$. The set of all configurations of the MPDS M is denoted by $\mathcal{C}(M)$. The initial configuration c_M^{init} of the MPDS M is $(q_0, \perp, \ldots, \perp, \perp)$. Given $\tau = (q, op, q') \in \Delta$ and two configurations $c = (q, \gamma_1, \cdots, \gamma_n) \in \mathcal{C}(M)$ and $c' = (q', \gamma_1', \cdots, \gamma_n') \in \mathcal{C}(M)$, we say $c \xrightarrow{\tau} c'$ iff one of the following holds.

- $\tau = (q, \mathbf{Push}_i(a), q')$, $\gamma_i' = a.\gamma_i$ and $\forall j \in [1..n] \setminus \{i\}$, $\gamma_j' = \gamma_j$
- $\tau = (q, \mathbf{Pop}_i(a), q')$, $\gamma_i = a.\gamma_i'$ and $\forall j \in [1..n] \setminus \{i\}$, $\gamma_j' = \gamma_j$
- $\tau = (q, \mathbf{Zero}_i, q')$, $\gamma_i' = \gamma_i = \perp$ and $\forall j \in [1..n] \setminus \{i\}$, $\gamma_j' = \gamma_j$
- $\tau = (q, \mathbf{Int}, q')$ and $\forall j \in [1..n]$, $\gamma_j' = \gamma_j$

A *computation* π of M starting from a configuration c is a (possibly infinite) sequence of the form $c_0 \xrightarrow{\tau_1} c_1 \xrightarrow{\tau_2} \cdots$ such that $c_0 = c$ and $c_{i-1} \xrightarrow{\tau_i} c_i$ for all $1 \leq i \leq |\tau_1 \tau_2 \cdots|$. Given a finite computation $\pi_1 = c_0 \xrightarrow{\tau_1} c_1 \xrightarrow{\tau_2} c_2 \cdots \xrightarrow{\tau_m} c_m$ and a (possibly infinite) computation $\pi_2 = c_{m+1} \xrightarrow{\tau_{m+2}} c_{m+2} \xrightarrow{\tau_{m+3}} \cdots$, π_1 and π_2 are said to be *compatible* if $c_m = c_{m+1}$. Then, we write $\pi_1 \bullet \pi_2$ to denote the computation $\pi \stackrel{\text{def}}{=} c_0 \xrightarrow{\tau_1} c_1 \xrightarrow{\tau_2} c_2 \cdots \xrightarrow{\tau_m} c_m \xrightarrow{\tau_{m+2}} c_{m+2} \xrightarrow{\tau_{m+3}} \cdots$. Given a configuration $c = (q, w_1, w_2, \cdots, w_n)$, we will use $Stack_i(c)$ to denote the stack-i content i.e. w_i and $State(c)$ to denote the state q.

It is easy to see that a multi-pushdown system with just one stack is a pushdown system. When refering to pushdown systems, we will omit any references to the stack number.

2.1 Bounded phase

The bounded phase restriction on an MPDS was introduced in [11]. Informally a phase is a sequence of operations in which the **Pop** and **Zero** operations are performed on only one stack. In a *bounded-phase* computation, there is an a-priori bound on the number of phases that it can involve.

Definition 2. Phase: *A* Phase *of a stack* $i \in [1..n]$ *is a computation involving pops and zero tests only from stack-i i.e. it is a computation of the form* $\pi = c_0 \xrightarrow{\tau_1} c_1 \xrightarrow{\tau_2} \cdots$ *in which* $\tau_1, \tau_2, \ldots \in \Delta^{\downarrow_i}$, *where* $\Delta^{\downarrow_i} = \Delta \cap (Q \times (\mathbf{Op} \setminus \bigcup_{j \neq i} \cup_{a \in \Gamma} \{\mathbf{Pop}_j(a)\} \cup \{\mathbf{Zero}_j\}) \times Q)$.

Bounded Phase computation: *Given* $k \in \mathbb{N}$, *a computation* $\pi = c_0 \xrightarrow{\tau_1} c_1 \xrightarrow{\tau_2} \cdots$ *is said to be* k phase-bounded *if it can be seen as a concatenation of at most* k*-phases i.e.* $\pi = \pi_1 \bullet \pi_2 \bullet \ldots \bullet \pi_l$ *such that* $\pi_1, \cdots, \pi_l$ *are phases and* $l \leq k$.

3 Parity games

Parity game is a two player game that is played on a directed graph (possibly infinite). Informally the game can be thought of as one that starts from a designated node in which a token is placed. Each of the nodes in the graph are owned by one of the two players. Further every node in the graph is assigned a number from a predetermined finite set of natural numbers, we will refer to this number as the rank of the node. The game proceeds in rounds. In each round, the player who owns the node in which token is placed makes a move. We will assume that the graph has no dead ends and that a player can always make a move. During a move, a player removes the token from a node and places it on one of the adjacent nodes. The winner of the game is determined by the minimum rank visited infinitely often in the play. The game is formalised below.

Definition 3. *Parity game is defined over game graph* $\mathcal{G} = (V, E, \tau, \sigma)$ *where* V *is (possibly infinite) set of nodes,* $E \subseteq V \times V$ *is set of edges,* $\tau : V \mapsto [0, 1]$ *is a function that defines ownership of the node and* $\sigma : V \mapsto [1..m]$ *for some* $m \in \mathbb{N}$ *is a ranking function that assigns a rank to each node.*

For any node $s \in V$*, we define* $E(s) = \{s' \mid (s, s') \in E\}$*. We say a* π *is a finite play of* $\mathcal{G}$ *iff* $\pi = s_1 s_2 \cdots s_n$ *such that for all* $i \in [1 \ldots n-1]$*,* $(s_i, s_{i+1}) \in E$ *and* $E(s_n) = \emptyset$*.* π *is said to be infinite play of* $\mathcal{G}$ *iff* $\pi = s_1 s_1 s_2 \cdots$ *such that for all* $i \in \mathbb{N}$*, we have* $(s_i, s_{i+1}) \in E$*. We will assume w.l.o.g. that graphs we deal with do not have any dead end nodes and hence assume that all our plays are infinite. For any infinite play* $\pi = s_0 s_1 s_2 \cdots$*, we let* S^{∞}_{π} *to be the set of all nodes that appear infinitely often in the play* π*. We define* $Parity(\pi) = min(inf(\pi))$ *mod* 2*, where* $inf(\pi) = \{\sigma(s) \mid s \in S^{\infty}_{\pi}\}$ *i.e. it is the parity of the minimum rank that is seen infinitely often along the run. An infinite play* π *is winning for player-*0 *iff* $Parity(\pi)$ *is* 0*, otherwise it is winning for player-*1*.*

