Rabin's Theorem

Johnson Tan

June 29, 2019

1 Introduction

One of the most powerful results in regards to the decidability of certain logics is Rabin's Tree Theorem. In this report we give an exposition for the proof of Rabin's Tree theorem. We first begin with the notion of automaton that can process infinite strings, in particular we prove McNaughtons theorem a result pertaining to the determinizations of these generalized automaton. After that we study infinite duration games and prove the determinacy of deterministic parity games. Finally we introduce infinite binary trees as well as the appropriate automatons that accept them and prove Rabin's Tree Theorem.

2 Finite Automata

Let Σ be an alphabet, i.e. a finite set of elements which we call letters. We denote the set of all finite strings of letters by Σ^*. Then a finite automaton can be intuitively thought of as a machine that processes these finite strings.

Definition 2.1 (Nondeterministic Finite Automaton)**.** A nondeterministic finite automata M over Σ consists of the following

- A finite set of states Q with a distinguished initial state q_0;
- For each letter $a \in \Sigma$, a set of transitions $\delta_a \subseteq Q \times Q$;
- A set of accepting states $F \subseteq Q$.

In the case that δ_a are functions for all $a \in \Sigma$, we then say that M is a deterministic finite automata.

We now describe how M processes a finite string $w = w_1 \ldots, w_n \in \Sigma^*$,

- Starting at the initial position we read the first letter of w.
- We then change state to $q \in Q$ such that $(q_0, q) \in \delta_{w_1}$. In the case that such q is not unique, we then occupy all possible choices.
- We repeat this process except for all states that is being occupied.
- After performing this step for the last letter, if at least one of the final states is contained in F then we say that M accepts w, otherwise we say that M rejects w.

Given a set of finite strings $L \subseteq \Sigma^*$, we say that a nondeterministic automaton M over Σ recognizes L if for all finite strings $w \in \Sigma^*$,

$$w \in L \iff M \text{ accepts } w.$$

Definition 2.2 (Relational Structure)**.** A relational structure $\mathcal{A}$ consists of

- A set A which we call the domain of $\mathcal{A}$;
- A finite set of relations over A, $R_i \subseteq A^{n_i}$ for $i = 1, \ldots, n$.

Example 1. Let $w \in \Sigma^*$ be a finite word of length n. Then we can view w as a relational structure in the following way

- The domain consists of the set $\{1, \ldots, n\}$;
- For each $a \in \Sigma$, we have the unary relation $R_a \subseteq A$ such that $i \in R_a$ if and only if the i^{th} letter of w is a.
- A binary relation $\geq \subseteq \{1, \ldots, n\}^2$ for the usual ordering on this set.

Given a relational structure on $w \in \Sigma^*$, we can then study it's first order logic. We inductively define the set of first order formulas as the following,

- For $a \in \Sigma$, the statement $a(x)$ where x is a variable;
- The statement $x \geq y$;
- If φ and ψ are valid statements, then so are $\varphi \wedge \psi$,$\varphi \vee \psi$, and $\neg\varphi$.
- For a first order formula $\varphi = \varphi(x)$ for some variable x, then so are $\forall x\varphi(x)$ and $\exists x\varphi(x)$. In this case we say that x is a free variable in φ and is bounded after we quantify over it. We say φ is a first order sentence if it has no free variables.

Given a first order sentence φ, we can consider it's interpretation in w as well as other words in the obvious way.

3 Infinite Automata

Given that we have the notion of automatons that process finite words, it is natural to consider generalized automatons that process infinite words. We can view an infinite word over Σ as an infinite sequence of letters from σ. Then we denote the set of all such infinite words by Σ^ω. Attempting to process an infinite word $w \in \Sigma^\omega$ with a finite automaton as in the case for finite words, we run into the issue of the accepting condition not being well defined.

Definition 3.1 (Nondeterministic Büchi Automata)**.** A nondeterministic Büchi automata A over a finite alphabet Σ consists of

1. A finite set of states Q and an initial state denoted q_0.
2. For $a \in \Sigma$, we have the transition relation $\Delta \subseteq Q \times \Sigma \times Q$.
3. A set of accepting transitions $F \subseteq \Delta$.

Given an infinite word $w \in \Sigma^\omega$, we can process it with A similarly to the finite word case. In this case, an infinite word $w \in \Sigma^\omega$ is accepted by A if there exists a run such that one of the transitions that occurs infinitely often is one of the accepting transitions.

Example 2. Consider the alphabet $\Sigma = \{a, b\}$. Then the following nondeterministic Büchi automata (with accepting edges colored red)

$$\begin{tikzcd}
\text{start} \arrow[r] & q_0 \arrow[loop above, "{a,b}"] \arrow[rr, "b"] & & q_1 \arrow[loop above, "b"]
\end{tikzcd}$$

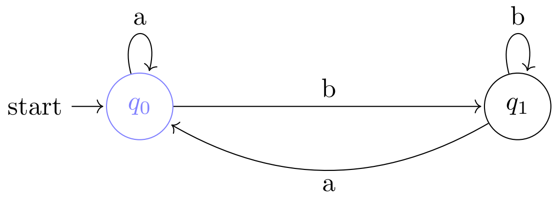
recognizes the ω-language consisting of infinite words with only finitely many appearences of the letter a.

Remark 1. Unlike automata for finite words, the set of languages recognized by deterministic Büchi automata is strictly contained in those recognized by nondeterministic Büchi automata.

Proof. Consider the nondeterministic Büchi automata above and suppose it could be determinized. The set $\{a, b\}^\omega$ can be interpreted as an infinite binary tree with vertices labeled by an element of $\{a, b\}^*$. Then each edge in corresponds to a transition as the automata is deterministic. Furthermore every subtree has an accepting transition as there is an infinite path representing an infinite word of the form $v \cdot b^\omega$. Hence we can produce a path with infinitely accepting edges and occurences of b. $\square$

Definition 3.2 (Nondeterministic Muller Automata)**.** A nondeterministic Muller automata is similar to nondeterministic Büchi automata except that instead of having a set of accepting transitions, we have a collection of subsets of transitions $F \subseteq \mathcal{P}(\Delta)$. An infinite word $w \in \Sigma^\omega$ is accepted if there exists a run such that the set of transitions that occur infinitely often is in F.

