Enriched μ–Calculus Pushdown Module Checking

Alessandro Ferrante[1], Aniello Murano[2], and Mimmo Parente[1]

[1] Università di Salerno, Via Ponte don Melillo, 84084 - Fisciano (SA), Italy
[2] Università di Napoli Federico II, Via Cintia, 80126 - Napoli, Italy

Abstract. The model checking problem for open systems (called *module checking*) has been intensively studied in the literature, both for finite–state and infinite–state systems. In this paper, we focus on pushdown module checking with respect to decidable fragments of the fully enriched μ–calculus. We recall that finite–state module checking with respect to fully enriched μ–calculus is undecidable and hence the extension of this problem to pushdown systems remains undecidable as well. On the contrary, for the fragments of the fully enriched μ–calculus we consider here, we show that pushdown module checking is decidable and solvable in double–exponential time in the size of the formula and in exponential time in the size of the system. This result is obtained by exploiting a classical automata–theoretic approach via *pushdown nondeterministic parity tree automata*. In particular, we reduce in exponential time our problem to the emptiness problem for these automata, which is known to be decidable in EXPTIME. As a key step of our algorithm, we show an exponential improvement of the construction of a nondeterministic parity tree automaton accepting all models of a formula of the considered logic. This result, does not only allow our algorithm to match the known lower bound, but also to investigate decision problems related to the fragments of the enriched μ-calculus in a greatly simplified manner.

1 Introduction

In system design, one of the most challenging problem is to check for system correctness. *Model-checking* is a very important development in the area of formal design verification, which allows to automatically verify, in a suitable way, the ongoing behaviors of *reactive systems* ([CE81,QS81]). In this verification method (for a survey, see [CGP99]), the behavior of a system, formally described by a mathematical model, is checked against a behavioral constraint, possibly specified by a formula in an appropriate temporal logic.

In system modeling, we distinguish between *closed* and *open* systems [HP85]. While the behavior of a closed system is completely determined by the state of the system, the behavior of an open system depends on the ongoing interaction with its environment [Hoa85]. Model checking algorithms used for the verification of closed systems are not appropriate for open systems. In the latter case, we should check the system with respect to arbitrary environments and take into account uncertainty regarding the environment. In [KVW01], model checking has been extended from closed finite–state systems to open finite–state systems. In

such a framework, the open finite–state system is described by a labeled state–transition graph called *module* whose set of states is partitioned into *system states* (where the system makes a transition) and *environment states* (where the environment makes a transition). Given a module $\mathcal{M}$, describing the system to be verified, and a temporal logic formula φ, specifying the desired behavior of the system, the problem of model checking a module, called *module checking*, asks whether for all possible environments, $\mathcal{M}$ satisfies φ. Module checking thus involves not only checking that the full computation tree $\langle T_{\mathcal{M}}, V_{\mathcal{M}} \rangle$ obtained by unwinding $\mathcal{M}$ (which corresponds to the interaction of $\mathcal{M}$ with a maximal environment) satisfies φ (which corresponds to model checking $\mathcal{M}$ with respect to φ), but also that all trees obtained from it, by pruning subtrees of environment nodes (these trees correspond to all possible choices of the environment and are collected in $exec(\mathcal{M})$) satisfy φ. To see an example, consider a two-drink dispenser machine that serves, upon request, tea or coffee. The machine is an open system and an environment for the system is an infinite line of thirsty people. Since each person in the line can prefer either both tea and coffee, or only tea, or only coffee, each person suggests a different disabling of the external nondeterministic choices. Accordingly, there are many different possible environments to consider. In [KVW01], it has been shown that while for linear logics model and module checking coincide, module checking for specification given in *CTL* and CTL^* is exponentially harder than model checking. Indeed, *CTL* and CTL^* module checking is, respectively, EXPTIME–complete and 2EXPTIME–complete, in the size of the formula, and both PTIME–complete in the size of the system.

Recently, module checking has been also investigated with respect to formulas of the *fully enriched μ–calculus* and some of its fragments. The *μ–calculus* is a propositional modal logic augmented with least and greatest fixpoint operators [Koz83]. *Fully enriched μ–calculus* is the extension the μ–calculus with *inverse programs*, *graded modalities*, and *nominals*. Intuitively, inverse programs allow to travel backwards along accessibility relations [Var98], nominals are propositional variables interpreted as singleton sets [SV01], and graded modalities enable statements about the number of successors and predecessors of a state [KSV02]. By dropping at least one of the additional constructs, we get a fragment of the fully enriched μ-calculus. Here, we call all this fragments (not fully) *enriched μ–calculus*, for simplicity. In [BP04], it has been shown that satisfiability is undecidable in the *fully enriched μ–calculus*. On the other hand, it has been shown in [SV01,BLMV06] that satisfiability for enriched μ-calculus is decidable and EXPTIME-complete. The upper bound result is based on an automata–theoretic approach via *two-way graded alternating parity tree automata (*2GAPT*)*. Intuitively, these automata generalize alternating automata on infinite trees in a similar way as the fully enriched μ–calculus extends the standard μ–calculus: $2GAPT$ can move up to a node's predecessor (by analogy with inverse programs), move down to *at least n* or *all but n* successors (by analogy with graded modalities), and jump directly to the roots of the input forest (which are the analogues of nominals). Using these automata, along with the fact that the en-

riched μ-calculus enjoys the *quasi-forest model property*[1], it has been shown in [SV01,BLMV06] that it is possible to build a $2GAPT$ accepting all trees encoding of all quasi-forest models of any enriched μ-calculus formula. Then, the exponential-upper bound follows from the fact that $2GAPT$ can be exponentially translated in *nondeterministic graded parity tree automata (*GNPT*)*, and the emptiness problem for $GNPT$ is solvable in PTIME [KPV02].

Coming back to the module checking problem for the fully enriched μ–calculus and its fragments, in [FM07] this problem has been deeply investigated with respect to finite–state systems. To see an example, consider the previous two-drink dispenser machine with the extra ability of having (bounded) multiple choice for coffee and tea. Suppose now that we want to check that whenever k coffees are served it is because we choose them in the past. This can be performed using a formula of the fully enriched μ–calculus, which truth value depends on the past (using backward modality) and k identical coffee events in the future (using graded modalities). In [FM07], it has been shown that module checking is undecidable for formulas of the fully enriched μ–calculus, while it is decidable and EXPTIME–complete if enriched μ–calculus formulas are considered.

In [BMP05], the module checking technique has been also extended to infinite-state systems by considering *open pushdown systems* (*OPD* for short). These are pushdown systems augmented with finite information that allow to partition the set of configurations (in accordance with the control state and the symbol on the top of the stack) into *system configurations* and *environment configurations*. To see an example of an open pushdown system, consider an extension of the above mentioned two drink–dispenser machine, with the additional constraint that a coffee can be served only if the number of coffees served up to that time is smaller than that of teas served. Such a machine can be clearly modeled as an open pushdown system (the stack is used to guarantee the inequality between served coffees and teas). In [BMP05], it has been shown that pushdown module checking is 2EXPTIME–complete for *CTL* and 3EXPTIME–complete for CTL^*.

In this paper, we extend the pushdown module checking problem to the *enriched μ-calculus* and, by exploiting an automata-theoretic approach via pushdown tree automata, we show that this problem is decidable and solvable in 2EXPTIME. As for finite–state open systems dealing with backward modalities, given an *OPD* $\mathcal{S}$ and a module $\mathcal{M}$ induced from $\mathcal{S}$, we interpret the logic on the unwinding of $\mathcal{M}$ by considering, for each configuration, the past configurations as inverse next configurations. In this case, in order to use a pushdown tree automaton we constraint the stack operations of the *OPD* in such a way that whenever it reads the top symbol of the stack, it can only cancel it from the stack, change it with another stack symbol, or add a new stack symbol on top of it. Also note that the use of two-way pushdown tree automata would be useless along with our approach since they recognize recursively-enumerable languages [Sal96] and thus the emptiness problem is undecidable for them.

The algorithm we propose to solve the considered problem works as follows. Given an *OPD*, a module $\mathcal{M}$ induced by the configurations of the *OPD*, and an

[1] A quasi forest is a forest where nodes can have roots as successors

enriched μ-calculus formula φ, we first build in a polynomial time a pushdown Büchi tree automaton (PD–NBT) $\mathcal{A}_{\mathcal{M}}$, accepting $exec(\mathcal{M})$. The construction of $\mathcal{A}_{\mathcal{M}}$ we propose here extends that used in [BMP05] by also taking into account that $\mathcal{M}$ requires to be unwound in a quasi-forest, rather than a tree, with both nodes and edges labeled. Thus, the set $exec(\mathcal{M})$ is a set of quasi-forests, and the automaton $\mathcal{A}_{\mathcal{M}}$ we build will accept all trees encodings of all quasi-forests of $exec(\mathcal{M})$. From the formula side, accordingly to [BLMV06], we can build in a polynomial time a $2GAPT$ $\mathcal{A}_{\neg\varphi}$ accepting all models of $\neg\varphi$, with the intent to check that no models of $\neg\varphi$ are in $exec(\mathcal{M})$. Thus, we check that $\mathcal{M} \models_r \varphi$ by checking whether $\mathcal{L}(\mathcal{A}_{\mathcal{M}}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi})$ is empty. To the best of our knowledge, the latter problem can be only solved in triple-exponential time. For example, by recalling that $2GAPT$ can be exponentially translated into $GNPT$ [BLMV06], that $GNPT$ can be exponentially translated into nondeterministic parity tree automata (NPT) [KSV02], that intersecting a PD–NBT with an NPT gives a pushdown parity tree automaton [BMP05], and that the emptiness problem for the latter is solvable in EXPTIME [KPV02]. In this paper, by showing a non-trivial exponential reduction of $2GAPT$ into NPT, we show a 2EXPTIME upper bound for the addressed problem. Since the pushdown module checking problem for CTL is 2EXPTIME-hard, we get that the problem is then 2EXPTIME-complete.