For any $i \in [0, 1]$, we will let $V_i = \{s \mid s \in V \wedge \tau(s) = i\}$ i.e. it is the set of positions owned by player-i. A strategy function f for player-0 is defined as $f : V^*V_0 \mapsto 2^V \setminus \emptyset$. An infinite play $\pi = v_0 v_1 v_2 \cdots$ is said to be confirming to a strategy function f iff for any prefix of the play $\pi' = v_0 \cdots v_i \in V^*V_0$, $v_{i+1} \in f(\pi')$. A strategy function f is said to be winning for player-0 from any node s, if the set of all possible plays π which start from the node s and confirms to the strategy function f are winning for player-0. The strategy function for player-1 is defined analogously. We say a node s is winning for player-0 (or player-1) iff there is a strategy function that is winning for player-0 (or player-1) from that position. A strategy function f of player-i is called memory less strategy or positional strategy if it is of the form $f : V_i \mapsto 2^V \setminus \emptyset$, i.e. it only depends on a single node. Any given play $\pi = v_0 v_1 \cdots$ is said to be confirming to the memoryless strategy function f of player-i, if for all nodes $v_j \in V_i$ ($j \in \mathbb{N}$), we have $v_{j+1} \in f(v_j)$.

A natural question in this setting is whether for any position s, one of the two players has a winning strategy (determinacy) from that position and if so whether the strategy is memoryless (memoryless determinacy). The determinacy

of parity games follows from a very general result due to Martin's determinacy theorem [4] which establishes the determinacy for a much wider class of games. Memory less determinacy theorem (Theorem-1) for parity games [2] establishes that we not only have determinacy, we also have that the winning player has a memoryless winning strategy.

Theorem 1. *[2] Given a parity game $G = (V, E, \tau, \sigma)$, there is a partition of nodes V, $V = W_0 \uplus W_1$ and memoryless strategy functions σ_0 and σ_1 such that σ_i is winning for player-i from each positions in W_i.*

Determining the winning sets and strategies in such games is an interesting problem, which is easy to solve for finite games and not so for infinite games.

4 Bounded phase parity games on MPDS

In this paper, we are interested in parity games played over the configuration graphs of a MPDS with bounded-phase restriction (also refered to as the *bounded phase parity games*). For purpose of defining the bounded-phase parity games, we will first enhance the configurations of a multi-pushdown system with the information about the number of phases remaining and the identity of the currently active stack.

Definition 4 (Bounded-phase parity games). *Given a multi-pushdown system $M = (n, Q, \Gamma, \Delta, q_0)$ and a constant k, we define the set of enhanced configurations of M, $\mathcal{E}^k(M)$ as $\mathcal{C} \times [0..n] \times [1..k]$. Such an enhanced configuration, apart from containing the configuration of multi-pushdown system, also records the currently active stack and number of remaining phases. We will omit the k and simply refer to it as $\mathcal{E}(M)$ when ever k is clear from the context. At the beginning of any computation, we let the current stack component (the penultimate component) of $\mathcal{E}(M)$ to be 0, indicating that none of the stacks are active. From such a position, a stack gets active on the very first pop or zero test. Given any two configurations $(c, i, j), (c', i', j') \in \mathcal{E}(M)$, we say $(c, i, j) \rightsquigarrow (c', i', j')$ iff $c \xrightarrow{\tau} c'$ and one of the following holds.*

- *If $\tau = (q, \mathbf{Pop}_l, q')$ or $\tau = (q, \mathbf{Zero}_l, q')$ for some $l \in [1..n]$ and $i = 0$ then $j = j' = k$ and $i' = l$.*
- *if $\tau = (q, \mathbf{Push}_k(a), q')$ for some $k \in [1 \dots n]$ or $\tau = (q, \mathbf{Int}, q')$ or $\tau = (q, \mathbf{Pop}_i, q')$ or $\tau = (q, \mathbf{Zero}_i, q')$ then $i' = i, j' = j$*
- *if $\tau = (q, \mathbf{Pop}_l(a), q')$ or $\tau = (q, \mathbf{Zero}_l, q')$ for some $l \neq i$ and $j > 1$ then $i' = l, j' = j - 1$*

Let $\tau : Q \mapsto [0, 1]$ be a map that designates each of the states to a player, let $\sigma : Q \mapsto [1..m]$ be a map that assigns a rank to each of the states and let k be any natural number, then a k-bounded-phase parity game *is the parity game played on the game graph $\mathcal{G} = (\mathcal{E}(M), \rightsquigarrow, \tau, \sigma)$, where τ, σ are extended to configurations as follows. For any $(c, i, j) \in \mathcal{E}(M)$, we let $\tau((c, i, j)) = \tau(\text{State}(c))$ and $\sigma((c, i, j)) = \sigma(\text{State}(c))$. We will refer to such games as $\mathcal{G} = (k, M, \tau, \sigma)$.*

Given a bounded phase parity game $\mathcal{G} = (k, M, \tau, \sigma)$ and a node $s \in \mathcal{E}(M)$, in this paper we are interested in the problem of determining whether there is a strategy function g that is winning for player-0 from the node s.

5 Some results on parity games

In this section, we will prove/recall some lemmas that we will use later. The following lemma states that if there is a mapping from one game graph to another such that any move in the former can be simulated in the latter, and if such a simulation preserves the player and the rank at each position of the play, then the winning positions are also preserved by the mapping.

Lemma 1. *Let $G = (V_G, E_G, \tau_G, \sigma_G)$ and $H = (V_H, E_H, \tau_H, \sigma_H)$ be games graphs and let $\mathbf{F} : V_G \longrightarrow V_H$ be any function such that for any position $x \in V_G$*

1. *$\sigma_G(x) = \sigma_H(\mathbf{F}(x))$, the function is rank preserving.*
2. *$\tau_G(x) = \tau_H(\mathbf{F}(x))$ i.e. x and $\mathbf{F}(x)$ belongs to the same player i.*
3. *If $x \to x'$ then $\mathbf{F}(x) \to \mathbf{F}(x')$.*
4. *If $\mathbf{F}(x) \to y$ then there exists x' such that $x \to x'$ and $\mathbf{F}(x') = y$.*

Then, any position x is winning for player 0 (player-1) in G if and only if $\mathbf{F}(x)$ is winning for player 0 (resp player-1) in H.

Given a game graph $\mathcal{G} = (V, E, \tau, \sigma)$, $U \subseteq V$ is said to be a trap of $\mathcal{G}$ iff $E \cap U \times (V \setminus U) = \emptyset$. i.e. once the game enters U, there is no way for it to exit. The following Lemma states that given any parity game graph, the game graph obtained by fusing all the winning positions of player-0 (and that of player-1), into one node, preserves the winning positions.