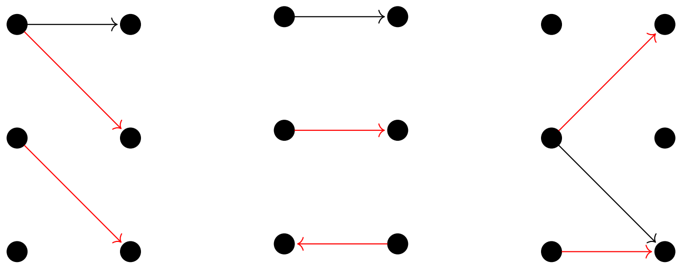
Example 3. Consider the nondeterministic Muller automata below where the a subset is included in F if it consists of edges going into q_0



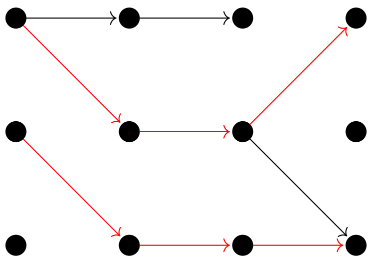
then an infinite word is recognized if and only if there are only finitely many occurences of the letter b.

We note that unlike Büchi automata's, we can determinise Muller automatas. That is given a nondeterministic Muller automata, there exists a deterministic Muller automata that recognizes the same infinite words. In particular one can show that given a nondeterministic Muller automaton there is an equivalent nondeterministic Büchi automaton. Similarly, given a nondeterministic Büchi automaton there is an equivalent deterministic Muller automaton. We first prove the latter result which is commonly known as McNaughton's theorem. To prove McNaughton's theorem, we first show it holds for a special class of regular languages known as universal Büchi languages and then show that we can always reduce to this case.

For $n \in \mathbb{N}$, we define the alphabet $[n]$ to consist of all functions of the form $\{1, \dots, n\} \times \{1, \dots, n\} \to \{\text{no edge, non-accepting, accepting edge}\}$. The letters in $[n]$ should be interpreted as directed balanced bipartite graphs with the edges assigned based off the function. The following are some examples of letters in $[3]$ with the above interpretation



Then with concatenation along the side with n vertices, an infinite words $w \in [n]^\omega$ can be represented by an infinite graph with vertices $\{1, \dots, n\} \times \mathbb{N}$ and edges always go from (p, m) to $(q, m+1)$ with $p, q \in \{1, \dots, n\}$. For example, if we considered the finite word of length 3 given by concatenating the above example, we would have



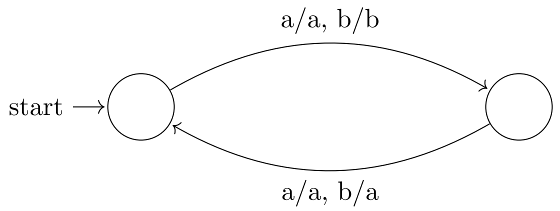
Definition 3.3 (Universal Büchi Language)**.** An infinite word $w \in [n]^\omega$ is in the universal Büchi Language B if as a graph it contains an infinite path starting at $(1, 1)$ such that it contains infinitely many accepting edges.

We now show that we can always transform the language recognized by a nondeterministic Büchi automaton into B. To do this we need the notion of a transducer, which intuitively can be thought of as an automata that sequentially rewrites a word.

Definition 3.4 (Sequential Transducer)**.** A sequential transducer consists between alphabets Σ and Γ consists of a set of states Q and transition functions $\delta_a : Q \times \Sigma \to Q \times \Gamma$ for $a \in \Sigma$.

The transition function is to be interpreted as a regular transition function with the added feature of rewriting the letter from Σ to one from Γ as given by the function.

Example 4. The transducer defined below converts every other letter into the letter a



.

Lemma 1. *Languages recognized by detemrinistic Muller automata are closed under inverse images of sequential transducers, i.e. if A in the diagram below is a deterministic Muller automaton and f is a seuqential transducer, there is a deterministic Muller automaton B which makes the following diagram commute.*

$$\begin{tikzcd}
\Sigma^\omega \arrow[r, "f"] \arrow[dr, "B"] & \Gamma^\omega \arrow[d, "A"] \\
& \{yes, no\}
\end{tikzcd}$$

The above lemma gives us a concrete way to perform our reduction. Suppose we have a nondeterministic Büchi automaton A over Σ and recognizes the language L. We wish to construct a sequential transducer $f : \Sigma^\omega \to [n]^\omega$ such that $f^{-1}(B) = L$. If in addition we can show that B is recognized by a deterministic Muller automata, then by the above lemma L is also recognized by a deterministic Muller automaton.

Without loss of generality assume the states of A are $\{1, \ldots, n\}$ with the initial state represented by 1. Then the transition relation for a letter $a \in \Sigma$ can be represented by an letter in the alphabet $[n]$. Then consider the one state transducer and letter-to-letter conversion described above. Then it is easy to see that $w \in L$ if and only if it's image under this transducer is in B. It suffices now to show that B is recognized by some deterministic Muller automaton. Using the above lemma once again, we reduce to looking at a subset of B.

Definition 3.5. An infinite word $w \in [n]^\omega$ is a tree if every vertice has at most one incoming edge and every edge is part of a path starting from the initial vertex.

We will now construct a sequential transducer that essentially transforms an infinite word $w \in [n]^\omega$ into an equivalent tree in the sense that $w \in B$ implies that $f(w) \in B$. In particular, this transducer will only remove edges from the existing graph.

Definition 3.6 (Profiles)**.** Suppose π is a path in $w \in [n]^\omega$. Then the profile of π is the finite word in $\{$non-accepting, accepting$\}^*$ corresponding to the edges of the path. Under the lexicographic ordering with non-accepting $<$ accepting, we say that π is profile optimal if it is minimal among all paths with the same starting and end vertices.

Remark 2. Let e be the last edge in the path π and ρ the subpath excluding e. Then it is easy to see that if π is profile optimal, then we must also have ρ also be profile optimal.

The process of selecting which edges to keep will be based on the above notion of profile optimal. We now describe the transducer desired. The set of states are all functions from $\{1, \ldots, n\}$ to $\{$non-accepting, accepting$\}$, with the initial state being the function that only sends 1 to accepting. In particular the set of states keep track of which vertices are targets of profile optimal paths. Then the transition are described as follows,

1. If in state q and $a \in [n]$ is read, first remove all edges that do not begin on an accepting vertex.

2. If there are remaining edges that go into the same vertex, remove the ones that are non-accepting.

3. If there are multiple remaining accepting edges that go into the same vertex, choose the one whose source vertex is smallest.

Remark 3. From the construction above, we note that any vertex reachable by a path from the initial vertex in the original path also shares this property in the new graph. Furthermore every finite path in the new graph is profile optimal.