Let us stress that by using our exponential improvement on translating $2GAPT$ into NPT we can also get results concerning decision problems for the enriched μ–calculus (such as the satisfiability and module checking problems [SV01,BLMV06,FM07]) with more simplified proofs.

2 Preliminaries

Labeled Forests. For a finite set X, we denote the *size* of in X by $|X|$, the set of words over X by X^*, the empty word by ε, and with X^+ we denote $X^* \setminus \{\varepsilon\}$. Given a word w in X^* and a symbol x of X, we use $w \cdot x$ to denote the word wx. Let $\mathbb{N}$ be the set of positive integers. For $n \in \mathbb{N}$, let $\mathbb{N}$ be denote the set $\{1, 2, \ldots, n\}$. A *forest* is a set $F \subseteq \mathbb{N}^+$ such that if $x \cdot c \in F$ where $x \in \mathbb{N}^+$ and $c \in \mathbb{N}$, then also $x \in F$. The elements of F are called *nodes*, and words consisting of a single natural number are *roots* of F. For each root $r \in F$, the set $T = \{r \cdot x \mid x \in \mathbb{N}^* \text{ and } r \cdot x \in F\}$ is a *tree* of F (the tree *rooted in* r). For $x \in F$, the nodes $x \cdot c \in F$ where $c \in \mathbb{N}$ are the *successors* of x, denoted $sc(x)$, and x is their *predecessor*. The number of successors of a node x is called the *degree* of x ($deg(x)$). The degree h of a forest F is the maximum of the degrees of all nodes in F and the number of roots. A forest with degree k is a h-*ary* forest. A full h-ary forest is a forest having h roots and all nodes with degree h.

Let $F \subseteq \mathbb{N}^+$ be a forest and x a node in F. As a convention, we take $x \cdot \varepsilon = \varepsilon \cdot x = x$, $(x \cdot c) \cdot -1 = x$, and $n \cdot -1$ as undefined, for $n \in \mathbb{N}$. We call x a *leaf* if it has no successors. A *path* π in F is a word $\pi = a_1 a_2 \ldots$ of F such that a_1 is a root of F and for every $a_i \in \pi$, either a_i is a leaf (i.e., π ends in a_i) or a_i is a predecessor of a_{i+1}. Given two alphabets Σ_1 and Σ_2, a (Σ_1, Σ_2)–labeled forest is a triple $\langle F, V, E\rangle$, where F is a forest, $V : F \to \Sigma_1$ maps each node of F to a letter in Σ_1, and $E : F \times F \to \Sigma_2$ is a partial function that maps

each pair (x, y), with $y \in children(x)$, to a letter in Σ_2. As a particular case, we consider a forest without labels on edges as a Σ_1–labeled forest $\langle F, V\rangle$, and a *tree* as a forest containing exactly one tree. A *quasi–forest* is a forest where each node may also have roots as successors. For a node x of a quasi–forest, we set $children(x)$ as $sc(x) \setminus \mathbb{N}$. All the other definitions regarding forests easily extend to quasi–forest. Notice that in a quasi–forest, a root can also have several predecessors, while every other node has just one. Clearly, a quasi–forest can be always transformed in a forest by removing root successors.

Fully Enriched μ–Calculus. Let AP, Var, $Prog$, and Nom be finite and pairwise disjoint sets of *atomic propositions*, *propositional variables*, *atomic programs*, and *nominals*. A *program* is an atomic program a or its converse a^-. The set of *formulas of the fully enriched μ–calculus* is the smallest set such that (i) **true** and **false** are formulas; (ii) p and $\neg p$, for $p \in AP \cup Nom$, are formulas; (iii) $x \in Var$ is a formula; (iv) if φ_1 and φ_2 are formulas, α is a program, n is a non–negative integer, and y is a propositional variable, then the following are also formulas: $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$, $\langle n, \alpha\rangle\varphi_1$, $[n, \alpha]\varphi_1$, $\mu y.\varphi_1(y)$, and $\nu y.\varphi_1(y)$.

Observe that we use positive normal form, i.e., negation is applied only to atomic propositions. We call μ and ν *fixpoint operators*. A propositional variable y occurs *free* in a formula if it is not in the scope of a fixpoint operator. A *sentence* is a formula that contains no free variables.

We refer often to the *graded modalities* $\langle n, \alpha\rangle\varphi_1$ and $[n, \alpha]\varphi_1$ as respectively *atleast formulas* and *allbut formulas* and assume that the integers in these operators are given in binary coding: the contribution of n to the length of the formulas $\langle n, \alpha\rangle\varphi$ and $[n, \alpha]\varphi$ is $\lceil \log n \rceil$ rather than n. We refer to fragments of the fully enriched μ–calculus as follows. We say that a formula φ of the fully enriched μ–calculus is also a formula of the *hybrid graded*, *full hybrid*, or *full graded* μ–calculus if φ does not have inverse programs, graded modalities, or nominals, respectively. If at least one of the above holds, we also say that φ is an *enriched μ–calculus* formula. To avoid confusion, we observe that enriched formulas are also formulas of the full calculus, while the converse is not always true. We recall that enriched formulas enjoy the quasi-forest model property (as showed in [BLMV06] and [SV01]), while fully enriched formulas does not [BP04].

The semantics of the fully enriched μ–calculus is defined with respect to a *Kripke structure*, i.e., a tuple $\mathcal{K} = \langle W, W_0, R, L\rangle$ where W is a non–empty set of *states*, $W_0 \subseteq W$ is the set of initial states, $R : Prog \to 2^{W \times W}$ is a function that assigns to each atomic program a transition relation over W, and $L : AP \cup Nom \to 2^W$ is a labeling function that assigns to each atomic proposition and nominal a set of states such that the sets assigned to nominals are singletons and subsets of W_0. To deal with inverse programs, we extend R as follows: for each $a \in Prog$, set $R(a^-) = \{(v, u) : (u, v) \in R(a)\}$. If $(w, w') \in R(\alpha)$, we say that w' is an *α–successor* of w. Informally, an *atleast* formula $\langle n, \alpha\rangle\varphi$ holds at a state w of a Kripke structure $\mathcal{K}$ if φ holds at least in $n + 1$ α–successors of w. Dually, the *allbut* formula $[n, \alpha]\varphi$ holds in a state w of a Kripke structure $\mathcal{K}$ if φ holds in all but at most n α–successors of w. Note that $\neg\langle n, \alpha\rangle\varphi$ is equivalent

to $[n,\alpha]\neg\varphi$, and that the modalities $\langle\alpha\rangle\varphi$ and $[\alpha]\varphi$ of the standard μ–calculus can be expressed as $\langle 0,\alpha\rangle\varphi$ and $[0,\alpha]\varphi$, respectively.

To formalize semantics, we introduce valuations. Given a Kripke structure $\mathcal{K} = \langle W, W_0, R, L\rangle$ and a set $\{y_1,\ldots,y_n\}$ of variables in *Var*, a *valuation* $\mathcal{V} : \{y_1,\ldots,y_n\} \to 2^W$ is an assignment of subsets of W to the variables $y_1,\ldots,y_n$. For a valuation $\mathcal{V}$, a variable y, and a set $W' \subseteq W$, we denote by $\mathcal{V}[y \leftarrow W']$ the valuation obtained from $\mathcal{V}$ by assigning W' to y. A formula φ with free variables among $y_1,\ldots,y_n$ is interpreted over the structure $\mathcal{K}$ as a mapping $\varphi^{\mathcal{K}}$ from valuations to 2^W, i.e., $\varphi^{\mathcal{K}}(\mathcal{V})$ denotes the set of points that satisfy φ under valuation $\mathcal{V}$. The mapping $\varphi^{\mathcal{K}}$ is defined inductively as follows:

- $\mathbf{true}^{\mathcal{K}}(\mathcal{V}) = W$ and $\mathbf{false}^{\mathcal{K}}(\mathcal{V}) = \emptyset$;
- for $p \in AP \cup Nom$, we have $p^{\mathcal{K}}(\mathcal{V}) = L(p)$ and $(\neg p)^{\mathcal{K}}(\mathcal{V}) = W \setminus L(p)$;
- for $y \in Var$, we have $y^{\mathcal{K}}(\mathcal{V}) = \mathcal{V}(y)$;
- $(\varphi_1 \wedge \varphi_2)^{\mathcal{K}}(\mathcal{V}) = \varphi_1^{\mathcal{K}}(\mathcal{V}) \cap \varphi_2^{\mathcal{K}}(\mathcal{V})$ and $(\varphi_1 \vee \varphi_2)^{\mathcal{K}}(\mathcal{V}) = \varphi_1^{\mathcal{K}}(\mathcal{V}) \cup \varphi_2^{\mathcal{K}}(\mathcal{V})$;
- $(\langle n,\alpha\rangle\varphi)^{\mathcal{K}}(\mathcal{V}) = \{w : |\{w' \in W : (w,w') \in R(\alpha) \text{ and } w' \in \varphi^{\mathcal{K}}(\mathcal{V})\}| \geq n+1\}$;
- $([n,\alpha]\varphi)^{\mathcal{K}}(\mathcal{V}) = \{w : |\{w' \in W : (w,w') \in R(\alpha) \text{ and } w' \notin \varphi^{\mathcal{K}}(\mathcal{V})\}| \leq n\}$;
- $(\mu y.\varphi(y))^k(\mathcal{V}) = \bigcap\{W' \subseteq W : \varphi^{\mathcal{K}}([y \leftarrow W']) \subseteq W'\}$;
- $(\nu y.\varphi(y))^k(\mathcal{V}) = \bigcup\{W' \subseteq W : W' \subseteq \varphi^{\mathcal{K}}([y \leftarrow W'])\}$.