Lemma 2. *Let $G = (V_G, E_G, \tau_G, \sigma_G)$ be a parity game and let $V_H \subseteq V_G$ be a trap of G. Suppose V_{H0} and V_{H1} are the winning positions for the players 0 and 1 respectively, in the subgame V_H. Then, consider the game graph $G' = (V_{G'}, E_{G'}, \tau_{G'}, \sigma_{G'})$ constructed as follows:*

1. *Delete the subgame V_H, add two new positions q_w and q_l and add edges from q_w to q_w and q_l to q_l.*
2. *For $s \to t$ in E with $s \notin V_H$ and $t \in V_{H0}$, add an edge from s to q_w.*
3. *For $s \to t$ in E with $s \notin V_H$ and $t \in V_{H1}$, add an edge from s to q_l.*
4. *For all $v \in V_G \setminus V_H$, we let $\tau_{G'}(v) = \tau_G(v)$, $\tau_{G'}(q_w) = 0$, $\tau_{G'}(q_l) = 1$.*
5. *For all $v \in V_G \setminus V_H$, $\sigma_{G'}(v) = \sigma(v)$ and $\sigma_{G'}(q_w) = 0$, $\sigma_{G'}(q_l) = 1$.*

Then, any position in V_G that is not in V_H is winning for any player in G if and only if it is winning for that player in the game G'.

In [1, 7], T. Cachat and O. Serre independently proved that the set of all winning positions of a particular player in a parity game played on a pushdown system is effectively regular. This can also be obtained using tree automata techniques as shown in [3].

Definition 5 (Parity game on PDS). *Given a pushdown system* $P = (Q, \Gamma, \Delta, q_0)$ *and mappings* $\tau : Q \mapsto [0,1]$ *and* $\sigma : Q \mapsto [1..m]$*, parity game on PDS is simply a parity game played on the game graph* $\mathcal{G} = (\mathcal{C}(P), \rightarrow, \tau, \sigma)$*, where* τ *and* σ *are extended to configurations as follows. For any configuration of the form* $c = (q, \gamma)$*,* $\tau(c) = \tau(q)$ *and* $\sigma(c) = \sigma(q)$*.*

Theorem 2. *[1, 7] The set of all winning positions of player* 0 *(or player* 1*) in a pushdown game can be effectively characterised by an exponential sized finite state automaton over the alphabet* $\Gamma \cup Q$*. Such an automaton accepts a word* $wq \in \Gamma^* Q$ *if and only if the configuration* (q, w) *is winning for player* 0 *(or player* 1*).*

6 Decidability of bounded phase parity games

Theorem 3. *Given a* k *bounded phase parity game, deciding whether player* 0 *can win from the initial configuration can be done in time which is* NON-ELEMENTARY *in the number of phases.*

In this section we prove the theorem-3 which states that the winner of a bounded phase parity game can be decided in non-elementary time. The proof of the theorem is obtained by inductively solving the k bounded-phase parity game. The intuitive idea is to first show that if the game is a single phase game, then the game graph of such a game actually corresponds to just the positions of a pushdown game and by Theorem 2 we know the set of winning positions are recognisable. Secondly observe that that the positions in the game graph are stratified in the following sense – if (c', i', k') is reachable from (c, i, k) then $k' \leq k$. From this, we know that if the game were to enter the last phase, it will continue to remain in that phase. Hence any position in the last phase corresponds to a position of a pushdown game, which is known to be recognisable. Using this information, we will go onto show how to reduce the k bounded-phase game to a $k-1$ bounded-phase game.

6.1 Decidability of a 1-phase game

In an 1-phase game, the configurations can be of the form $(c, i, 1)$ with $i \neq 0$ or of the form $(c, 0, 1)$. We will show in each of the cases that the set of positions winning for player-0 is a recognisable set (i.e. it can be effectively determined). For the sub-game involving only configurations of the form $(c, i, 1)$, we will show that such positions correspond to positions of a pushdown game. Now using the fact that the set of all positions winning for player-0 in a pushdown game is a recognisable set, we will show that the nodes that are winning for player-0 in sub-game involving configurations of the form $(c, i, 1)$ is also a recognisable set. For the case involving configurations of the form $(c, 0, 1)$, we will reduce such a sub-game to a parity game involving only finitely many states.

Lemma 3. *Let $\mathcal{G} = (1, M, \tau, \sigma)$ be a bounded-phase parity game, with $M = (n, Q, \Gamma, \Delta, q_0)$. We can effectively determine the set of all positions of the form $\mathcal{E}^1(M)$, that are winning for player 0. Further the size of such an automaton that recognises the winning positions is at most exponential.*

Proof. The nodes in $\mathcal{E}^1(M)$ are either of the form $(c, 0, 1)$ or of the form $(c, i, 1)$ for some $i \neq 0$. We will first consider the nodes of the form $(c, i, 1)$ and show that the winner can be determined. The general idea of the proof is to first construct a pushdown system for each $i \in [1..n]$, from the given multi-pushdown system M. Such a pushdown system will simulate the moves of stack-i by using its own stack for any operations on stack i, and ignoring the pushes on other stacks. The pushdown system (corresponding to stack-i) is defined as, $P_i = (Q, \Gamma, \delta_i, q_0)$, where δ_i is defined as

- For every $\tau = (q, \mathbf{Pop}_i(a), q') \in \Delta$, we add $\tau' = (q, \mathbf{Pop}(a), q') \in \delta_i$. We add similar transitions for $\tau = (q, \mathbf{Zero}_i(a), q') \in \Delta$, $\tau = (q, \mathbf{Push}_i(b), q') \in \Delta$ and $\tau = (q, \mathbf{Int}, q') \in \Delta$
- For $j \neq i$ and for every $\tau = (q, \mathbf{Push}_j(b), q') \in \Delta$, we add $\tau' = (q, \mathbf{Int}, q') \in \delta_i$.

The winning positions of each player of the sub-game with configurations of the form $(c, i, 1)$ with $i \neq 0$, can be captured using the pushdown game $\mathcal{H} = (\mathcal{C}(P_i), \rightarrow, \tau, \sigma)$. Let the function $\mathbf{F} : \mathcal{E}^1(M) \mapsto \mathcal{C}(P_i)$ be given by $\mathbf{F}(((q, \gamma_1, \gamma_2, \cdots, \gamma_n), i, 1)) = ((q, \gamma_i))$. The function $\mathbf{F}$ simply disregards content of stacks other than i and keeps stack i intact. Following lemma shows that such a mapping will preserve the properties required by Lemma 1.