From the above remark it is not unreasonable to believe the resulting tree produced is in B if the original graph was also in B. It remains to show that there is a deterministic Muller automaton that recognizes the subset of B consisting of trees. To do this we employ the same idea used above in that we only need to keep track of a small part of the tree as we process it.

Definition 3.7. Let $t \in [n]^\omega$ and $d \in \mathbb{N}$. An important node for depth d is node that satisfies either of the following:

1. It is the root,
2. a node at depth d,
3. a node which is the closest common ancestor of two nodes at depth d

Given a tree $t \in [n]^\omega$ and $d \in \mathbb{N}$, we can construct a finite tree where the vertices consist of the important nodes at depth d and there is an edge between two vertices if there is a path between them in the original graph. In addition, we say an edge in this case is accepting if the corresponding path has at least one accepting edge and non-accepting otherwise.

We now give a description of the deterministic Muller automaton. The states consist of all possible finite graphs described above including their labellings. The transitions for $a \in [n]$ are given in the following steps:

1. Attach the graph for a to the end of the current state, ex.
2. Remove paths that do not extend to the new maximal depth, ex.
3. Remove unary nodes and collapse the two edges into a single edge, ex.
4. Assign arbitrary identifiers, ex.

With the above definitions, we can have 3 things occur after a transition:

1. edges are deleted during 2 and 3.
2. multiple edges are collapsed into a single edge during step 3. In the case that one of the edges besides the first one is an accepting edge, then we say that the identifier is refreshed.
3. Nothing could happen.

The following lemma gives us an equivalence between a tree being accepted into B and the 3 events described above.

Lemma 2. *For every tree in $[n]^\omega$, the following are equivalent:*

- *The tree contains a path from the root with infinitely many accepting edges;*
- *some identifier is deleted finitely often but refreshed infinitely often.*

Proof. The backwards direction is straight forward. An identifier corresponds to a path in the original graph and a refreshing of an identifier corresponds to the path being appended by an accepting edge. Hence if the identifier is deleted only finitely many times implies that eventually the path associated to its is fixed, and being refreshed gives us that it is in B. □

Thus we have to express the second property in terms of a Muller condition. Let $F_i \subseteq P(\Delta)$ consist of all subsets of Δ such that it contains at least one transition that refreshes i and no transitions that delete i. Then let $F = \bigcup_i F_i$.

Theorem 1 (McNaughton)**.** *For every nondeterministic Büchi automaton there exists an equivalent deterministic Muller automaton*

To complete the determinization of Muller automatons, we need to show that the languages recognized by nondeterministic Muller automatons are also recognized by nondeterministic Büchi automatons. Suppose we have nondeterministic Muller automaton with states Q, transition relation Δ, and accepting condition $F = \{F_0, \ldots, F_n\}$. Consider the nondeterministic Büchi automaton with:

- States $Q \cup \bigcup_{i=0}^{n} \{i\} \times F_i \times 2^{F_i}$
- Transitions consist of 3 parts
  1. The original transitions Δ,
  2. If the automaton reads the letter a while in state q and there is a transition in Δ into $q' \in F_i$, then we include a transition into $(i, q', \varnothing)$.
  3. If the automaton reads the letter a while in state (i, q, R) and there is a transition in Δ into $q' \in F_i$, then we include a transition into $(i, q', R \cup \{q\})$ if $R \neq F_i$ and $(i, q', \varnothing)$ otherwise.
- The accepting condition is $F' = \bigcup_{i=0}^{n} \{i\} \times F_i \times \{F_i\}$

Then as we process an infinite string with this automaton, we are keeping track of which accepting set of states are being fully visited. After fully visiting all the states in an accepting set, we clear the memory and start over. It follows that a string is accepted in this automaton if and only if the set of states it visits infinitely often is precisely one of the accepting states.

4 Infinite Duration Games

The goal of this section is to prove the memoryless determinancy of deterministic parity games. In particular, we prove the more general Büchi-Landweber theorem which pertains to the determinancy of infinite duration games with ω-regular winning conditions.

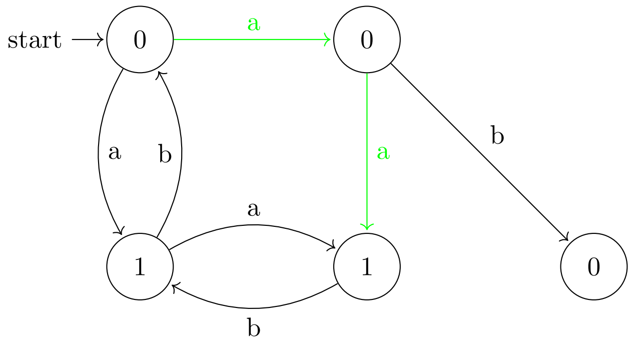
Definition 4.1 (Games)**.** An infinite duration game between 2 players consists of the following data,

1. A directed graph G with a distinguished vertex v_0. The vertices of G will be denoted $\text{Pos}(G)$ and the distinquished vertex to be thought of as the initial position.
2. A partition of $\text{Pos}(G)$ into groups controlled by each player, i.e. $C : \text{Pos}(G) \to \{0, 1\}$.
3. A labeling of the edges of G by symbols in a finite alphabet Σ, i.e. $L : E(G) \to \Sigma$.
4. A function $W : \Sigma^\omega \to \{0, 1\}$ to be interpreted as the winning condition for the infinite duration game. We are particularly interested in the case that W is ω-regular

The game G is played as follows. The game starts at the initial position with the player that controls it choosing an outgoing edge from it. The player that controls the position that this edge leads to chooses an outgoing edge from that position. This process is repeated indefinitely unless a position is reached such that there is no outgoing edge, i.e. a dead end, in which case the other player wins. In the case that the game continues on indefinitely, then this produces an infinite string of characters from Σ, in which case the game is decided by the winning condition W.

In order to make the notion of playing an infinite duration game well-defined, we make the assumption that the players will always make use of a strategy. To be precise a strategy for player i is a function $S_i : \text{Pos}(G) \times E(G)^* \to \text{Pos}(G)$ such that $S_i(v, w) \in N(v)$, where $N(v)$ are the neighbours of v. Then given strategies for both player, there exists a unique game that occurs. We say a strategy for player i is winning if it always wins regardless of the other players strategy. Finally we say that a game is determined if one of the players has a winning strategy.

Example 5. Consider the following game over $\Sigma = \{a, b\}$ with winning condition for player 0 if the label a appears infinitely often.