For a state w of a Kripke structure $\mathcal{K}$, we say that $\mathcal{K}$ *satisfies* φ at w if $w \in \varphi^{\mathcal{K}}$. In what follows, a formula φ *counts* up to b if the maximal integer in *atleast* and *allbut* restrictions used in φ is $b-1$.

Open Kripke Structures. In this paper we consider open systems, i.e., systems that interact with their environment and whose behavior depends on this interaction. The (global) behavior of such a system is described by a *module* $\mathcal{M} = \langle W_s, W_e, W_0, R, L\rangle$, which is a Kripke structure where the set of states $W = W_s \cup W_e$ is partitioned in *system states* W_s and *environment states* W_e.

Given a module $\mathcal{M}$, we assume that its states are ordered and the number of successors of each state w is finite. For each state $w \in W$, we denote by $succ(w)$ the ordered tuple (possibly empty) of w's successors. When $\mathcal{M}$ is in a system state w_s, then all the states in $succ(w_s)$ are possible next states. On the other hand, when $\mathcal{M}$ is in an environment state w_e, the possible next states (that are in $succ(w_e)$) depend on the current environment. Since the behavior of the environment is not predictable, we have to consider all the possible sub–tuples of $succ(w_e)$. The only constraint, since we consider environments that cannot block the system, is that not all the transitions from w_e are disabled. In particular, the environment can never disable the transition from which it comes. In more details, suppose that w_e is reached from a node w via an a-labeled edge, then the a^--successor w of w_e cannot be disabled. This is also intuitively consistent with the fact that past environment choices cannot be changed.

The set of all (maximal) computations of $\mathcal{M}$ starting from W_0 is described by a $(W, Prog)$–labeled quasi–forest $\langle F_{\mathcal{M}}, V_{\mathcal{M}}, E_{\mathcal{M}}\rangle$, called *computation quasi–forest*, which is obtained by unwinding $\mathcal{M}$ in the usual way. In particular, since we are dealing with a logic based on inverse operators, the parent of each node is

not be duplicated as its child using inverse programs. Notice that this is coherent with the semantics of the logic we consider here. The problem of deciding, for a given branching–time formula φ over $AP \cup Nom$, whether $\langle F_{\mathcal{M}}, L \circ V_{\mathcal{M}}, E_{\mathcal{M}} \rangle$ satisfies φ at a root node, denoted $\mathcal{M} \models \varphi$, is the usual *model–checking problem* [CE81,QS81]. On the other hand, for an open system $\mathcal{M}$, $\langle F_{\mathcal{M}}, V_{\mathcal{M}}, E_{\mathcal{M}} \rangle$ corresponds to a very specific environment, i.e., a maximal environment that never restricts the set of its next states. Therefore, when we examine a branching–time formula φ w.r.t. $\mathcal{M}$, the formula φ should hold not only in $\langle F_{\mathcal{M}}, V_{\mathcal{M}}, E_{\mathcal{M}} \rangle$, but in all quasi-forests obtained by pruning from $\langle F_{\mathcal{M}}, V_{\mathcal{M}}, E_{\mathcal{M}} \rangle$ subtrees rooted at children of environment nodes, as well as inhibiting some of their jumps to roots, if there are any. The set of these quasi–forests, which collects all possible behaviors of the environment, is denoted by $exec(\mathcal{M})$ and is formally defined as follows. $\langle F, V, E \rangle \in exec(\mathcal{M})$ iff for each $w_i \in W_0$, we have $V(i) = w_i$, and for $x = y \cdot z \in F$ with $z \in \mathbb{N}$, $V(x) = w$, $succ(w) = \langle w_1, \ldots, w_n, w_{n+1}, \ldots, w_{n+m} \rangle$, $succ(w) \cap W_0 = \langle w_{n+1}, \ldots, w_{n+m} \rangle$, and $T(x) = \emptyset$ if $y = \varepsilon$ and $T(x) = \{V(y)\}$ otherwise, there exists $S = \langle w_{i_1}, \ldots, w_{i_p}, w_{i_{p+1}}, \ldots, w_{i_{p+q}} \rangle$ sub-tuple of $succ(w) \backslash T(x)$ such that $p + q \geq 1$ if $y = \varepsilon$ and the following hold:

- if $w \in W_s$, then $S = succ(w) \backslash T(x)$;
- $children(x) = \{x \cdot 1, \ldots, x \cdot p\}$ and for $1 \leq j \leq p$, $V(x \cdot j) = w_{i_j}$, and $E(x, x \cdot j) = \alpha$ if $(w, w_{i_j}) \in R(\alpha)$;
- for $1 \leq j \leq q$, let $x_j \in \mathbb{N}$ such that $V(x_j) = w_{i_{p+j}}$, it holds that $E(x, x_j) = \alpha$ if $(w, w_{i_{p+j}}) \in R(\alpha)$;

In the following, we consider quasi–forests in $exec(\mathcal{M})$ as labeled with $(2^{AP \cup Nom}, Prog)$, i.e., taking the label of a node x to be $L(V(x))$. For a module $\mathcal{M}$ and a formula φ of the enriched μ–calculus we say that $\mathcal{M}$ satisfies φ, denoted $\mathcal{M} \models_r \varphi$, if all quasi-forests in $exec(\mathcal{M})$ satisfy φ. The problem of deciding whether $\mathcal{M}$ satisfies φ is called *enriched μ–calculus module checking*.

Open Pushdown Systems. An *OPD* over $AP \cup Nom \cup Prog$ is a tuple $\mathcal{S} = \langle Q, \Gamma, \flat, C_0, \Delta, \rho_1, \rho_2, Env \rangle$, where Q is a finite set of (control) *states*, Γ is a finite *stack alphabet*, $\flat \notin \Gamma$ is the *stack bottom symbol*. We set $\Gamma_\flat = \Gamma \cup \{\flat\}$, $Conf = Q \times (\Gamma^* \cdot \flat)$ to be the set of *(pushdown) configurations* and for each configuration $(q, A \cdot \gamma)$ we set $top((q, A \cdot \gamma)) = (q, A)$ to be a *top configuration*. The function $\Delta : Prog \to 2^{(Q \times \Gamma_\flat) \times (Q \times \{push, pop, sub\} \times \Gamma)}$ is a finite set of transition rules. The set $C_0 \subseteq Conf$ is a finite set of *initial configurations*, $\rho_1 : AP \to 2^{Q \times \Gamma_\flat}$ is a labeling function, which associates to each atomic proposition p a set of top configurations in which p holds, $\rho_2 : Nom \to C_0$ is a labeling function, which associates to each nominal exactly one initial configuration, and $Env \subseteq Q \times \Gamma_\flat$ is specifies the set of *environment configurations*. The size $|\mathcal{S}|$ of $\mathcal{S}$ is $|Q| + |\Delta| + |\Gamma|$.

The *OPD* moves in accordance with the transition relation Δ. Thus, $((q, A), (q', op, B)) \in \Delta(\alpha)$ implies that if the *OPD* is in state q and the top of the pushdown store is A, it can move along with an *α–transition* to state q', and either delete A if $op = pop$, or insert B on the top of the stack if $op = push$, or substitute A with B if $op = sub$. We assume that if $\flat$ is popped it gets pushed right back, and that it only gets pushed in such cases. Thus, $\flat$ is always present

at the bottom of the pushdown store and nowhere else. Note that we make this assumption also about the various pushdown automata we use later. Also note that the possible operations of the system, the labeling functions, and the designation of configurations as environment configurations, are all dependent only on the current control state and the top of the pushdown store.

An *OPD* $\mathcal{S}$ induces a module $\mathcal{M}_{\mathcal{S}} = \langle W_s, W_e, W_0, R, L\rangle$, where:

- $W_s \cup W_e = Conf$, i.e., the set of pushdown configurations, and $W_0 = C_0$;
- $W_e = \{c \in Conf \mid top(c) \in Env\}$.
- $((q, A \cdot \gamma), (q', \gamma')) \in R(\alpha)$ *iff* there exists $((q, A), (q', op, B)) \in \Delta(\alpha)$ such that either $op = pop$ and $\gamma' = \gamma$, or $op = push$ and $\gamma' = BA \cdot \gamma$, or $op = sub$ and $\gamma' = B \cdot \gamma$;
- $L(p) = \{c \in Conf \mid top(c) \in \rho_1(p)\}$ for $p \in AP$; $L(o) = \rho_2(o)$ for $o \in Nom$.

The *enriched (μ-calculus) pushdown module checking* problem is to decide, for a given *OPD* $\mathcal{S}$ and an enriched μ–calculus formula ψ, whether $\mathcal{M}_{\mathcal{S}} \models_r \psi$.