Lemma 4. *The mapping $\mathbf{F}$ preserves the following properties. For any $v \in \mathcal{E}^1(M)$, we have $\tau(v) = \tau(\mathbf{F}(v))$ and $\sigma(v) = \sigma(\mathbf{F}(v))$. For any $u = (c, i, 1)$, $v = (c', i, 1) \in \mathcal{E}(M)$, if $(u \rightsquigarrow v)$ then we have $\mathbf{F}(u) \rightarrow \mathbf{F}(v)$. Suppose for some $v \in \mathcal{E}(M)$, we have $\mathbf{F}(v) \rightarrow d$, then there is an $u \in \mathcal{E}(M)$ such that $\mathbf{F}(u) = d$ and $v \rightsquigarrow u$*

Thus using Lemma 1, the position $(c, i, 1)$ in our subgame is winning for a player-i if and only if $\mathbf{F}((c, i, 1))$ is winning for player-i in the pushdown game $(\mathcal{C}(P_i), \rightarrow, \tau, \sigma)$. Thus, the set of all winning positions of a 1-phase game involving stack-i is given by $\mathcal{S} = \{(c, i, 1) \mid \mathbf{F}((c, i, 1)) \in \mathcal{R}_{P_i}\}$ where $\mathcal{R}_{P_i}$ is the set of winning positions in the game $(\mathcal{C}(P_i), \rightarrow, \tau, \sigma)$. It is easy to see that $\mathcal{S}$ is recognisable set since $\mathcal{R}_{P_i}$ is recognisable by Theorem 2.

Finally we consider the positions of the form $(c, 0, 1)$. Any configuration (c', i, k') reached from configuration $(c, 0, 1)$ must necessarily have $k' = 1$. Further, if the game ever enters a position with $i \neq 0$, we may immediately determine the winner of the game from thereon (Since we already know how to compute the set of all winning positions of a 1-phase game involving stack-i). This allows us to formulate a finite state game whose solution determines the winning positions of the form $(c, 0, 1)$. Note that the game can remain in a position of the form $(c, 0, 1)$ iff the transitions involve only push moves or internal moves. The moment

a pop move is made, the stack is fixed and the game enters a configuration of the form $(c, i, 1)$, for some $i \in [1..n]$.

Let $B_i = (Q_{B_i}, \Gamma \cup Q, s_i, \delta^{B_i}, F_i)$ be the deterministic finite state automaton that accepts a word of the form $\perp w^R q$ (where $(q, w\perp)$ is a configuration of the pushdown system P_i) iff it belongs to the winning positions of the game $(\mathcal{C}(P_i), \rightarrow, \tau, \sigma)$. Such an automata is guaranteed by Theorem 2, we note that the size of such an automata is exponential in the size of the pushdown system. The finite state game we have in mind is one which instead of keeping track of the contents of each stack i, only keeps track of the top of stack symbol and the state reached by B_i on reading the contents of that stack. We plan to do this only for the push and the internal moves and hence it is indeed feasible. Any pop or zero test moves would commit to a stack. In this case we may immediately determine the winner using the state of B_i.

The state space of the finite state game H is $(Q \times \Gamma^n \times Q_{B_1} \times Q_{B_2} \cdots Q_{B_n}) \cup \{q_w, q_l\}$, we will refer to this as $V(H)$. The state q_w is entered on determining that the game will be won by player 0 and q_l if it is determined that the game will be lost by player 0. The edges $\rightarrow_H$ of the game graph are given as follows:

1. $q_w \rightarrow q_w$ and $q_l \rightarrow q_l$
2. For all $i \in [1..n]$, we have if $(q, \mathbf{Push}_i(b), q') \in \Delta$, then we have $(q, a_1, \cdots, a_n, p_1, \cdots p_n) \rightarrow (q', a_1, \cdots, b, \cdots, a_n, p_1, \cdots, \delta^{B_i}(p_i, a_i), \cdots, p_n)$, for all $a_1, a_2, \cdots, a_n \in \Gamma$ and for all $j \in [1..n], p_j \in Q_{B_j}$.
3. If $(q, \mathbf{Int}, q') \in \Delta$ then we have $(q, a_1, \cdots, a_n, p_1, \cdots p_n) \rightarrow (q', a_1, \cdots, a_n, p_1, \cdots, p_n)$. This handles the case of internal moves.
4. If $(q, \mathbf{Pop}_i(a_i), q') \in \Delta$ then if $\delta^{B_i}(p_i, q') \in F_i$, we have $(q, a_1, \cdots, a_n, p_1, \cdots, p_n) \rightarrow q_w$ else if $\delta^{B_i}(p_i, q') \notin F_i$, we have $(q, a_1, \cdots, a_n, p_1, \cdots, p_n) \rightarrow q_l$
5. If $(q, \mathbf{Zero}_i, q') \in \Delta$ then, if $\delta^{B_i}(s_i, \perp.q') \in F_i$, we have $(q, a_1, \cdots, a_{i-1}, \perp, a_{i+1} \cdots, a_n, p_1, \cdots, p_{i-1}, s_i, p_{i+1}, p_n) \rightarrow q_w$ else if $\delta^{B_i}(s_i, \perp.q') \notin F_i$, we have $(q, a_1, \cdots, a_{i-1}, \perp, a_{i+1} \cdots, a_n, p_1, \cdots, p_{i-1}, s_i, p_{i+1}, \cdots, p_n) \rightarrow q_l$

Now consider the ranking function σ' that assigns 0 to q_w, 1 to q_l, i.e. $\sigma(q_w) = 1$ and $\sigma(q_l) = 0$ and for all other positions of the form $c = (q, a_1, \cdots, a_n, p_1, \cdots, p_n)$, we let $\sigma'(c) = \sigma(q)$. Similarly, consider τ' that assigns $\tau'(q_w) = 0$ and $\tau'(q_l) = 1$. Further we let $\tau'(c) = \tau(q)$ for any $c = (q, a_1, \cdots, a_n, p_1, \cdots, p_n)$. We claim that nodes in the subgame involving configurations of the form $(c, 0, 1)$ can be reduced to the finite state parity game given by $H = (V(H), \rightarrow_H, \sigma', \tau')$.

The idea now is to provide a mapping from positions of the form $(c, 0, 1)$ in G to positions in H. For this, we wish to first eliminate from G, using Lemma 2, any positions of the form $(c, i, 1)$ for $i \neq 0$. Note that, the set of all position $S = \{(c, i, 1) \mid (c, i, 1) \in \mathcal{E}(M), i \neq 0\}$ is a trap in G. Let $W_i \subseteq S$ be the set of winning positions for player-i. Now consider the game graph G' obtained by deleting S from G, adding two new vertices p_{win}, p_{lose} replacing all the edges to W_0 by edges to p_{win} and the edges to W_1 by edges to p_{lose}. Then by application of Lemma 2, a position in $\mathcal{E}(M) \setminus S$ is winning for any player iff it is winning in G'. Observe that the set $\mathcal{E}(M) \setminus S$ is exactly $\{(c, 0, 1) | c \in \mathcal{C}(M)\}$. Now consider the mapping $\mathbf{F}$ from positions of G' to positions in H defined as $\mathbf{F}((q, a_1\gamma_1, a_2\gamma_2, \cdots, a_n\gamma_n), 0, 1)) = ((q, a_1, \cdots, a_l, \delta_1^P(x_1, \gamma_1^R), \delta_2^P(x_2, \gamma_2^R), \cdots, \delta_n^P(x_n, \gamma_n^R))$ and $\mathbf{F}(p_{\text{win}}) = q_w$

and $\mathbf{F}(p_{\text{lose}}) = q_l$. Notice that such a mapping preserves the properties required by lemma-1. As a result, we get the following lemma.