Then a winning strategy for player 0 is given by always using the green transitions whenever he can. In fact as long as player 0 does not use the transitions on the right labeled b that leads to a dead end, then he will always win.

Definition 4.2 (Finite memory strategy)**.** Let G be a game. Then a strategy for player i with memory M is given by:

- a deterministic automaton with states M and input alphabet V; and
- for every position $v \in \mathrm{Pos}(G)$, a function $f_v : M \to N(v)$.

A strategy is then given by inputting the current string into the DA, then using the function f_v to decide which position to move to. In the case where M is a single state, then we say that the strategy is memoryless. We note that the functions f_v for positions not controlled by player i are not needed in this case.

Theorem 2 (Büchi-Landweber)**.** *Let Σ be finite and an ω-regular winning condition. Then there exists a finite set M such that for every game with the same winning condition, one of the players has a winning strategy that uses memory M.*

We first reduce to looking at parity games.

Definition 4.3 (Parity condition)**.** A parity condition is any function of the form

$$w \in I^\omega \mapsto \begin{cases} 0 & \min(\inf_{\text{word}} w) = 0 \mod 2, \\ 1 & \text{else} \end{cases}$$

for some finite $I \subseteq \mathbb{N}$. A parity game is a game where the winning condition is a parity condition.

Definition 4.4. A deterministic partiy automata is a deterministic parity automata such that the states are ranked by integers, and a run is accepting if the smallest rank appearing infinitely often is even.

Lemma 3. *Every deterministic Muller automata is equivalent to a deterministic parity automata.*

Proof (Lemma). We first show that given a deterministic Muller automaton M, we can construct a deterministic parity automata that recognizes the same language.

Claim 1. *For every finite alphabet Σ, there exists a deterministic automaton with input alphabet Σ, a totally ordered state space Q, and a function $g : Q \to \mathcal{P}(\Sigma)$ with the following property. For every input word, the set of letters appearing infinitely often in the input is obtained by applying g to the smallest state that appears infinitely often in the run*

We apply the claim in the following way, take Σ to be the states S of M. Then there exists a deterministic automaton A with set of states Q and function $g : Q \to \mathcal{P}(S)$ described above. We can view this new automaton as one over the alphabet Γ of M. In particular, as the M process through $w \in \Gamma^\omega$ we get an infinite word $w' \in S^\omega$. Then using g we can rank the states Q such that it satisfies our Muller condition.

It remains to show that every deterministic parity automata is equivalent to a deterministic Muller automata . Let P be a deterministic Muller automata. Consider $F \subseteq \mathcal{P}(\Delta)$ consisting of sets where the minimum rank of all states which are targets for these transition is even. Then using the same states and transition as P with this accepting condition, it is easy to see that this deterministic Muller automata recognizes the same language as P. □

Proof (Reduction). Let G be a game with an ω-regular winning condition recognized by a deterministic parity automata P with I the set of all possible ranks of P. Define a new game denoted the product game $G \times P$ as follows

- The positions consist of pairs of positions from G and P, i.e. $V(G) \times V(P)$.
- There is an edge $(v, q) \xrightarrow{b} (w, p)$ with $b \in I$ if there is an edge $v \xrightarrow{a} w$ such that the state q moves to state p when reading a with b being the rank of q.
- The initial position and which positions each players control are inherited from G.
- Finally the winning condition is given by the parity condition on I.

Claim 2. *For every $v \in V(G)$ the following are equivalent:*

1. *player i wins from position v in the original game;*
2. *player i wins from position (v, q) in the product game, where q is the initial state of the deterministic parity automata recognizing $L = Win^{-1}(i)$.*

We note that since the product game is a deterministic parity automata, then using the assumption that there is a memoryless strategy for it and the above claim we have a strategy for G with memory P. In particular this is done by viewing the product game as both G and P running simutaneously.

□

We now prove the memoryless determinancy of parity games. The proof is by induction on the number of ranks on the position of the parity game. We note that the base case of no ranks is vacuously true. For the induction step, we need notion of attractors.

Definition 4.5. Let $X \subseteq E(G)$, then define the i-attractor of X inductively as follows

1. Define $X_0 = \varnothing$;
2. For an ordinal $\alpha > 0$, let X_α be the set of all positions such that
   (A) it belongs to X_β for some $\beta < \alpha$
   (B) is controlled by the opponent and every outward edge is in X or goes to a position in (A);
   (C) is controlled by player and has at least one outward edge in X or goes to a position in (A).

Then the i-attractor of X is the stabilizing set. We note that this is well defined as our graph is finite.

Remark 4. Suppose that the current position of the game is in the i-attractor of X. Then player i has a memoryless strategy that guarantees after a finite step the game will use an edge from X or a dead end. That is at every position controlled by player i, move to the neighbouring position that is contained in X_α with α minimal.

Let P be a parity game and assume that the minimal rank is 0. For $i \in \{0, 1\}$, define W_i to be the set of positions where player i has a memoryless winning strategy if starting from that position and $U = V(P) \setminus (W_1 \cup W_2)$.

Lemma 4. *Let $i \in \{0, 1\}$. There is a memoryless strategy σ_i for player i, such that if the game starts in W_i, then player i wins by playing σ_i.*

Proof. By axiom of choice there exists a well-ordering of the vertices in W_i. For $w \in W_i$, define it's companion be the least position v such that the strategy for v also works for w. Define the consolidated strategy as follows: when in position w, play according to the strategy of the companion of w. Then the sequence of companions is non-increasing hence must eventually stabilize at some companion v. This strategy is then winning as $v \in W_i$. □

Consider the game where we restrict P to only positions in U. Note that this restriction does not create any new dead ends. Suppose a position has all its outgoing edges to $W_0 \cup W_1$, then in the case that all of them go into W_i for some i is trivial. For the mixed case, then the player that controls that position can always choose to move to a position he controls in which case he has a winning strategy. Then taking X to be the set of all rank 0 edges in U, let A be the 0-attractor for X. Any position controlled by player 1 only has outgoing edges into $A \cup W_0$, as otherwise that position would have originally been in W_1. It follows that if the initial position is contained in A then player 0 can always use the attractor strategy to force a move that uses and edge in U with rank 0 or in W_0.

Define the following game H:

- Restrict G to positions and edges in $U \setminus A$; and
- remove all remaining rank 0 edges, i.e. those controlled by player 1.

Then this new game has 1 less rank than the original, hence we can apply the induction hypothesis to partition $U \setminus A$ into sets U_i such that player i has a memoryless winning strategy on U_i in H.