3 Tree Automata

Two-way Graded Alternating Parity Tree Automata ($2GAPT$). These automata are an extension of nondeterministic tree automata that can send several copies of itself to the same successor (alternating), send copies of itself to the predecessor (two-way), specify a number n of successors to which copies of itself are sent without specifying which successors these exactly are (graded), and accept trees along with a parity condition [BLMV06]. To give a formal definition, let us start with some technicalities.

For a given set Y, let $B^+(Y)$ be the set of positive Boolean formulas over Y (i.e., Boolean formulas built from elements in Y using $\wedge$ and $\vee$), where we also allow the formulas **true** and **false** and $\wedge$ has precedence over $\vee$. For a set $X \subseteq Y$ and a formula $\theta \in B^+(Y)$, we say that X satisfies θ iff assigning **true** to elements in X and assigning **false** to elements in $Y \setminus X$ makes θ **true**. For $b > 0$, let $\langle [b]\rangle = \{\langle 0\rangle, \langle 1\rangle, \ldots, \langle b\rangle\}$, $[[b]] = \{[0], [1], \ldots, [b]\}$, and $D_b = \langle [b]\rangle \cup [[b]] \cup \{-1, \varepsilon\}$.

Formally, a $2GAPT$ on Σ-labeled trees is a tuple $\mathcal{A} = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$, where Σ is the input alphabet, $b > 0$ is a counting bound, Q is a finite set of states, $\delta : Q \times \Sigma \to B^+(D_b \times Q)$ is a transition function, $q_0 \in Q$ is an initial state, and $\mathcal{F}$ is a parity acceptance condition (see below). Intuitively, an atom $(\langle n\rangle, q)$ (resp. $([n], q)$) means that $\mathcal{A}$ sends copies in state q to $n + 1$ (resp. all but n) different successors of the current node, (ε, q) means that $\mathcal{A}$ sends a copy (in state q) to the current node, and $(-1, q)$ means that $\mathcal{A}$ sends a copy to the predecessor of the current node. A *run* of $\mathcal{A}$ on an input Σ-labeled tree $\langle T, V\rangle$ is a tree $\langle T_r, r\rangle$ in which each node is labeled by an element of $T \times Q$. Intuitively, a node in T_r labeled by (x, q) describes a copy of the automaton in state q that reads the node x of T. Runs start in the initial state and satisfy the transition relation. Thus, a run $\langle T_r, r\rangle$ with root z has to satisfy the following: (i) $r(z) = (1, q_0)$ for the root 1 of T and (ii) for all $y \in T_r$ with $r(y) = (x, q)$ and $\delta(q, V(x)) = \theta$, there is a (possibly empty) set $S \subseteq D_b \times Q$, such that S satisfies θ, and for all $(d, s) \in S$, the following hold:

- If $d \in \{-1, \varepsilon\}$, then $x \cdot d$ is defined, and there is $j \in \mathbb{N}$ such that $y \cdot j \in T_r$ and $r(y \cdot j) = (x \cdot d, s)$;
- If $d = \langle n \rangle$ (resp., $d = [n]$), there are distinct $i_1, \ldots, i_t$, with $t = n + 1$ (resp., $t = deg(x) - n$) such that for all $1 \leq j \leq t$, there is $j' \in \mathbb{N}$ such that $y \cdot j' \in T_r$, $x \cdot i_j \in T$, and $r(y \cdot j') = (x \cdot i_j, s)$;

Note that if $\theta = \mathbf{true}$, then y does not need to have successors. This is the reason why T_r may have leaves. Also, since there exists no set S as required for $\theta = \mathbf{false}$, we cannot have a run that takes a transition with $\theta = \mathbf{false}$.

A run $\langle T_r, r \rangle$ is *accepting* if all its infinite paths satisfy the acceptance condition. In the parity acceptance condition, $\mathcal{F}$ is a set $\{F_1, \ldots, F_k\}$ such that $F_1 \subseteq \ldots \subseteq F_k = Q$ and k is called the *index* of the automaton. An infinite path π on T_r *satisfies* $\mathcal{F}$ if there is an even i such that π contains infinitely many states from F_i and finitely many states from F_{i-1}. An automaton *accepts* a tree iff there exists an accepting run of the automaton on the tree. We denote by $\mathcal{L}(\mathcal{A})$ the set of all Σ-labeled tree that $\mathcal{A}$ accepts.

A $2GAPT$ is a $2APT$ if $D_b = \{-1, \varepsilon, 1, \ldots, h\}$ [Var98]. Moreover, it is an NPT if $D_b = \{1, \ldots, h\}$ and the transition relation δ is in disjunctive normal form, where in each conjunct each direction appears at most once [KVW00].

We now recall a result on $2GAPT$ and enriched μ-calculus formulas.

Lemma 1 ([BLMV06,SV01]). *Given an enriched μ-calculus sentence φ that has ℓ* atleast *subsentences and counts up to b, we can built a* 2GAPT *with $\mathcal{O}(|\varphi|^2)$ states, index $|\varphi|$, and counting bound b that accepts exactly the trees encodings of the quasi-forest models of φ having degree at most* $\max\{|Nom| + 1, \ell(b+1)\}$.

Nondeterministic Pushdown Parity Tree Automata. A *PD–NPT* (without ε-transitions), on Σ-labeled full h-ary trees, is a tuple $\mathcal{P} = \langle \Sigma, \Gamma, \flat, Q, q_0, \gamma_0, \rho, \mathcal{F} \rangle$, where Σ is a finite input alphabet, Γ, $\flat$, $\Gamma_\flat$, Q, q_0, and γ_0 are as in *OPD*, $\rho : Q \times \Sigma \times \Gamma_\flat \to 2^{(Q \times \{push, pop, sub\} \times \Gamma)^h}$ is a transition function, and $\mathcal{F}$ is a parity condition over Q. We define $push(\gamma, B) = B \cdots \gamma$, $pop(A \cdot \gamma, B) = \gamma$, and $sub(A \cdot \gamma, B) = B \cdot \gamma$. Intuitively, when $\mathcal{P}$ is in state q, reading an input node x labeled by $\sigma \in \Sigma$, and the stack contains a word $A \cdot \gamma$ in $\Gamma^* \cdot \flat$, then $\mathcal{P}$ chooses a tuple $\langle (q_1, op_1, B_1), \ldots, (q_h, op_h, B_h) \rangle \in \rho(q, \sigma, A)$ and splits in h copies such that for each $1 \leq i \leq h$, a copy in state q_i, and stack content obtained by upgrading the stack accordingly with op_i and B_i (as for *OPD*) is sent to the node $x \cdot i$ in the input tree. A run of $\mathcal{P}$ on a Σ-labeled full h-ary tree $\langle T, V \rangle$ is a $(Q \times \Gamma^* \cdot \flat)$-labeled tree $\langle T, r \rangle$ such that $r(\varepsilon) = (q_0, \gamma_0)$ and for each $x \in T$ with $r(x) = (q, A \cdot \gamma)$, there is $\langle (q_1, op_1, B_1), \ldots, (q_h, op_h, B_h) \rangle \in \rho(q, V(x), A)$ where, for all $1 \leq i \leq h$, $r(x \cdot i) = (q_i, op_i(A \cdot \gamma, B_i))$. The notion of accepting path is defined with respect to the control states that appear infinitely often in the path (thus without taking into account any stack content). Then the notions given for $2GAPT$ regarding accepting runs, accepted trees, and accepted languages, along with the parity condition, easily extend to *PD–NPT*. We also consider Büchi condition $\mathcal{F} \subseteq Q$, which simply is a special parity condition $\{\emptyset, \mathcal{F}, Q\}$. In the following, we denote with *PD–NBT* a *PD–NPT* with a Büchi condition. The *emptiness* problem for an automaton $\mathcal{P}$ is to decide whether $\mathcal{L}(\mathcal{P}) = \emptyset$.

We now recall two useful results on the introduced automata.

Proposition 1 ([KPV02]). *The emptiness problem for a* PD–NPT *on Σ-labeled full h-ary trees, having index m, n states, and transition function ρ, can be solved in time exponential in $n \cdot m \cdot h \cdot |\rho|$.*

Proposition 2 ([BMP05]). *On Σ-labeled full h-ary trees, given a* PD–NBT $\mathcal{P} = \langle \Sigma, \Gamma, Q, q_0, \gamma_0, \rho, Q \rangle$ *and an* NPT $\mathcal{A} = \langle \Sigma, Q', q'_0, \delta, \mathcal{F}' \rangle$*, there is a* PD–NPT $\mathcal{P}'$ *such that $\mathcal{L}(\mathcal{P}') = \mathcal{L}(\mathcal{P}) \cap \mathcal{L}(\mathcal{A})$. Moreover, $\mathcal{P}'$ has $|Q| \cdot |Q'|$ states, the same index as $\mathcal{A}$, and the size of the transition relation is bounded by $|\rho| \cdot |\delta| \cdot h$.*

4 Deciding Enriched Pushdown Module Checking

In this section, we show that enriched pushdown module checking is decidable and solvable in 2EXPTIME. Since *CTL* pushdown module checking is 2EXPTIME-hard, we get that the addressed problem is 2EXPTIME-complete. For the upper bound, the algorithm works as follows. Given an *OPD* $\mathcal{S}$, and a module $\mathcal{M}_\mathcal{S}$ induced by $\mathcal{S}$, by combining and extending the constructions given in [BMP05] and [FM07], we first build in polynomial-time a *PD–NBT* $\mathcal{A}_\mathcal{S}$ accepting all trees encoding of all quasi-forests belonging to $exec(\mathcal{M}_\mathcal{S})$. Then, given an enriched μ-calculus formula φ, accordingly to [SV01,BLMV06], we can build in polynomial-time a $2GAPT$ $\mathcal{A}_{\neg\varphi}$ (Lemma 1) accepting all models of $\neg\varphi$, with the intent to check that no models of $\neg\varphi$ are in[2] $exec(\mathcal{M}_\mathcal{S})$. Then, accordingly to the basic idea of [KVW01], we check that $\mathcal{M} \models_r \varphi$ by checking whether $\mathcal{L}(\mathcal{A}_\mathcal{S}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi})$ is empty. Finally, we get the result by using an EXPTIME translation of the latter to the emptiness problem for *PD–NPT*. As a key step here, we show a nontrivial EXPTIME reduction of $2GAPT$ into *NPT*. This exponentially improves the translation algorithm known from the literature and it is turns to be also useful to get simplified decision procedures regarding enriched μ-calculus, such as satisfiability [BLMV06] and finite-state module checking [FM07].