Lemma 5. *A position* $(c, 0, 1)$ *is winning for player-i in* G' *if and only if* $\mathbf{F}((c, 0, 1))$ *is winning for player-i in* H.

In addition note that the set of positions of the form $(c, 0, 1)$ that are winning for player-0 are precisely those in $S_{\text{Win}} = \{w \mid f(w) \text{ is winning for player-0}\}$ and this clearly is a recognizable set. This completes the proof of Lemma 3.

6.2 Decidability of a *k* phase game

The idea is to use the fact that the 1-phase sub-game of a k-phase game is determined. Notice that after execution of $k-1$ phases, what remains is a 1-phase sub-game. In this 1-phase sub-game, the stack contents of all other stacks (exclusing the currently active stack) are irrelevant and hence it can easily be simulated by a pushdown automata.

Let $\mathcal{K} = \{(c, i, 1) \mid (c, i, 1) \in \mathcal{E}^k \wedge i \in [1..n]\}$. Recall the pushdown automata P_i constructed in Lemma 3. As in the case of Lemma 3, we can provide a mapping $\mathbf{F}$ from the sub-game involving positions from $\mathcal{K}$ to positions in the game $\mathcal{H} = (\mathcal{C}(P_i), \rightarrow, \tau, \sigma)$, such that $\mathbf{F}$ satisfies the properties of Lemma 1 (as a matter of fact, the game graph $\mathcal{H}$ is isomorphic to the trap consisting of positions of the form $(c, i, 1), i \neq 0$ in the game graph of a 1-phase parity game). From this, we get the following Lemma which states that the set of winning positions of a 1-phase sub-game can be effectively determined using the set of winning positions of the pushdown system P_i.

Lemma 6. $s \in \mathcal{K}$ *is winning for player-*0 *iff* $\mathbf{F}(s)$ *is winning for player-*0 *in the pushdown game* $\mathcal{H} = (\mathcal{C}(P_i), \rightarrow, \tau, \sigma)$

Now to handle the case of k-phase game, we first invoke Theorem-2 to obtain $B_i = (Q_{B_i}, \Gamma \cup Q, s_i, \delta^{B_i}, F_i)$ that recognises the winning positions of the pushdown system P_i. Suppose at the end of $k-1$ phase, we know the state that the automata B_i reaches on reading stack i, then, at the beginning of phase k, we can determine whether player-0 is winning from that position or not. The case for 1-phase game was easy since we had only pushes to contend with (and hence it was possible to simulate B_i using only the state space). However, in case of a $k-1$ phase game, we need to also handle pop operations. Hence it is not possible to simulate B_i automata by just keeping it in the state space. The informal idea is to keep the B_i automata as part of the state space and simulate it on each push onto the stack-i. In addition, on each push, along with the stack symbol we also store in the stack the state of B_i that was reached before the current push. Now each time a pop operation is performed, we can retrieve the correct state of the B_i automata and delegate it to the state space. The details are formalised below.

Let (k, M, τ, σ) be a k-bounded-phase game with $M = (n, Q, \Gamma, \Delta, q_0)$ and $k > 1$. We define a new MPDS as $M(k) = (n, Q_{M(k)}, \Gamma_{M(k)}, \Delta', q_0^{M(k)})$, where

$Q_{M(k)} = Q \times Q_{B_1} \times \cdots \times Q_{B_n} \times \Gamma^n \times [0..n] \times [2..k] \cup \{q_w, q_l\}$, $\Gamma_{M(k)} = \bigcup_{i \in [1..n]} (\Gamma \times Q_{B_i}) \cup \{\bot\}$, $q_0^{M(k)} = (q_0, s_1, \cdots, s_n, \bot^n, 0, k)$ and the transition relation Δ' is defined as follows