Claim 3. *U_1 is empty*

Proof. We construct a memoryless winning strategy for player 1 when starting with a node in U_1, hence U_1 must have initially been included in W_1. Consider the following memoryless strategy for player 1

- When in U_1, use the strategy inherited from H. In this case there are two possible outcomes. Either the game stays in U_1 in which case player 1 wins, or eventually the game visits a position from $A \cup W_0 \cup W_1$ or an edge with rank 0 controlled by player 1.
- If in W_1, use the attractor strategy described before.

We claim that if starting in U_1 with the above strategy, then the game will either stay in U_1 or eventually there will be some play that lands in W_1. We first note that all transitions in U that are ranked 0 originate from positions owned by player 1. Since we removed these transitions for our strategy when in the restricted game, it follows that we never use an edge of rank 0 with both endpoints in U. Furthermore a similar argument shows that we never land in $A \cup W_0$. Thus this is a winning strategy for player 1, hence U_1 must have originally been contained in W_1. □

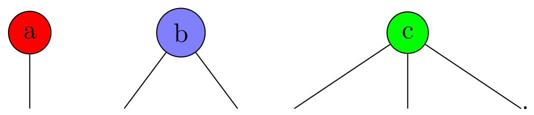
It follows by combining the individual strategies for each partition of the positions of the game, player 0 has a winning strategy when starting in all positions except for those in W_1. This gives us that deterministic parity games are determined by a memoryless strategy.

5 Monadic Second Order Logic of Infinite Trees

We now look at the full generalization of finite automatons for infinite trees. Similar to the case for infinite words, we run into issues when trying to directly apply previously defined finite automatons onto infinite trees. In particular, we no longer have a good intuition for how a machine should process an infinite tree.

Definition 5.1. A ranked alphabet Σ is an alphabet such that associated to each letter is an element of $\mathbb{N}$ known as it's rank or arity. A infinite tree over Σ is a tree such that a letter is assigned to each vertex such that the number of children is equal to it's rank.

Example 6. Let $\Sigma = \{a/1, b/2, c/3\}$ be a ranked alphabet. These alphabets can be visualized as nodes with branches as follows



Then every infinite tree can be produced by concatenating these nodes onto the ends of each other. In the case that all of the ranked alphabets have rank 1, then an infinite tree is precisely an infinite word. As such we may consider extending the notion of an infinite word automaton to one for infinite trees.

Definition 5.2 (Nondeterministic Parity Tree Automaton)**.** A nondeterministic parity tree automaton M over a ranked alphabet Σ consists of

- A finite set of states Q with a distinguished root state;
- A parity ranking function $Q \to \mathbb{N}$;
- For $a \in \Sigma$ of rank n, a transition relation $\delta_a \subseteq Q^n \times Q$.

We say that an infinite tree $t \in \Sigma^\omega$ is accepted by M if there is an assignment of states to the nodes of t such that it satisfies the following conditions:

1. Suppose we have a node labelled $a \in \Sigma$ of rank n. If it's i^{th}-children was assigned the state q_i and itself assigned state q, then we must have that $(q_1, \dots, q_n, q) \in \delta_a$.
2. There exists an infinite branch π such that the maximal parity among those states that occur infinitely often is even.

Example 7. We provide some intuition on the above definition. Let Σ be an unranked finite alphabet and $\mathcal{A}$ a nondeterministic parity automaton over Σ. In this case, as $\mathcal{A}$ processes an infinite word $w \in \Sigma^\omega$ we get a sequence of states of A for a specific run, i.e.

$$w_1 w_2 w_3 w_4 w_5 \cdots \Longrightarrow q_0 q_1 q_2 q_3 q_4 q_5 \dots,$$

where q_0 is the initial state. In particular, we point out that for the case where Σ is a ranked alphabet consisting of only rank 1 letters, then a nondeterministic parity tree automaton is precisely a nondeterministic parity automaton.

Example 8. Let $\Sigma = \{a/2, b/2\}$, then the set of all trees containing at least one node that is labelled a is recognizable. Consider the following infinite tree automaton,

- Two states $Q = \{Y/0, N/1\}$;
- The root is N;
- The transition relation is given by the table

	a	b
(N,N)	N	N
(N,Y)	-	-
(Y,N)	-	-
(Y,Y)	N,Y	Y

We defer the proof that this recognizes precisely the desired trees as it will follow directly from a later result.

Before we state Rabin's theorem, we need to first introduce the notion of studying the logic of a tree. In particular from now on we focus only on binary trees, i.e. letters in our ranked alphabet have arity 2. Given a tree $t \in \Sigma^\omega$, we can view t as an infinite relational structure in the following way,

- The universe of t consists of all the nodes of the tree

- For each $a \in \Sigma$, we have a predicate R_a consisting of all nodes which have the label a.
- We have the binary relation $child_R$ consisting of all pair of nodes (x, y) such that x is the right child of y. Similarly we define the binary relation $child_L$ for the case of left child.

For our case, we will be interested in studying the monadic second order logic of these relational structures. Monadic second order logic is the same as first order logic except we allow ourselves to quantify over sets. In particular, we syntactically define an MSO formula as follows,

- If φ is a first order formula, then it is also an MSO formula;
- The formula $x \in X$, where x is a first order variable and X is a variable representing a set;
- If $\varphi(X)$ is an MSO formula (i.e. it contains a set variable), then $\forall X \varphi(X)$ and $\exists X \varphi(X)$ are also MSO formulas.

We note that one might consider the case where we alternate between quantifying over variables and sets, ex. $\forall x \exists X \exists y \forall Y$. One can show that every MSO formula is equivalent to one where we begin with a string of quantification over sets, followed by a string of quantification over elements, then finally a formula with no bounded variables. From this we say that a collection of infinite trees $L \subseteq \Sigma^\omega$ is MSO definable if there is an MSO sentence φ such that it is true in $t \in \Sigma^\omega$ if and only if $t \in L$.