Let us start dealing with $\mathcal{A}_\mathcal{S}$. Before building the automaton, there are some technical difficulties to overcome, which are also new with respect to [BMP05]. First note that $\mathcal{A}_\mathcal{S}$, as being a *PD–NBT*, deals with trees having labels only on nodes, while $exec(\mathcal{M})$ contains quasi-forests with both edges and nodes labeled. To solve this problem, for each quasi-forest in $exec(\mathcal{M})$, the automaton $\mathcal{A}_\mathcal{S}$ accepts a corresponding encoding tree obtained by *(i)* adding a new root connecting all roots of the quasi-forest, *(ii)* moving the label of each edge to the target node of the edge (using a new atomic proposition p_α, for each program α), and *(iii)* substituting "jumps to roots" with new atomic propositions $\uparrow_o^\alpha$ (representing an α-labeled edge from the current node to the unique root node labeled by nominal o). Let $AP^* = AP \cup \{p_\alpha \mid \alpha \text{ is a program}\} \cup \{\uparrow_o^\alpha \mid \alpha \text{ is a program and } o \text{ is a nominal}\}$, we denote with $\langle T, V^* \rangle$ the $2^{AP^* \cup Nom}$-labeled tree *encoding* of a quasi-forest $\langle F, V, E \rangle \in exec(\mathcal{M})$, obtained using the above transformations.

Another technical difficulty to handle with is related to the fact that quasi-forests of $exec(\mathcal{M})$ (and thus their encoding) may not be full h-ary, since the

[2] For a better readability, in the rest of the paper we will use $\mathcal{M}$ instead of $\mathcal{M}_\mathcal{S}$.

OPD from which $\mathcal{M}$ is induced nodes may have different degrees. Moreover, quasi-forests of $exec(\mathcal{M})$ may not share the same structure, since they are obtained by pruning some subtrees from the computation quasi-forest $\langle F_{\mathcal{M}}, V_{\mathcal{M}}, E_{\mathcal{M}}\rangle$ of $\mathcal{M}$. Let $\langle T_{\mathcal{M}}, V^*_{\mathcal{M}}\rangle$ be the h-ary *computation tree* of $\mathcal{M}$ obtained from $\langle F_{\mathcal{M}}, V_{\mathcal{M}}, E_{\mathcal{M}}\rangle$ using the above encoding. By extending an idea of [KVW01,BMP05], we consider each tree $\langle T, V^*\rangle$, encoding of a quasi-forest $\langle F, V, E\rangle$ of $exec(\mathcal{M})$, as a $2^{AP^*\cup Nom\cup\{t\}} \cup \{\bot\}$-labeled full h-ary tree $\langle T_{\mathcal{M}}, V^{**}\rangle$ (where $\bot$ and t are fresh proposition names not belonging to $AP^* \cup Nom$) in the following way: first we add proposition t to the label of all leaf nodes of the forest; second, for each node $x \in T_{\mathcal{M}}$ with p children $x \cdot 1, \ldots, x \cdot p$ (note that $0 \leq p \leq h$), we add the children $x \cdot (p+1), \ldots, x \cdot h$ and label these new nodes with $\bot$; finally, for each node x labeled by $\bot$ we add recursively h children labeled by $\bot$. Thus, for each node $x \in T_{\mathcal{M}}\backslash\{root(T_{\mathcal{M}}\}$, if $x \in F$ then $V^{**}(x) = V^*(x)$, otherwise $V^{**}(x) = \{\bot\}$ and therefore the proposition $\bot$ is used to denote both "disabled" states and "completion" states. In this way, all trees encoding quasi-forests of $exec(\mathcal{M})$ are full h-ary trees, and they differ only in their labeling. Moreover, the environment can also disable jumps to roots. This is performed by removing from enabled environment nodes some of $\uparrow^{\alpha}_{o}$ labels. Notice that since we consider environments that do not block the system, each node, which corresponds to a root in the quasi–forest and is associated with an environment state, has at least one successor not labeled by $\{\bot\}$, unless it has $\uparrow^{\alpha}_{o}$ in its label. Putting in practice the construction proposed above, we obtain the following result, where $\widehat{exec}(\mathcal{M})$ the set of all $2^{AP^*\cup Nom\cup\{t\}} \cup \{\bot\}$-labeled full h-ary trees obtained from $T_{\mathcal{M}}, V^*_{\mathcal{M}}$ in the above described manner (the detailed construction is reported in the full version which can be found at the url `http://people.na.infn.it/~murano/pubblicazioni/EPD-full.pdf`).

Lemma 2. *Given an* OPD $\mathcal{S} = \langle Q, \Gamma, \flat, C_0, \Delta, \rho_1, \rho_2, Env\rangle$ *with branching degree* h*, we can build a* PD–NBT $\mathcal{A}_{\mathcal{S}} = \langle \Sigma, \Gamma, \flat, Q', q'_0, \gamma_0, \delta, Q\rangle$*, which accepts exactly* $\widehat{exec}(\mathcal{M})$*, such that* $|\Sigma| = 2^{AP^*\cup Nom}$*,* $|Q'| = O(|Q|^2 \cdot |\Gamma|)$*, and* $|\delta|$ *is polynomially bounded by* $h \cdot |\Delta|$*.*

Let us now go back to the enriched μ-calculus formula φ. We recall by Lemma 1 that we can build in polynomial-time a $2GAPT$ $\mathcal{A}_{\neg\varphi}$ accepting all models of $\neg\varphi$. As we have pointed out before, if we show an exponential-time translation of $\mathcal{A}_{\neg\varphi}$ into an NPT, we get the desired 2Exptime upper bound for the problem we are investigating. Indeed, by Proposition 2, we have that $\mathcal{L}(\mathcal{A}_{\mathcal{S}}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi})$ can be accepted by a *PD–NPT* whose size is exponential in the size of φ and polynomial in the size of $\mathcal{S}$. Hence, we get the result by using the fact that the emptiness problem for *PD–NPT* can be checked in exponential-time (Proposition 1). The remaining part of this section is devoted to show how to translate in Exptime a $2GAPT$ into an NPT. The translation we propose uses the notions of *strategies*, *promises* and *annotations*, which we now recall.

Let $\mathcal{A} = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$ be a $2GAPT$ with $\mathcal{F} = \langle F_1, \ldots, F_k\rangle$ and $\langle T, V\rangle$ be a Σ-labeled tree. Recall that $D_b = \langle [b]\rangle \cup [[b]] \cup \{-1, \varepsilon\}$ and $\delta : (Q \times \Sigma) \to B^+(D_b \times Q)$. For each control state $q \in Q$, let $index(q)$ be the minimal i such

that $q \in F_i$. A *strategy tree* for $\mathcal{A}$ on $\langle T, V\rangle$ is a $2^{Q\times D_b\times Q}$-labeled tree $\langle T, \mathsf{str}\rangle$ such that, defined $head(w) = \{q : (q, d, q') \in w\}$ as the set of *sources* of w, it holds that (i) $q_0 \in head(\mathsf{str}(root(T)))$ and (ii) for each node $x \in T$ and state q, the set $\{(q, q') : (q, d, q') \in \mathsf{str}(x)\}$ satisfies $\delta(q, V(x))$.

A *promise tree* for $\mathcal{A}$ on $\langle T, V\rangle$ is a $2^{Q\times Q}$-labeled tree $\langle T, \mathsf{pro}\rangle$. We say that pro *fulfills* str for V if the states promised to be visited by pro satisfy the obligations induced by str as it runs on V. Formally, pro fulfills str for V if for every node $x \in T$, the following hold: "for every $(q, \langle n\rangle, q') \in \mathsf{str}(x)$ (resp. $(q, [n], q') \in \mathsf{str}(x)$), at least $n + 1$ (resp. $deg(x) - n$)) successors $x \cdot j$ of x have $(q, q') \in \mathsf{pro}(x \cdot j)$".