1. if $(q, \mathbf{Push}_i(b), q') \in \Delta$ then we have for all $i \in [1..n]$, $p_i \in Q_{B_i}$, $m \in [0..n]$, $l \in [2..k]$ and $a_i \in \Gamma$, $((q, p_1, \cdots, p_n, a_1, \cdots, a_n, m, l)$, $\mathbf{Push}_i(a_i, p_i)$, $(q', p_1, \cdots, p_{i-1}, \delta_i^P(p_i, a_i), \cdots, p_n, a_1, \cdots, a_{i-1}, b, a_{i+1}, \cdots, a_n, m, l)) \in \Delta'$. We always store the top of stack and current state of B_i in the state space. Every time we push, the previously stored top of stack in the state (a_i) and the previously stored state of B_i (p_i) is pushed into the actual stack.
2. if $(q, \mathbf{Int}, q') \in \Delta$ then we have for all $i \in [1..n]$, $p_i \in Q_{B_i}$ and $a_i \in \Gamma$, $((q, p_1, \cdots, p_n, a_1, \cdots, a_n, m, l)$, $\mathbf{Int}_i, (q', p_1, \cdots, p_n, a_1, \cdots, a_n, m, l)) \in \Delta'$.
3. For each $(q, \mathbf{Pop}_j(a_j), q') \in \Delta$ we add the following transitions.
   - $((q, p_1, \cdots, p_n, a_1, \cdots, a_n, 0, k), \mathbf{Pop}_j(b_j, p'_j), (q', p_1, \cdots, p_{j-1}, p'_j, p_{j+1}, \cdots, p_n, a_1, \cdots, a_{j-1}, b_j, a_{j+1}, \cdots, a_n, j, k)) \in \Delta'$, for all $b_j \in \Gamma$. This transition corresponds to the case where no pop or zero test operation were performed previously.
   - $((q, p_1, \cdots, p_n, a_1, \cdots, a_n, j, l), \mathbf{Pop}_j(b_j, p'_j), (q', p_1, \cdots, p_{j-1}, p'_j, p_{j+1}, \cdots, p_n, a_1, \cdots, a_{j-1}, b_j, a_{j+1}, \cdots, a_n, j, l)) \in \Delta'$. This transition corresponds to poping from the currently active stack.
   - For any $l > 2, i \neq j$, $((q, p_1, \cdots, p_n, a_1, \cdots, a_n, i, l), \mathbf{Pop}_j(b_j, p'_j), (q', p_1, \cdots, p_{j-1}, p'_j, p_{j+1}, \cdots, p_n, a_1, \cdots, a_{j-1}, b_j, a_{j+1}, \cdots, a_n, j, l-1)) \in \Delta'$. This transition corresponding to a pop from stack-j when the currently active stack is i.
   - For any $i \neq j$ and $\delta(p'_j, q') \in F_j$, $((q, p_1, \cdots, p_n, a_1, \cdots, a_n, i, 2), \mathbf{Int}, q_w) \in \Delta'$.
   - For any $i \neq j$ and $\delta(p'_j, q') \notin F_j$, $((q, p_1, \cdots, p_n, a_1, \cdots, a_n, i, 2), \mathbf{Int}, q_l) \in \Delta'$.
4. For each $(q, \mathbf{Zero}_j, q') \in \Delta$ we add the following transitions.
   - $((q, p_1, \cdots, p_n, a_1, \cdots, a_{j-1}, \bot, \cdots, a_n, 0, k), \mathbf{Zero}_j, (q', p_1, \cdots, p_{j-1}, \cdots, p_n, a_1, \cdots, a_{j-1}, \bot, \cdots, a_n, j, k)) \in \Delta'$.
   - $((q, p_1, \cdots, p_n, a_1, \cdots, a_{j-1}, \bot, \cdots, a_n, j, l), \mathbf{Zero}_j, (q', p_1, \cdots, p_{j-1}, \cdots, p_n, a_1, \cdots, a_{j-1}, \bot, \cdots, a_n, j, l)) \in \Delta'$, for all $l \in [2..k]$.
   - For all $l > 2$ and $i \neq j$, $((q, p_1, \cdots, p_n, a_1, \cdots, \bot, a_{j-1}, \cdots, a_n, i, l)$, $\mathbf{Zero}_j, (q', p_1, \cdots, p_{j-1}, \cdots, p_n, a_1, \cdots, a_{j-1}, \bot, \cdots, a_n, j, l-1)) \in \Delta'$.
   - For any $i \neq j$ and $\delta(s_j, q') \in F_j$, $((q, p_1, \cdots, p_n, a_1, \cdots, a_{j-1}, \bot, \cdots, a_n, i, 2)$, $\mathbf{Int}, q_w) \in \Delta'$.
   - For any $i \neq j$ and $\delta(s_j, q') \in F_j$, $((q, p_1, \cdots, p_n, a_1, \cdots, a_{j-1}, \bot, \cdots, a_n, i, 2)$, $\mathbf{Int}, q_w) \in \Delta'$.
5. We further add $(q_l, \mathbf{Int}, q_l)$ and $(q_w, \mathbf{Int}, q_w)$ to the transitions

Observe that any run of such a system may involve at most $k-1$ phases, as every change of phase results in a reduction in the last component. After $k-1$ reductions, we end up in one of the states q_w or q_l. For correctness of the construction, we first define a ranking function σ' as follows. $\sigma'(q_w) = 0$, $\sigma'(q_l) = 1$ and for all other states $s = (q, p_1, \cdots, p_n, a_1, \cdots, a_n) \in Q_{M(k)}$, we let

$\sigma'(s) = \sigma(q)$. Similarly we define τ' as $\tau'(q_w) = 0$, $\tau'(q_l) = 1$ and for all other states $s = (q, p_1, \cdots, p_n, a_1, \cdots, a_n) \in Q_{M(k)}$, we let $\tau'(s) = \tau(q)$ and we show that we may associate positions of the form (c, i, k) in the bounded-phase game on (k, M, τ, σ) with positions of the form $(d, i, k-1)$ in the bounded-phase game on $(k-1, M(k), \tau', \sigma')$ that preserves the winner.

For a sequence $w = a_n a_{n-1} \dots a_1 a_0 \in (\Gamma \setminus \{\bot\})^+ \bot$ and $1 \leq j \leq l$, let $\rho_j(w) = (a_{n-1}, p_{n-1}) \dots (a_2, p_2)(a_1, p_1)(a_0, p_0)\bot$ (we let $\rho_j(\bot) = \bot$) where $p_0 = s_j$ and for all $i \in [1..n]$, $p_i = \delta^{B_j}(p_{i-1}, a_{i-1})$. Further, let $\boldsymbol{\delta_j}(w) = \delta^{B_j}(p_{n-1}, a_{n-1})$ (we let $\boldsymbol{\delta_j}(\bot) = p_0$). We now define the map $\mathbf{F}$ from the k-bounded-phase parity game on (k, A, τ, σ) to the $k-1$-bounded-phase parity game on the game $(k-1, A(k), \tau, \sigma)$ as $\mathbf{F}((q, \gamma_1, \cdots, \gamma_n), i, j) = (((q, \boldsymbol{\delta_1}(\gamma_1), \cdots, \boldsymbol{\delta_n}(\gamma_n), \mathrm{Top}(\gamma_1), \cdots, \mathrm{Top}(\gamma_l), i, j), \rho_1(\gamma_1), \cdots, \rho_l(\gamma_n)), i, j)$, if $j > 1$ (where Top is a function that returns top of the stack), $\mathbf{F}((q, \gamma_1, \cdots, \gamma_n), i, 1) = q_w$ if $(q, \gamma_1, \cdots, \gamma_n)$ is winning for player-0 and $\mathbf{F}((q, \gamma_1, \cdots, \gamma_n), i, 1) = q_l$ if $(q, \gamma_1, \cdots, \gamma_n)$ is losing for player-0. Now using arguments similar to lemma-3, we get the following.

Lemma 7. *The map* $\mathbf{F}$ *satisfies the following properties*

1) *The ownership and the rank of all positions* (c, i, j) *with* $j > 1$ *are preserved.* **2)** *For any configuration* (c, i, j) *if* $(c, i, j) \to (c', i', j')$ *with* $j' > 1$ *then* $\mathbf{F}((c, i, j)) \to \mathbf{F}((c', i', j'))$. **3)** *For any configuration* (c, i, j) *if* $\mathbf{F}(c, i, j) \to d$ *for any* $d \notin \{q_w, q_l\}$ *then there is* (c', i', j') *with* $(c, i, j) \to (c', i', j')$, $j' > 1$ *such that* $\mathbf{F}(c', i', j') = d$. **4)** *If* $(c, i, 2) \to (c', i', 1)$, *then* $\mathbf{F}(c, i, 2) \to q_w$ *iff* $(c', i', 1)$ *is a winning position and* $\mathbf{F}(c, i, 2) \to q_l$ *iff* $(c', i', 1)$ *is a losing position for player-*0.

Hence we can effectively determine the set of all positions that are winning for player 0, *in the bounded-phase parity game* (k, M, τ, σ) .