Theorem 3 (Rabin's Theorem)**.** *The following conditions are equivalent for every set of infinite trees over a finite ranked alphabet where all letters have arity* 2*:*

1. *Definable in MSO;*
2. *recognized by a nondeterministic parity tree automaton.*

We first give a sketch of the proof for the backward direction, that is given a nondeterministic parity tree automaton $\mathcal{A}$ there is an MSO sentence φ such that $t \in \Sigma^\omega$ is accepted by $\mathcal{A}$ if and only if φ is true in t. Suppose $\mathcal{A}$ has states $Q = \{Q_0, \dots, Q_n\}$ with root state Q_0, transitions δ_a, and parity ranking $P : Q \to \mathbb{N}$. Then a tree being accepted by $\mathcal{A}$ can be interpreted as a colouring of of nodes by the states

$$\exists X_0 \cdots \exists X_n$$

such that every node has exactly one state

$$\forall x \bigvee_{q \in \{0,\dots,n\}} (x \in X_q \wedge \bigvee_{p \neq q} x \notin X_p),$$

the root is given the root state

$$\forall x \,\mathrm{root}(x) \implies x \in X_0,$$
$$\mathrm{root}(x) = \neg \exists y (child_R(x, y) \vee child_L(x, y)),$$

for every node, a transition of the automaton is used

$$\bigwedge_{a \in \Sigma} \forall x \, a(x) \implies \bigvee_{(q_1, q_2, q) \in \delta_a} (x \in X_q \wedge child_L(x) \in X_{q_1} \wedge child_R(x) \in X_{q_2},$$

$$child_L(x) \in X_{q_1} = \exists y \in X_{q_1}(child_L(y, x)),$$
$$child_R(x) \in X_{q_2} = \exists y \in X_{q_2}(child_R(y, x)),$$

there is an infinite branch containing the root

$$\exists X (\forall x, y \in X (x \leq y \vee y \leq x) \wedge \exists z \in X (\forall w (w \leq z)) \wedge \forall v \in X (\forall u (child_L(u, v) \vee child_R(u, v) \implies u \in X))),$$

such that the states that the minimal rank among the states that occur infinitely often is even. We defer giving the explicit formulas for the above statement and for the descendant relation $x \leq y$. By construction we have that the trees that satisfy the disjunction of these sentences are precisely those that are accepted by the automaton.

We now give the sketch for the other direction. Given an MSO sentence φ, we wish to construct a nondeterministic parity tree automaton $\mathcal{A}$ that accepts precisely those trees where φ is true. We do this by inducting on the length of the MSO formula. We note that inducting on the length of an MSO formula requires us to define what it means for an MSO formula with free variables to be true in the relational structure of an infinite binary tree. Before this we will first define an alternative MSO logic that is equivalent to the one defined above. In particular, this alternative logic consists of only set variables, that is we only allow quantification over sets.

Consider the following set of syntactic sentences generated by the following relations, connectives and quantifications,

$$a(X) \mid \mathrm{sing}(X) \mid child_L(X,Y) \mid child_R(X,Y) \mid X \subseteq Y \mid \exists X\, \varphi \mid \varphi \wedge \psi \mid \neg\varphi.$$

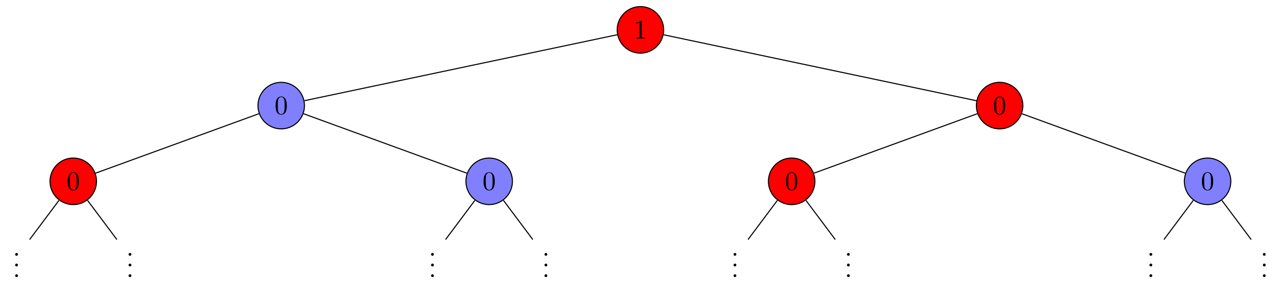
We interpret $a(X)$ for $a \in \Sigma$ in $t \in \Sigma^\omega$ as follows, X is contained in this relation if and only if $X = \{x\}$ and x is contained in the relation in the original logic. The child relations are defined the same way. The $\mathrm{sing}(X)$ consists of all the singleton sets. Then one can show that the set of trees definable by these sentences are exactly the same as those definable by MSO sentences.

Example 9. We can express the relation $\mathrm{sing}(X)$ by the following MSO formula,

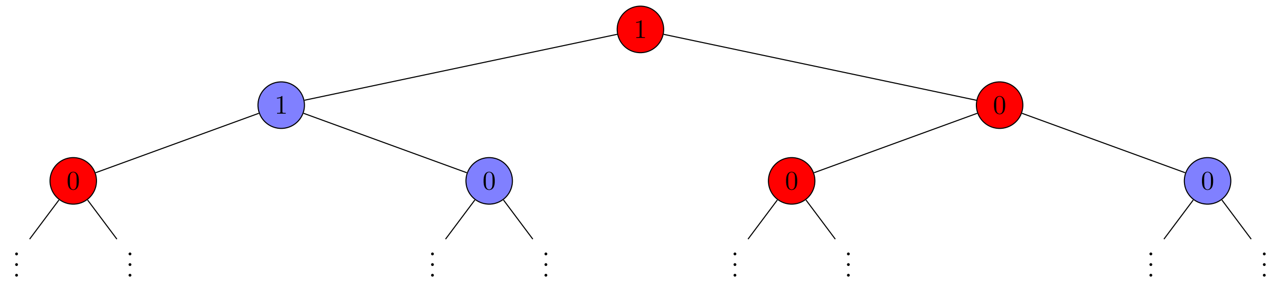
$$\varphi(X) = \forall x \in X \forall y \in X (x = y).$$

Given a syntactic formula φ from above, which we from now on denote as an MSO' formula, we want to define what the set of trees defined by φ is. In particular, the majority of the work comes from the case where we have free variables in the formula. Let Σ be a fixed binary ranked alphabet and $\varphi = \varphi(X_1, \ldots, X_n)$ be an MSO' formula. Define the binary ranked alphabet $\Gamma = \Sigma \times \{0,1\}^n$, then we can evaluate φ in t with the set variable X_i replaced with the set of all nodes which were labeled with a letter whose projection onto the i^{th} coordinate is 1. Then denote the language defined by φ to be the set of all trees over Γ that when evaluated to true. We note that in the case that φ has no free variables, the above definition is our usual definitions.

Example 10. Consider the alphabet $\{a, b\}$ and let $\varphi(x) = a(X)$. Then the tree



where the only node labeled 1 is the root is evaluated to true. We note that the location of the 1 is important, as if we consider the same tree but with 1 at a different position or in additional positions then it may not evaluate to true. This is shown by the tree below which are given the same color labeling but not the same number labeling.