An *annotation tree* for $\mathcal{A}$ on $\langle T, \mathsf{str}\rangle$ and $\langle T, \mathsf{pro}\rangle$ is a $2^{Q\times\{1,\ldots,k\}\times Q}$-labeled tree $\langle T, \mathsf{ann}\rangle$ such that for every node $x \in T$ and $(q, d_1, q_1) \in \mathsf{str}(x)$ the following conditions hold:

- if $d_1 = \varepsilon$ then $(q, index(q_1), q_1) \in \mathsf{ann}(x)$;
- if $d_1 \in \{1, \ldots, k\}$ then for all $d_2 \in \{1, \ldots, k\}$ and $q_2 \in Q$ such that $(q_1, d_2, q_2) \in \mathsf{ann}(x)$, we have $(q, \min(d_1, d_2), q_2) \in \mathsf{ann}(x)$;
- if $d_1 = -1$ and $x = y \cdot i$, then for all $d_2, d_3 \in \{1, \ldots, k\}$ and $q_2, q_3 \in Q$ such that $(q_1, d_2, q_2) \in \mathsf{ann}(y)$, $(q_2, d_3, q_3) \in \mathsf{str}(y)$ and $(q_2, q_3) \in \mathsf{pro}(x)$, holds that $(t, \min(index(q_1), d_2, index(q_3)), q_3) \in \mathsf{ann}(x)$;
- if $d_1 \in [[b]] \cup [\langle b\rangle]$, $y = x \cdot i$ and $(q, q_1) \in \mathsf{pro}(y)$, then for all $d_2, d_3 \in \{1, \ldots, k\}$ and $q_2, q_3 \in Q$ such that $(q_1, d_2, q_2) \in \mathsf{ann}(y)$ and $(q_2, -1, q_3) \in \mathsf{str}(y)$, holds that $(t, \min(index(q_1), d_2, index(q_3)), q_3) \in \mathsf{ann}(x)$.

A downward path induced by str, pro and ann on $\langle T, V\rangle$ is a sequence $\langle x_0, q_0, t_0\rangle, \langle x_1, q_1, t_1\rangle, \ldots$ such that $x_0 = root(T)$, q_0 is the initial state of $\mathcal{A}$ and for each $i \geq 0$ holds that $x_i \in T$, $q_i \in Q$ and $t_i = \langle q_i, d, q_{i+1}\rangle \in \mathsf{str}(x_i) \cup \mathsf{ann}(x_i)$ is such that either *(i)* $d \in \{1, \ldots, k\}$ and $x_{i+1} = x_i$, or *(ii)* $d \in \langle [b]\rangle \cup [[b]]$ and there exists $c \in \{1, \ldots, deg(x_i)\}$ such that $x_{i+1} = x_i \cdot c$ and $(q_i, q_{i+1}) \in \mathsf{pro}(x_{i+1})$. In the first case we set $index(t_i) = d$ and in the second case we set $index(t_i) = \min\{j \in \{1, \ldots, k\} \mid q_{i+1} \in F_j\}$. Moreover, for a downward path π we set $index(\pi)$ as the minimum index that appears infinitely often in π and we say that π is *accepting* if $index(\pi)$ is even.

The following lemma relates the language accepted by a $2GAPT$ with strategies, promises and annotations.

Lemma 3 ([BLMV06]). *Let $\mathcal{A}$ be a* 2GAPT*. A Σ-labeled tree $\langle T, V\rangle$ is accepted by $\mathcal{A}$ iff there exist a strategy tree $\langle T, \mathsf{str}\rangle$ and a promise tree $\langle T, \mathsf{pro}\rangle$ for $\mathcal{A}$ on $\langle T, V\rangle$ such that* pro *fulfills* str *for V, and an annotation tree $\langle T, \mathsf{ann}\rangle$ for $\mathcal{A}$ on $\langle T, V\rangle$, $\langle T, \mathsf{str}\rangle$ and $\langle T, \mathsf{pro}\rangle$ such that every downward path induced by* str*,* pro *and* ann *on $\langle T, V\rangle$ is accepting.*

The above lemma has been used in [BLMV06] to show how to translate in exponential-time a $2GAPT$ into a $GNPT$. In the next lemma, we improve this result on by showing an exponential-time translation of a $2GAPT$ into an NPT.

Given an alphabet Σ for the input tree of a $2GAPT$ with transition function δ, let D_b^δ be the subset containing only the elements of D_b appearing in δ. Then we denote by Σ' the extended alphabet for the combined trees, i.e., $\Sigma' = \Sigma \times 2^{Q\times D_b^\delta\times Q} \times 2^{Q\times Q} \times 2^{Q\times\{1,\ldots k\}\times Q}$.

Lemma 4. *Let $\mathcal{A}$ be a* 2GAPT *running on Σ–labeled trees with n states, index k and counting bound b that accepts h-ary trees. We can build in* EXPTIME *an* NPT $\mathcal{A}'$ *running on Σ'–labeled h-ary trees that accepts a tree iff $\mathcal{A}$ accepts its projection on Σ.*

Proof. Let $\mathcal{A} = \langle \Sigma, b, Q, q_0, \delta, \mathcal{F} \rangle$ with $\mathcal{F} = \langle F_1, \ldots, F_k \rangle$. By Lemma 3, we build $\mathcal{A}'$ as the intersection of three *NPT*s $\mathcal{A}'$, $\mathcal{A}''$, and $\mathcal{A}'''$, each having size exponential in the size of $\mathcal{A}$, such that

- $\mathcal{A}'$ accepts a Σ'-labeled tree $\langle T, (V, \mathsf{str}, \mathsf{pro}, \mathsf{ann}) \rangle$ iff str is a strategy for $\mathcal{A}$ on $\langle T, V \rangle$ and pro fulfills str for V,
- $\mathcal{A}''$ accepts a Σ'-labeled tree $\langle T, (V, \mathsf{str}, \mathsf{pro}, \mathsf{ann}) \rangle$ iff ann is an annotation for $\mathcal{A}$ on $\langle T, V \rangle$, $\langle T, \mathsf{str} \rangle$ and $\langle T, \mathsf{pro} \rangle$, and
- $\mathcal{A}'''$ accepts a Σ'-labeled tree $\langle T, (V, \mathsf{str}, \mathsf{pro}, \mathsf{ann}) \rangle$ iff every downward path induced by str, pro and ann on $\langle T, V \rangle$ is accepting.

The automaton $\mathcal{A}' = \langle \Sigma', Q', q_0', \delta', \mathcal{F}' \rangle$ works as follows: on reading a node x labeled $(\sigma, \eta, \rho, \omega)$, then it locally checks whether η satisfies the definition of strategy for $\mathcal{A}$ on $\langle T, V \rangle$. In particular, when $\mathcal{A}'$ is in its initial state, we check that η contains a transition starting from the initial state of $\mathcal{A}$. Moreover, the automaton $\mathcal{A}'$ sends to each child $x \cdot i$ the pairs of states that have to be contained in $\mathsf{pro}(x \cdot i)$, in order to verify that pro fulfills str. To obtain this, we set $Q' = 2^{Q \times Q} \cup \{q_0'\}$ and $\mathcal{F}' = \{\emptyset, Q'\}$. To define δ', we first give the following definition. For each node $x \in T$ labeled $(\sigma, \eta, \rho, \omega)$, we set

$$
\begin{aligned}
S(\eta) = \{ & \langle S_1, \ldots, S_{deg(x)} \rangle \in (2^{Q \times Q})^{deg(x)} \text{ such that} \\
& [\text{for each } (q, \langle m \rangle, p) \in \eta \text{ there exist distinct } i_1, \ldots, i_{m+1} \text{ such that} \\
& (q, p) \in S_{i_j}, \text{ for all } j \in \{1, \ldots, m+1\}] \text{ and} \\
& [\text{for each } (q, [m], p) \in \eta \text{ there exist distinct } i_1, \ldots, i_{deg(x)-m} \text{ such that} \\
& (q, p) \in S_{i_j}, \text{ for all } j \in \{1, \ldots, deg(x) - m\}]\}
\end{aligned}
$$

to be the set of all tuples with size $deg(x)$, each *fulfilling* all the graded modalities in $\mathsf{str}(x)$. Notice that $|S(\eta)| \leq 2^{hn^2}$. Then we have

$$
\delta'(q, (\sigma, \eta, \rho, \omega)) = \begin{cases} S(\eta) & \text{if } \forall\ p \in \mathit{head}(\eta),\ \{(d, p') \mid (p, d, p') \in \eta\} \text{ satisfies } \delta(p, \sigma) \\ & \text{and } [(q = q_0^1 \text{ and } q_0 \in \mathit{head}(\eta)) \text{ or } (q \neq q_0^1 \text{ and } q \subseteq \rho)] \\ \mathbf{false} & \text{otherwise,} \end{cases}
$$

Hence, in $\mathcal{A}'$ we have $|Q'| = 2^{n^2}$, $|\delta'| \leq 2^{n^2(k+1)}$, and index 2.

$\mathcal{A}'' = \langle \Sigma', Q'', q_0'', \delta'', \mathcal{F}'' \rangle$ works in a similar way to $\mathcal{A}'$. That is, for each node x, it first locally checks whether the constraints of the annotations are verified; then it sends to the children of x the strategy and annotation associated with x, in order to successively verify whether the promises associated with the children nodes are consistent with the annotation of x. Therefore, in the automaton A'' we have $Q'' = 2^{Q \times D_b^\delta \times Q} \times 2^{Q \times \{1, \ldots, k\} \times Q}$, $q_0'' = (\emptyset, \emptyset)$, $\mathcal{F}'' = \{\emptyset, Q''\}$, and for a state $(\eta_{prev}, \omega_{prev})$ and a letter $(\sigma, \eta, \rho, \omega)$ we have

$$
\delta''((\eta_{prev}, \omega_{prev}), (\sigma, \eta, \rho, \omega)) = \begin{cases} \langle (\eta, \omega), \ldots, (\eta, \omega) \rangle & \text{if the local conditions for the} \\ & \text{annotations are verified} \\ \mathbf{false} & \text{otherwise.} \end{cases}
$$

Hence, in $\mathcal{A}''$ we have $|Q''| \leq 2^{n^2(|\delta|+k)}$, $|\delta''| \leq h \cdot 2^{n^2(|\delta|+k)}$, and index 2.