We have shown how to reduce a k-bounded-phase game to a $k-1$-bounded-phase game. However note that each such a reduction is exponential in the size of the system. Since we do as many such reductions as the number of phases, the overall complexity will be a tower of exponents. Hence the overall reduction is NON-ELEMENTARY in nature. This completes the proof of theorem 3. Next we show that such a blow up cannot be avoided.

7 Lower bounds for bounded phase parity game

We show that the satisfiability of a first order formula with ordering relation over natural numbers, can be reformulated as a bounded-phase parity game over *MPDS*. We first briefly recall the first order theory of natural numbers with ordering relation ($FO(<)$).

Let $\mathcal{V}$ be countably infinite set of variables, we will use $x, y, z, x_1, x_2 \cdots$ to refer to the variables in $\mathcal{V}$. The set of terms in $FO(<)$ is defined as $t := x \mid t < t \mid t = t$. The set of formulas is defined to be $\Psi := t \mid \neg t \mid \Psi \vee \Psi \mid \Psi \wedge \Psi \mid \forall x \Psi \mid \exists x \Psi$. The notion of free, bound (quantified) variable are defined as usual. We write $Var(\Psi) \subseteq \mathcal{V}$ to denote the set of all free variables (unquantified variables) of Ψ.

Given any formula Ψ over variables $\mathcal{V}$, we define a valuation function as $\mu : \mathcal{V} \mapsto \mathbb{N}$ in the usual way. Given any formula Ψ and a valuation function μ,

we call μ a model of Ψ, iff $\mu \models \Psi$. A formula with no free variables is called a sentence. A sentence is said to be satisfied iff there is some valuation function that satisfies it. Note that the negation is defined only on the atomic formulas. However, given any formula Ψ, we can easily obtain another formula Duel(Ψ) such that for any model μ of Ψ, $\mu \models \Psi$ iff $\mu \not\models$ Duel(Ψ). w.l.o.g we will assume that formulas that we deal henceforth with will be in prenix normal form.

Given a formula Ψ and its model μ we define the linearisation of μ w.r.t. Ψ to be a word of the form $x_1 a^{j_1} x_2 a^{j_2} \cdots x_n a^{j_n} \perp$, where $\{x_1, \cdots, x_n\} = Var(\Psi)$ and for each $k \in [1..n]$, $\mu(x_{i_k}) = j_k + j_{k-1} \cdots j_n$. Similarly, for any set of variables $\mathcal{V}$, we say a string $(\alpha = x_n a^{i_n} x_{n-1} a^{i_{n-1}} \cdots x_1 a^{i_1}) \in (\mathcal{V} \cup a)^*$ is a valuation string if for all $l, k \in [1..n]$, we have $l \neq k \implies x_l \neq x_k$ (i.e. each x_i appears at most once). Firstly, given any valuation string $\alpha = x_n a^{i_n} x_{n-1} a^{i_{n-1}} \cdots x_1 a^{i_1}$ and a set of variable $\mathcal{V}$, we define $\mu_\alpha^{\mathcal{V}}$ as, for any $j \in [1..n]$, $\mu_\alpha^{\mathcal{V}}(x_j) = a^{i_j} + a^{i_{j-1}} + \cdots + a^{i_1}$, i.e. it maps the variables x_j, to a value equal to number of a's appearing before it in α. For any $x \in \mathcal{V}$ such that x does not appear in α, we let $\mu_\alpha(x) = 0$.

Given any formula Ψ, we use $Cl(\Psi)$ to indicate the set of all formulas obtained by closing the formula Ψ over subformulas. Note that even if Ψ is a sentence, elements of $Cl(\Psi)$ can have free variables. We now show that satisfiability of first order formula over $(N, <)$ (known to have non-elementary complexity [9]) can be reduced to a parity games over the bounded-phase MPDS.

The informal idea is to construct an MPDS, in which the state space contains the subformulas of the given formula Ψ (i.e. $Cl(\Psi)$), along with some intermediary states. The MPDS starts with empty stack and the formula Ψ. At any point in the game, the *MPDS* maintains the unprocessed part of the formula $\phi \in Cl(\Psi)$ as part of its state space and the linear encoding (linearisation) of the current valuation μ (w.r.t. ϕ) in its stack . There are two parts to the game depending on whether the unprocessed part begins with a quantifier or not. If the unprocessed part of formula begins with a quantifier $\forall$, then player-1 strips off the quantifier and assigns a valuation to the corresponding variable by modifying the stack. If it begins with a $\exists$ quantifier then the valuation is provided by player-0. If the valuation that the player wishes to provide is less than the variables already in the stack, the elements are moved to stack-2 till the appropriate position is found, the variable is placed in this position and the elements from stack-2 are moved back to stack-1. If the valuation that the player wishes to provide is greater than all the variables present in the stack, extra a's are appended and the variable is placed. If the outer most operator is $\wedge$, then the player-1 chooses a subformula and the game proceeds. If the outer most operator is $\vee$, then the player-0 selects a subformula. The game proceeds till the unprocessed part is an atomic formula, in which case it can easily be verified based on the valuations in the stack.

We will formally describe the construction of the *MPDS* $M_\Psi = (2, Q, \Gamma = \{a, \perp\} \cup Var(\Psi), \Delta, q_0)$ in two parts. The first part describes the moves till we reach an atomic formula. It contains the following set of states $Cl(\Psi) \cup Cl(\Psi) \times \{\text{lt}, \text{gt}, m_{1,2}, m_{2,1}\} \cup Cl(\Psi) \times \{m_{1,2}, m_{2,1}\} \times (\{a\} \times Var(\Psi))$. The transition relation Δ is defined as follows. We will used $?x$ to denote either of $\exists x$ or $\forall x$.