We now restate our desired implication using these new definitions. Given a MSO' formula $\varphi(X_1, \ldots, X_n)$ over a binary ranked alphabet Σ, there exists a nondeterministic parity tree automaton $\mathcal{A}$ over $\Sigma \times \{0,1\}^n$ such that the following are equivalent for all $t \in (\Sigma \times \{0,1\}^n)^\omega$,

1. $\varphi(X_1, \ldots, X_n)$ is evaluated to true in t;
2. $\mathcal{A}$ accepts t.

We now prove this by induction on the length of MSO' sentences. We first consider the base case consisting of the formulas $a(X)$, $\text{sing}(X)$, $child_L(X,Y)$, $child_R(X,Y)$, $X \subseteq Y$.

1. $\varphi(X) = a(X)$

   Since $a(X)$ has a single free variable, we need to build an automaton over the alphabet $\Sigma \times \{0,1\}$. Consider the automaton $\mathcal{A}$ given by

   - We have states $Q = \{Y/0, N/0\}$ with the root state being N;
   - The transition relation δ is given by the table

	(a,0)	(a,1)	(b,0)	(b,1)
(N,N)	N	-	N	-
(N,Y)	-	-	-	-
(Y,N)	-	-	-	-
(Y,Y)	Y	N	Y	-

   It is easy to see that any tree that is accepted by $a(X)$ is accepted by this automaton. In fact for these trees there is only one possible assignment of states to the tree. The other direction requires a slightly more complicated justification but is similar in that it amounts to checking which transitions relations are allowed.

2. $\varphi(X) = \text{sing}(X)$

   We note that $\text{sing}(X)$ can be expressed as $\bigvee_{a\in\Sigma} a(X)$

3. $\varphi(X,Y) = child_L(X,Y)$

   Consder the automaton over $\Sigma \times \{0,1\}^2$ given as follows

   - We have states $Q = \{Y/0, N/1, N'/1\}$ with the root state given by N;
   - For our transition table we note it will be independent of the choice of labeling in Σ. As such we will only describe the transitions for (a,i,j) where $i,j = 0,1$.

	(a,0,0)	(a,0,1)	(a,1,0)	(a,1,1)
(N,N)	N	-	-	-
(N,N')	-	-	-	-
(N,Y)	-	-	-	-
(N',N)	-	N	-	-
(N',N')	-	-	-	-
(N',Y)	-	-	-	-
(Y,N)	-	-	-	-
(Y,N')	-	-	-	-
(Y,Y)	Y	-	N'	-

4. $\varphi(X,Y) = X \subseteq Y$

   Consider the automaton as follows

   - We have a single state $Q = \{Y/0\}$;

- As above our transition is independent of the choice of labeling in Σ.

	(a,0,0)	(a,0,1)	(a,1,0)	(a,1,1)
(N,N)	Y	-	Y	Y

We now consider the cases where we have to apply the inductive hypothesis. In particular from now on we assume that φ and ψ have equivalent nondeterministic parity tree automatons. For the case of negation, we require a more length discussion as it requires the most work.

1. $\exists X\, \varphi(X)$

   Suppose we have an automaton $\mathcal{A}$ for $\varphi(X_1, \dots, X_n)$ and we desire an automaton for $\exists X_1\, \varphi(X_1, \dots, X_n)$. Then define the automaton $\mathcal{B}$ consisting of the same states and assignment of ranks and root states. Let $\delta_{(a,i_1,i_2,\dots,i_n)}(q_1, q_2)$ correspond to the set of states in the transition table for $\mathcal{A}$ in the row for states (q_1, q_2) and column for label $(a, i_1, i_2, \dots, i_n)$. Then the transition table set of states in the transition table for $\mathcal{B}$ in the same row and column for label $(a, i_2, \dots, i_n)$ is $\delta_{(a,0,i_2,\dots,i_n)}(q_1, q_2) \cup \delta_{(a,1,i_2,\dots,i_n)}(q_1, q_2)$. We note that combining this result with the result for the base case of $a(X)$, then we immediately get the result from example 8.

2. $\varphi \vee \psi$

   In the case where both formulas share the same free variables, then all that needs to be done is to consider automaton where we the states consist of the union of the states for the automatons for φ and ψ and setting the root to be the same. Then we just replicate the automatons on their corresponding set of states.

For the case of taking the negation of an MSO' formula, we need to show that the complement of the language recognized by a nondeterministic parity tree automaton is also recognized by one. To do this we consider an equivalent infinite tree automaton such that we can take the complement of.

Definition 5.3 (Alternating Parity Tree Automaton)**.** An alternating parity tree automaton $\mathcal{A}$ over a binary ranked alphabet Σ consists of the following information

- A set of states Q such that we have a parity ranking $Q \to \mathbb{N}$ and an ownership assignment $Q \to \{0, 1\}$;
- A distinguished root state;
- for each $a \in \Sigma$, we have a transition relation $\delta_a \subseteq Q \times \{\varepsilon, 0, 1\} \times Q$.

Given $t \in \Sigma^\omega$, define the deterministic parity game $\mathcal{G}_\mathcal{A}(t)$ with states given by the product of Q with the nodes of t and parity and ownership inherited from Q. We describe the transitions in $\mathcal{G}_\mathcal{A}(t)$ as follows. Suppose we are in state (q, v) and we read the letter $a \in \Sigma$, then if $(q, x, p) \in \delta_a$ there is a transition labeled a that goes to state $(p, v \cdot x)$ where $v \cdot x$ consists of moving to the left (or right) child if $x = 0(or1)$ and staying in the same node if $x = \varepsilon$. Then we say that $\mathcal{A}$ accepts t if player 0 has a (memoryless) winning strategy in $\mathcal{G}_\mathcal{A}(t)$.

Theorem 4 (Dealternation Theorem)**.**

1. *For every nondeterministic parity tree automaton, one can compute an alternating one that recognises the same language.*
2. *Languages recognised by alternating parity tree automata are closed under complement.*
3. *For every alternating parity tree automaton, one can compute an nondeterministic one that recognises the same language.*

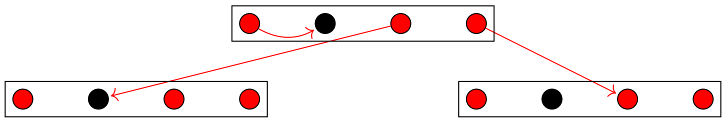
Thus closure under complementation follows immediately after the above theorem. We note that complementing the language recognized by a alternating parity tree automaton amounts to just reversing the assignment of ownership for the states and increasing all the parity rankings by 1. Given a nondeterministic parity tree automaton, we construct an alternating parity tree automaton that recognizes the same language by simulating the original automaton when playing $G_{\mathcal{A}}(t)$.