Finally, to define the automaton $\mathcal{A}'''$ we start by constructing a $2APT$ $\mathcal{B}$ polynomial in the size of $\mathcal{A}$ that accepts $\langle T, (V, \mathsf{str}, \mathsf{pro}, \mathsf{ann})\rangle$ iff there is a downward path induced by str, pro and ann on $\langle T, V\rangle$ that is not accepting. The automaton $\mathcal{B} = \langle \Sigma', Q^B, q_0^B, \delta^B, \mathcal{F}^B\rangle$ (which in particular does not need direction -1) essentially chooses, in each state, the downward path to walk on, and uses an integer to store the index of the state. We use a special state $\sharp$ not belonging to Q to indicate that $\mathcal{B}$ is proceeding in accordance with an annotation instead of a strategy. Therefore the set of states is $Q^B = ((Q \cup \{\sharp\}) \times \{1, \ldots, k\} \times Q) \cup \{q_0^B\}$.

To define the transition function on a node x, let us introduce a function f that for each $q \in Q$, strategy $\eta \in 2^{Q \times D_b^\delta \times Q}$, and annotation $\omega \in 2^{Q \times \{1,\ldots,k\} \times Q}$ gives a formula satisfied along downward paths consistent with η and ω. That is in each node x, the function f either proceeds according to the annotation ω or the strategy η (note that f does not check that the downward paths is consistent with any promise). Formally, f is defined as follows:

$$f(q,\eta,\omega) = \bigvee_{\substack{(q,d,p)\in\omega \\ d\in\{1,\ldots,k\}}} \langle \varepsilon, (\sharp, d, p)\rangle \quad \vee \quad \bigvee_{\substack{(q,d,p)\in\eta \\ d\in\langle[b]\rangle\cup[[b]]}} \bigvee_{c\in\{1,\ldots,deg(x)\}} \langle c, (q, index(p), p)\rangle$$

where $index(p)$ is the minimum i such that $p \in F_i$. Then we have

$$\delta^B(q_0^B, (\sigma,\eta,\rho,\omega)) = f(q_0,\eta,\omega)$$

$$\delta^B((q,d,p),(\sigma,\eta,\rho,\omega)) = \begin{cases} \mathbf{false} & \text{if } q \neq \sharp \text{ and } (q,p) \notin \rho \\ f(p,\eta,\omega) & \text{otherwise.} \end{cases}.$$

A downward path is accepted if the minimum index that appears infinitely often in the path is odd. Therefore, $\mathcal{F}^B = \langle F_1^B, \ldots, F_{k+1}^B, Q^B\rangle$ where $F_1^B = \emptyset$ and, for all $i \in \{2, \ldots, k+1\}$, $F_i^B = \{(q,d,p) \in Q^B \mid d = i-1\}$. Thus, $|Q^B| = kn(n+1)+1$, $|\delta^B| = k \cdot |\delta| \cdot |Q^B|$ and the index is $k+2$. Then, since $\mathcal{B}$ is alternating, we easily complement it in polynomial time into a resulting $2APT$ $\overline{\mathcal{B}}$ accepting a tree iff all the downward paths induced by str, pro and ann on $\langle T, V\rangle$ are accepting. Finally, following [Var98] we can build in exponential-time the desired automaton $\mathcal{A}'''$. □

Using the above lemma, along with Lemma 1, we get that given an enriched μ-calculus formula φ, we can build in exponential-time an NPT $\mathcal{A}_{\neg\varphi}$ accepting all models of $\neg\varphi$. Now, recall that given a module $\mathcal{M}$ induced by an OPD S and the automaton $\mathcal{A}_\mathcal{S}$ accepting all trees encoding of quasi-forests belonging to $exec(\mathcal{M})$ (see Lemma 2), it is possible to check whether $\mathcal{M} \models_r \varphi$ by checking whether $\mathcal{L}(\mathcal{A}_\mathcal{S}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi})$ is empty. Also, recall that by Proposition 2, $\mathcal{L}(\mathcal{A}_\mathcal{S}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi})$ can be accepted by a PD–NPT $\mathcal{A}$ whose size is exponential in the size of φ and polynomial in the size of $\mathcal{S}$. Finally, by recalling that the emptiness of $\mathcal{A}$ can be checked in Exptime (Proposition 1) and that the pushdown module checking problem for CTL is 2Exptime-hard with respect to the size of the formula and Exptime-hard in the size of the system [BMP05], we get the following result.

Theorem 1. *The pushdown module checking problem for enriched μ–calculus is* 2EXPTIME*–complete w.r.t. the size of the formula and* EXPTIME*–complete w.r.t. the size of the system.*

References

[BLMV06] P.A. Bonatti, C. Lutz, A. Murano, and M.Y. Vardi, *The complexity of enriched μ-calculi*, ICALP'06, LNCS 4052, 2006, pp. 540–551.

[BMP05] Laura Bozzelli, Aniello Murano, and Adriano Peron, *Pushdown module checking.*, LPAR, 2005, pp. 504–518.

[BP04] P.A. Bonatti and A. Peron, *On the undecidability of logics with converse, nominals, recursion and counting*, Artif. Intelligence **158 : 1** (2004), 75–96.

[CE81] E.M. Clarke and E.A. Emerson, *Design and synthesis of synchronization skeletons using branching time temporal logic*, Proc. of Work. on Logic of Programs, LNCS 131, 1981, pp. 52–71.

[CGP99] E.M. Clarke, O. Grumberg, and D.A. Peled, *Model checking*, MIT Press, Cambridge, MA, USA, 1999.

[FM07] A. Ferrante and A. Murano, *Enriched μ-calculus module checking*, FOSSACS'07, LNCS 4423, 2007, p. 183197.

[Hoa85] C.A.R. Hoare, *Communicating sequential processes*, Prentice-Hall International. Upper Saddle River, NJ, USA, 1985.

[HP85] D. Harel and A. Pnueli, *On the development of reactive systems*, Logics and Models of Concurrent Systems, NATO Advanced Summer Institutes, vol. F-13, Springer-Verlag, 1985, pp. 477–498.

[KG96] Orna Kupferman and Orna Grumberg, *Buy one, get one free!!!*, J. Log. Comput. **6** (1996), no. 4, 523–539.

[Koz83] D. Kozen, *Results on the propositional mu-calculus.*, Theoretical Computer Science **27** (1983), 333–354.

[KPV02] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi, *Pushdown specifications.*, LPAR, 2002, pp. 262–277.

[KSV02] O. Kupferman, U. Sattler, and M.Y. Vardi, *The complexity of the graded μ-calculus*, CADE'02, LNAI 2392, 2002, pp. 423–437.

[KVW00] O. Kupferman, M.Y. Vardi, and P. Wolper, *An automata-theoretic approach to branching-time model checking*, J. of ACM **47** (2000), no. 2, 312–360.

[KVW01] ———, *Module checking*, Information & Computation **164** (2001), 322–344.

[QS81] J.P. Queille and J. Sifakis, *Specification and verification of concurrent systems in cesar*, 5^{th} Symp. on Programming, LNCS 137, 1981, pp. 337–351.

[Sal96] Kai Salomaa, *Yield-languages of two-way pushdown tree automata.*, Inf. Process. Lett. **58** (1996), no. 4, 195–199.

[SV01] U. Sattler and M.Y. Vardi, *The hybrid mu-calculus*, IJCAR'01, LNAI 2083, 2001, pp. 76–91.

[Var98] M.Y. Vardi, *Reasoning about the past with two-way automata*, ICALP'98, LNCS 1443, 1998, pp. 628–641.

A Proof of Lemma 2

Let $\mathcal{S} = \langle Q, \Gamma, \flat, C_0, \Delta, \rho_1, \rho_2, Env \rangle$ be an *OPD*. We construct a parity *PD–NBT* $\mathcal{A}_{\mathcal{S}}$ that accepts the trees in $\widehat{exec}(\mathcal{M}_{\mathcal{S}})$. As explained in section 4, $\widehat{exec}(\mathcal{M}_{\mathcal{S}})$

only contains full h-ary trees in which, if a node was a leaf of the original quasi–forest, then its label contains the proposition t, and nodes both pruned and non existing in the computation tree are labeled by $\bot$. Moreover, since we consider environments that do not block the system, for each node corresponding to a root of the computation quasi–forest associated with an enabled non-terminal environment state, at least one successor is not labeled by $\bot$. The *PD–NBT* $\mathcal{A}_\mathcal{S}$ essentially works in the following way: if it is reading a node x in the top configuration, then:

- if x is either a node labeled by $\bot$ or a leaf of the computation quasi–forest (a node which label contains the proposition t), then the automaton checks that x must have h children all labeled by $\bot$;
- if x corresponds to a state node of the induced module $\mathcal{M}_\mathcal{S}$ with r successors $w_1, \ldots, w_r$, without considering the node from which the system is coming and the jumps to roots, then the automaton checks that x must have r children corresponding to $w_1, \ldots, w_r$ and $h - r$ children labeled by $\bot$;
- if x corresponds to an environment non initial node of $\mathcal{M}_\mathcal{S}$ with p successors $w_1, \ldots, w_p$, not considering the node from which the system is coming and the jumps to roots, then the automaton checks that x must have $0 \leq s \leq r$ children corresponding to nodes from $w_1, \ldots, w_r$ and $h - s$ children labeled by $\bot$;
- if x corresponds to an environment initial node of $\mathcal{M}_\mathcal{S}$ with r successors $w_1, \ldots, w_r$, not considering the node from which the system is coming and the jumps to roots, and s jumps to roots, then then the automaton checks that x must have $0 \leq t_1 \leq r$ children corresponding to nodes from $w_1, \ldots, w_r$, $0 \leq t_2 \leq s$ propositions $\uparrow^o_\alpha$ in its label (with the clause that $t_1 + t_2 \geq 1$), and $h - t_1$ children labeled by $\bot$.