1. For all $\psi_1 \wedge \psi_2 \in Cl(\Phi)$, the transitions $(\psi_1 \wedge \psi_2, \mathbf{Int}, \psi_1), (\psi_1 \wedge \psi_2, \mathbf{Int}, \psi_2) \in \Delta$. Similarly for all $\psi_1 \vee \psi_2 \in Cl(\Phi)$, the transitions $(\psi_1 \vee \psi_2, \mathbf{Int}, \psi_1)$ and $(\psi_1 \vee \psi_2, \mathbf{Int}, \psi_2) \in \Delta$
2. For all $?x.\psi \in Cl(\Phi)$, we add $(?x.\psi, \mathbf{Int}, (?x.\psi, \mathrm{lt}))$ and $(?x.\psi, \mathbf{Int}, (?x.\psi, \mathrm{gt})) \in \Delta$, this transition enables guessing whether the current variable x needs to be inserted in between the existing variable (valuation falls below the current maximum) or needs to be inserted on top (is greater than the current maximum).
3. We also add $((?x.\psi, \mathrm{gt}), \mathbf{Push}_1(a), (?x.\psi, \mathrm{gt})) \in \Delta$ (pushes a into stack-1 to increase possible valuation for x) and $((?x.\psi, \mathrm{gt}), \mathbf{Push}_1(x), \psi) \in \Delta$ (Marks position of x and shift to the sub-formula).
4. We add $((?x.\psi, \mathrm{lt}), \mathbf{Int}, (?x.\psi, m_{1,2})) \in \Delta$ (Begin moving from stack-1 to 2), $((?x.\psi, m_{1,2}),\ \mathbf{Pop}_1(a),\ (?x.\psi, m_{1,2}, a)) \in \Delta$ and $((?x.\psi, m_{1,2}, a), \mathbf{Push}_2(a), (?x.\psi, m_{1,2})) \in \Delta$, $\forall a \in \Gamma \setminus \{\bot\}$ (moves values from stack-1 to 2)
5. Similarly we add $((?x.\psi, m_{1,2}), \mathbf{Push}_1(x), (?x.\psi, m_{2,1})) \in \Delta$ (Begin moving from stack-2 back to 2), $((?x.\psi, m_{2,1}),\ \mathbf{Pop}_2(a),\ (?x.\psi, m_{2,1}, a)) \in \Delta$ and $((?x.\psi, m_{2,1}, a),\ \mathbf{Push}_1(a),\ (?x.\psi, m_{2,1})) \in \Delta$, $\forall a \in \Gamma \setminus \{\bot\}$ (moves values from stack-2 to 1). We also add $((?x.\psi, m_{2,1}), \mathbf{Zero}_2, \psi) \in \Delta$ (Move to the next sub-formula) .

In the second part, we describe the state space starting at a state of the form $(x = y)$ or $(x < y)$ that determines winner of the game. It contains the following set of states $\{x = y, x < y, a_y \mid x, y \in V\} \cup \{T, F\}$. The transitions are as below.

1. $(x = y, \mathbf{Pop}_1(z), x = y) \in \Delta$, for all $z \in V \setminus \{x, y\} \cup \{a\}$, pop all elements other than x, y.
2. $(x = y, \mathbf{Pop}_1(z), z') \in \Delta$, for $z \in \{x, y\}, z' \in \{x, y\} \setminus \{z\}$, as soon as one of $\{x, y\}$ is seen (say x) goto a state expecting to see the other variable (y if we saw x previously).
3. For $x \in V$, we add $(x, \mathbf{Pop}_1(a), F) \in \Delta$, if we see an a when we are expecting a variable in $x \in V$, we goto the losing state F.
4. For $x, z \in V$, $z \neq x$ we add $(x, \mathbf{Pop}_1(z), x) \in \Delta$, if we see a variable other than x, we skip.
5. For $x \in V$, we add $(x, \mathbf{Pop}_1(x), T) \in \Delta$, if we see a variable x, we goto winning state.
6. We also add $(T, \mathbf{Int}, T)$ and $(F, \mathbf{Int}, F)$ to Δ.

The set transitions needed for $\neg(x = y), (x < y), \neg(x < y)$ are similar. We will now consider the bounded-phase parity game given by $(|\Psi|, \mathcal{C}(M_\Psi), \tau, \sigma)$ where $\sigma : Q \mapsto \{0, 1\}$ and $\tau : Q \mapsto \{0, 1\}$ are defined as:

- We let $\sigma(T) = 0$ and $\sigma(F) = 1$. We let $\sigma((\exists x.\Psi', gt)) = 1$ and $\sigma((\forall x.\Psi', gt)) = 0$ (this will ensure that either of the player cannot simply win by just pushing elements onto the stack). For all other $q \in Q$, we let $\sigma(q) = 0$.
- For any state s such that its subformula component is of the form, $\forall x.\Psi'$ or $\Psi_1 \wedge \Psi_2$, we let $\tau(s) = 1$ (player-1 position). Otherwise, $\tau(s) = 0$ i.e. we let all other states to be player-0 position.

Notice that along any positions in the game, where the state is only a subformula from $Cl(\Psi)$, the stack content of the first stack α is a valuation string. This is easy to see since by nature of the formula we have assumed that along any path, we can never encounter the same variable twice. Clearly such a $\mu^{\mathcal{V}}_{\alpha}$ function is a valuation function. We show in Lemma 8, that along positions in game graph where the state is only a subformula from $Cl(\Psi)$, the valuation function constructed out of the content of stack 1 is actually a model of the subformula iff player-0 has a winning strategy from that position.

Lemma 8. *Give any configuration $c \in \mathcal{C}(M)$ which is of the form $(\Psi, \alpha\bot, \bot)$ where $\alpha = x_n a^{i_n} x_{n-1} a^{i_{n-1}} \cdots x_1 a^{i_1} \in (V\Gamma^*)^*$ is a valuation string containing all the free variables of Ψ, then $\mu^{\mathcal{V}}_{\alpha} \models \Psi$ iff player-0 has a bounded-phase winning strategy from c.*

Corollary 1. *For any setence Ψ, Ψ is satisfiable iff $(\Psi, \bot, \bot)$ is winning for player 0 in the game $(|\Psi|, \mathcal{C}(M_\Psi), \tau, \sigma)$. Thus deciding bounded-phase games has a* Non-elementary *lower bound.*

References

1. Cachat, T.: Uniform solution of parity games on prefix-recognizable graphs. Electr. Notes Theor. Comput. Sci. 68 (2002)
2. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy (extended abstract). In: 32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991 (1991)
3. Kupferman, O., Piterman, N., Vardi, M.Y.: An automata-theoretic approach to infinite-state systems. In: Time for Verification, Essays in Memory of A. Pnueli (2010)
4. Martin, D.A.: Borel determinacy. Annals of Mathematics 102 (1975)
5. Qadeer, S., Rehof, J.: Context-bounded model checking of concurrent software. In: TACAS. LNCS, Springer (2005)
6. Saivasan, P.: Analysis of Automata-theoretic models of Concurrent Recursive Programs. Ph.D. thesis, Chennai Mathematical Institute (2016)
7. Serre, O.: Note on winning positions on pushdown games with [omega]-regular conditions. Inf. Process. Lett. 85 (2003)
8. Seth, A.: Games on multi-stack pushdown systems. In: LFCS. Springer (2009)
9. Stockmeyer., L.J.: The Complexity of Decision Problems in Automata Theory and Logic. Ph.D. thesis, MIT, Cambridge, Massasuchets, USA (1974)
10. Torre, S.L., Madhusudan, P., Parlato, G.: A robust class of context-sensitive languages. In: LICS. IEEE Computer Society (2007)
11. Torre, S.L., Madhusudan, P., Parlato, G.: Reducing context-bounded concurrent reachability to sequential reachability. In: CAV. LNCS, Springer (2009)
12. Walukiewicz, I.: Pushdown processes: Games and model checking. In: CAV. LNCS, Springer (1996)