Let $\mathcal{A}$ be a nondeterministic parity tree automaton with states Q. Define the the alternating automaton $\mathcal{A}'$ such that it has states $Q \cup Q^2$. The initial state is the root state of in Q, with player 0 controlling the states in Q and player 1 the states in Q^2. We describe the transitions as follow

- While in (q, v), player 0 can use the $a \in \Sigma$ labelled transition to the state $((p, r), v)$ if $(p, r, q) \in \delta_a$.
- While in $((p, r), v)$, player 1 can use the $b \in \Sigma$ labelled transition to the state $(p, v \cdot 0)$ or $(r, v \cdot 1)$

We assign the parity ranking to Q inherited the original automaton and to Q^2 the minimum of the pair. In particular, we have that the ranking of the states in Q^2 do not affect which player wins the game as it will never be the maximum.

Now suppose that $\mathcal{A}$ is an alternating parity tree automaton over Σ. Recall that a tree $t \in \Sigma^\omega$ is accepted by $\mathcal{A}$ if and only if player 0 has a memoryless winning strategy in $G_{\mathcal{A}}(t)$. Define the alphabet Γ to be the set of all functions from states controlled by player 0 to $Q \times \{\varepsilon, 0, 1\}$. The following a visual example of such a letter,



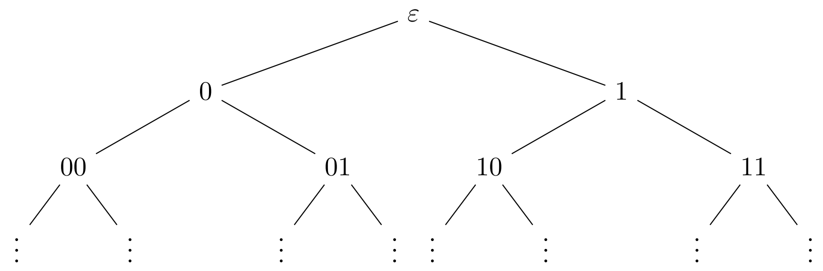
where player 1 controls the red nodes. Then we can represent a memoryless strategy for $G_{\mathcal{A}}(t)$ as a tree over Γ as follow: label node v with the function that maps state q to (p, x) such that the strategy goes from (q, v) to $(p, v \cdot x)$. We take the following claim without proof, although we note that the intuition behind the proof is based on viewing an infinite branches of such a tree as an infinite word. Then build a nondeterministic parity automaton that reads these branches and checks if the the strategy stays on said branch and fails the accepting condition. Apply McNaughton's theorem to receive an equivalent deterministic parity automaton. Finally build a nondeterministic parity tree automaton that runs the previous automaton on each branch.

Claim 4. *The language $\{(t, \sigma) : \sigma$ is a memoryless strategy for player 0 in $G_{\mathcal{A}}(t)\}$ is recognized by a nondeterministic parity tree automaton.*

A quick application of the equivalence between MSO sentences and nondeterministic parity tree automata is the following decidability result.

Corollary 1. *The monadic second order theory of the infinite binary tree is decidable.*

Proof. We first note that the infinite binary tree, which we denote by T_2, in this case is the relational structure with universe $\{0, 1\}^*$, and binary relations $child_L$ and $child_R$ for child relations.



Then we can naturally relate T_2 to be the unique tree over a binary ranked alphabet with a single letter, i.e. $\{a\}^\omega = \{T_2\}$. Let φ be an MSO sentence and L_φ the set of trees where φ is true. We note that we must either have $T_2 \in L_\varphi$ or $L_\varphi = \varnothing$. By Rabin's theorem there exists some nondeterministic parity tree automaton that recognizes L_φ, hence we just need to check whether or not $\mathcal{A}$ accepts any trees. The following claim completes the proof.

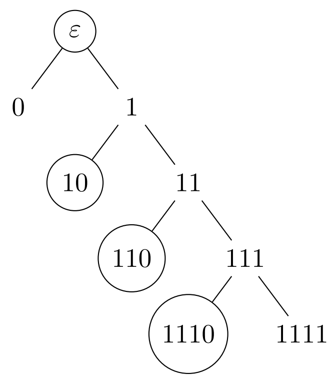
Claim 5. *The emptiness problem of alternating partiy tree automaton (and hence nondeterministic parity tree automaton) is decidable.*

□

We note that with enough work, we may be able to generalize the above results to the case of the infinite ternary tree T_3. Formally T_3 is the relational structure with universe $\{0, 1, 2\}^*$ and a binary relation for each child. However instead one can reduce the decidablility for T_3 to the above decidability result. In particular, we need to find a copy of the infinite ternary tree inside the infinite binary tree in the following way

- There is a bijection from T_3 into a definable subset $T \subseteq T_2$. That is there is some MSO formula with 1 free variable such that membership of a node in T is equivalent to the formula being true when evaluated on sid node;
- There is a translation of formulas for T_3 into formulas in T_2. In particular, we just need to show that every relation in T_3 is equivalent to some appropriate MSO formula.

Proof. Consider the formula $T(x) = \forall Y((x \in Y) \wedge \forall y(((y10 \in Y) \vee (y110 \in Y) \vee (y1110 \in Y)) \implies (y \in Y)) \implies \varepsilon \in Y)$. Nodes that satisfy this formula can be visualized as follows



In particular, we have that the node 10 corresponds to the node 0 in T_3, 110 to 1, and 1110 to 2. The translation for the middle child in terms of a formula with two free variables is $child_M(x, y) = \exists u \exists v(child_R(x, u) \wedge child_R(u, v) \wedge child_L(v, y))$. We note that in the above formulas, $y10 \in Y$ is shorthand for $\forall x(\exists z(child_L(x, z) \wedge child_R(z, y)) \implies x \in Y)$ □

References

[1] Wojciech Czerwiński Mikoł aj Bojańczyk. An automata toolbox, February 2018.

[2] David Muller and Paul E. Schupp. Alternating automata on infinite objects, determinacy and rabin's theorem. In M. Nivat and D. Perrin, editors, *Automata on Infinite Words*, pages 99–107, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[3] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Bull. Amer. Math. Soc.*, 74(5):1025–1029, 09 1968.

[4] Wolfgang Thomas. *Languages, Automata, and Logic*, pages 389–455. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.