In order to calculate the configuration from which the automaton is coming, we remember in each configuration the previous top configuration and the operation that transformed the previous configuration into the actual one. Then, the *PD–NBT* $\mathcal{A}_\mathcal{S} = \langle \Sigma, \Gamma, \flat, P, p_0, \alpha_0, \delta, P \rangle$ is defined as follows.

- $\Sigma = 2^{AP^* \cup Nom \cup \{t\}} \cup \{\bot\}$;
- $P = Q \times \Gamma \times \{pop, push, sub\} \times Q \times \{\bot, \top, \vdash\}$. Intuitively, the initial pair $Q \times \Gamma$ identifies the previous top configuration, the set $\{push, pop, sub\}$ denotes the operation that transformed the previous configuration in the actual one, and the successive Q is the actual state. The last symbol is used to decide the behavior of the automaton; in particular, if the symbol is $\bot$ then $\mathcal{A}_\mathcal{S}$ can read only the letter $\bot$, if the symbol is $\top$ then $\mathcal{A}_\mathcal{S}$ can read only letters in $2^{AP^* \cup Nom \cup \{t\}}$, and if the symbol is $\vdash$ then $\mathcal{A}_\mathcal{S}$ can read both letters in $2^{AP^* \cup Nom \cup \{t\}}$ and the letter $\bot$. In this last case, it is left to the environment to decide whether the transition to a configuration of the form $((q, A, l, q', \vdash), \alpha)$ is enabled. The three types of (control) states are used to ensure that the environment enables all transitions from enabled system configurations, enables at least one transition from each enabled non-terminal environment configuration, and disables transitions from disabled configurations.

– $p_0 = (q_0, \flat, pop, q_0, \top)$ and $\alpha_0 = \flat$.
– Let us suppose that from the current configuration, the *OPD* allows r transitions $(q_1, op_1, B_1), \ldots, (q_r, op_r, B_r)$ that do not induce jumps to roots and s transitions that induce jumps to roots (without considering one inverse transition that newly induce the configuration from which the *OPD* is coming, if such a configuration exists).

  Then, the transition function $\delta : P \times \Sigma \times \Gamma_\flat \rightarrow 2^{(P \times \{push, pop, sub\} \times \Gamma)^h}$ is defined as follows.

  - if the automaton is reading the letter $\sigma = \bot$, then for $m \in \{\bot, \vdash\}$, $q_{prev}, q \in Q$, $op \in \{push, pop, sub\}$ and $A_{prev} \in \Gamma$, we have

$$\delta((q_{prev},\, A_{prev}\, , op, q, m), \bot, A) = \{\ \langle\ \underbrace{((q, A, pop, q, \bot), pop, A), \ldots, ((q, A, pop, q, \bot), pop, A)}_{h \text{ pairs}}\ \rangle\ \}$$

    that is, if $\mathcal{A}_\mathcal{S}$ is in a node labeled by $\bot$, then it must have h children labeled by $\bot$;
  - if the automaton is reading a letter $\sigma \in 2^{AP^* \cup Nom \cup \{t\}}$ such that $t \in \sigma$, then for $m \in \{\top, \vdash\}$, $q_{prev}, q \in Q$, $op \in \{push, pop, sub\}$ and $A_{prev} \in \Gamma$, we have

$$\delta((q_{prev},\, A_{prev}\, , op, q, m), \sigma, A) = \{\ \langle\ \underbrace{((q, A, pop, q, \bot), pop, A), \ldots, ((q, A, pop, q, \bot), pop, A)}_{h \text{ pairs}}\ \rangle\ \}$$

    that is, if $\mathcal{A}_\mathcal{S}$ is in a terminal node of the computation quasi–forest (a node which label contains t), then it must have h children labeled by $\bot$;
  - if the automaton is reading a letter $\sigma \in 2^{AP^* \cup Nom \cup \{t\}}$ such that $t \notin \sigma$, and it is in a top configuration $(q, B) \notin Env$, then for $m \in \{\top, \vdash\}$, $q_{prev}, q \in Q$, $op \in \{push, pop, sub\}$ and $A_{prev} \in \Gamma$, we have

$$\delta((\ q_{prev}\ , A_{prev}, op, q, m), \sigma, A) = \{\ \ \langle\ ((q, A, op_1, q_1, \top), op_1, B_1), \ldots, ((q, A, op_r, q_r, \top), op_r, B_r), \underbrace{((q, A, pop, q, \bot), pop, A), \ldots, ((q, A, pop, q, \bot), pop, A)}_{h-r \text{ pairs}}\ \rangle\ \}$$

    that is, if $\mathcal{A}_\mathcal{S}$ is in a state node of $\mathcal{M}_\mathcal{S}$, then it has r children labeled by $\top$ and $h - r$ children labeled by $\bot$;
  - if the automaton is reading a letter $\sigma \in 2^{AP^* \cup Nom \cup \{t\}}$ such that $t \notin \sigma$, and it is in configuration $(q, B \cdot \gamma) \notin C_0$ such that $(q, B) \in Env$, then for $m \in \{\top, \vdash\}$, $q_{prev}, q \in Q$, $op \in \{push, pop, sub\}$ and $A_{prev} \in \Gamma$, we have

$$\delta((\ q_{prev}\ , A_{prev}, op, q, m), \sigma, A) = \{\ \ \langle\ ((q, A, op_1, q_1, \vdash), op_1, B_1), \ldots, ((q, A, op_r, q_r, \vdash), op_r, B_r), \underbrace{((q, A, pop, q, \bot), pop, A), \ldots, ((q, A, pop, q, \bot), pop, A)}_{h-r \text{ pairs}}\ \rangle\ \}$$

    that is, if $\mathcal{A}_\mathcal{S}$ is in an environment non initial node of $\mathcal{M}_\mathcal{S}$, then it has r children labeled by $\vdash$ and $h - r$ children labeled by $\bot$;

- if the automaton is reading a letter $\sigma \in 2^{AP^* \cup Nom \cup \{t\}}$ such that $t \notin \sigma$ and σ contains $t \geq 1$ propositions $\uparrow_\alpha^o$, and it is in top configuration $(q, B) \in Env$ then for $m \in \{\top, \vdash\}$, $q_{prev}, q \in Q$, $op \in \{push, pop, sub\}$ and $A_{prev} \in \Gamma$, we have

$$\delta((\, q_{prev}\, , A_{prev}, op, q, m), \sigma, A) =$$
$$\{ \quad \langle\, ((q, A, op_1, q_1, \vdash), op_1, B_1), \ldots, ((q, A, op_r, q_r, \vdash), op_r, B_r),$$
$$\underbrace{((q, A, pop, q, \bot), pop, A), \ldots, ((q, A, pop, q, \bot), pop, A)}_{h-r \text{ pairs}} \,\rangle\, \}$$

  that is, if $\mathcal{A}_\mathcal{S}$ is in an environment initial node of $\mathcal{M}_\mathcal{S}$ with any jump to root enabled, then it has r children labeled by $\vdash$ and $h - r$ children labeled by $\bot$;
- if the automaton is reading a letter $\sigma \in 2^{AP^* \cup Nom \cup \{t\}}$ such that $t \notin \sigma$ and σ contains $t = 0$ propositions $\uparrow_\alpha^o$, and it is in configuration $(q, B \cdot \gamma) \in C_0$ such that $(q, B) \in Env$ then for $m \in \{\top, \vdash\}$, $q_{prev}, q \in Q$, $op \in \{push, pop, sub\}$ and $A_{prev} \in \Gamma$, we have

$$\delta((\, q_{prev}\, , A_{prev}, op, q, m), \sigma, A) =$$
$$\{ \quad \langle\, ((q, A, op_1, q_1, \top), op_1, B_1), \ldots, ((q, A, op_r, q_r, \vdash), op_r, B_r),$$
$$\underbrace{((q, A, pop, q, \bot), pop, A), \ldots, ((q, A, pop, q, \bot), pop, A)}_{h-r \text{ pairs}} \,\rangle$$
$$\vdots$$
$$\langle\, ((q, A, op_1, q_1, \vdash), op_1, B_1), \ldots, ((q, A, op_r, q_r, \top), op_r, B_r),$$
$$\underbrace{((q, A, pop, q, \bot), pop, A), \ldots, ((q, A, pop, q, \bot), pop, A)}_{h-r \text{ pairs}} \,\rangle\, \}$$

  that is, if $\mathcal{A}_\mathcal{S}$ is in an environment initial node of $\mathcal{M}_\mathcal{S}$ with no jumps to root enabled, then it has $r - 1$ children labeled by $\vdash$, one children labeled by $\bot$ and $h - r$ children labeled by $\bot$.

The number of states of $\mathcal{A}_\mathcal{S}$ is $|P| = 9 \cdot |Q|^2 \cdot |\Gamma|$ and the size of the transition function is $|\delta| \leq 9|Q|^2|\Gamma|h^2$.